universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

"Kernel Preprocessing in 3D Adjoint Tomoraphy of Local Surface Sedimentary Structures"

verfasst von / submitted by

Jaroslav Valovčan, Bc.

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2019 / Vienna 2019

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 680
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Joint-Masterstudium Physics of the Earth (Geophysics)
Betreut von / Supervisor:	prof. RNDr. Peter Moczo, DrSc.

Affidavit

I hereby affirm that this Diploma Thesis represents my own written work and that I have used no sources and aids other than those indicated. All passages quoted from publications or paraphrased from these sources are properly cited and attributed.

The thesis was not submitted in the same or in a substantially similar version, not even partially, to another examination board and was not published elsewhere.

.....

Bratislava, 2019-07-04 Bc. Jaroslav Valovčan

Acknowledgements

I would like to thank to everyone who was helpful be any means. I am deeply grateful to my supervisor Prof. RNDr. Peter Moczo, DrSc. for his patient guidance and useful critique. I also wish to extend my special thanks to Mgr. Filip Michlík for his invaluable insight into problematic, helpful advice and practical suggestions.

Preface

One of the main goals of modern seismology is to predict ground motion at the site during future earthquakes. It requires both an accurate program for numerical simulation of propagation of seismic waves as well as a precise Earth model in the area through which the seismic waves are propagating. The former is met by various methods which were developed over last decades, the later is still a challenging task. It requires to solve an inverse problem. Although a huge progress has been made in this field, further improvements are still necessary in the global and regional scales but mainly in the local scale addressing strongly heterogeneous structures.

Fundamental mathematical tools were developed relatively long before recent increase of computational power. Therefore, for example often only times of the first arrivals were used for inversion. This was not sufficient especially for local structures. However, the recent improvements of computers have allowed to apply full waveform inversion methods which use as much information from seismic records as reasonable.

The adjoint tomography is a method which belongs to the family of full waveform inversion methods and which utilises the adjoint approach. Kubina et al., 2018 applied this method for inversion of 2D local surface sedimentary structures with promising results. The application to local structures requires a special treatment of kernel preconditioning due to ill-posedness of the problem.

In this thesis, we build on the work of Kubina et al., 2018 and focus on kernel preconditioning in 3D problem. We developed generalised methods for kernel preconditioning. Numerical tests confirmed their applicability. Consequently, they will be implemented in `FDAtom3D`, a program for the adjoint tomography in 3D local surface sedimentary structures created by Filip Michlík.

Contents

Introduction

Earthquakes are the phenomenon which releases the biggest amount of energy in a single event among all natural or artificial processes on the Earth. Therefore, there is a great potential for them to cause a harm. What is more, they often occur in the most populated areas in the world. It is not a coincidence, as those areas belong among places with the most propitious natural conditions to which the earthquakes contributed significantly. Therefore, we must pay them deserved attention.

Being able to forecast the earthquakes is a long-term goal of seismology. It is the biggest challenge, mainly due to structural complexity of the Earth's lithosphere and the complexity of processes of earthquakes preparation. Furthermore, it is impossible to install instruments deep inside the Earth which could monitor those processes directly, which makes earthquakes even more difficult to be predicted. Taken all together, it is not clear whether it will be possible to forecast earthquakes at all.

Having said that, there is a branch of seismology that assesses seismic hazard. It does not attempt to forecast earthquakes. It just evaluates a probability of the area of interest to be affected by an earthquake of a certain magnitude in a given time period.

It is not difficult to estimate a probability of an earthquake occurrence. It can be done simply by a research of historical earthquakes records. The noticeably more difficult task is to estimate a damage caused by an earthquake if it happens to occur. One must realise that not solely a magnitude of the earthquake and its distance determine a seismic ground motion, but also so called site effects must be taken into account, making it even more difficult to predict seismic intensity. It is a task for numerical modelling.

Being able to satisfactorily model ground motion, two crucial requirements must be met. First of all, we need an accurate soft-

ware for numerical modelling of propagation of seismic motion in a realistic medium. Many such programs have been created, so one must only choose which one best addresses his specific problem. Secondly, we need to know physical properties of the medium through which the propagation of seismic waves is to be modelled. They can be obtained by methods of seismic tomography. However, it is a difficult task, especially in local surface sedimentary structures.

Difficulty of determining of physical properties of medium arises from ill-posedness of the problem. Usually, inverse problems suffer from either non-existence or non-uniqueness of a solution. Non-uniqueness is often caused by a lack of data. To overcome these problems, a proper preconditioning is required.

Application of many numerical methods for solving of inverse problems had been limited by their high computational costs for years. It usually resulted in a reduction of amount of data that could have been used for an inversion. Often only high frequency approximations were used and only arrival times were inverted. We do not need to stress that such approximations were of no use for local surface sedimentary structures, as those are characterised by strong heterogeneity and typical wavelengths of these heterogeneities are of order of seismic waves wavelengths or even shorter. These difficulties were overcome only recently by enhancement of computers which allowed a direct application of so-called full waveform inverse methods.

Full waveform inverse methods use as much information from seismic records as reasonable. One of the first who utilised a full waveform inverse method in seismology was Tarantola, 1984. He developed an iterative algorithm for computation of bulk modulus, density and source-time function from seismic records of multiply reflected and refracted waves in an acoustic approximation.

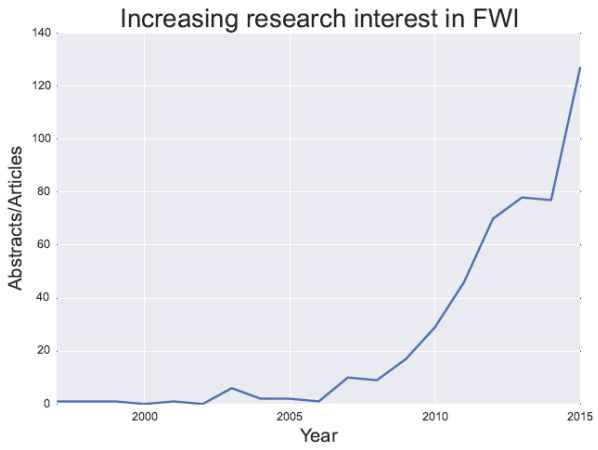


Figure 0.1: Annual appearance of words "FWI" or "Full Waveform Inversion" in articles (taken from Wellington, 2016)

This study was of a great importance not only because it addressed a particular inverse problem, but it also showed that it was possible to apply full waveform inverse methods with a then computational power. Nevertheless, a number of new studies dealing with full waveform inverse methods had been increasing only slightly until about a decade ago, as shown at Fig. 0.1. A sharp increase in last years can be attributed to enhancement of computational power, as well as to a development of an adjoint tomography method.

One of the first applications of the adjoint method in seismology can be found in Bamberger et al., 1979. He addressed a common problem in reflection seismic – determination of density and shear modulus distribution with depth by measurement of horizontal displacement at a free surface. He determined the acoustical impedance of the layered Earth. Further examples can be found in Tarantola, 1988 (incorporation of attenuation), Tromp et al., 2005 (work on kernels), Fichtner; Bunge et al., 2006a (comprehensive review of the method), Fichtner; Bunge et al., 2006b (discussion on sensitivity functionals and sensitivity kernels), Sieminski et al., 2007a (incorporation of anisotropy to inversion from body waves), Sieminski et al., 2007b (incorporation of anisotropy to inversion from surface waves), Q. Liu et al., 2008 (com-

putation of sensitivity kernels for global seismic wave propagation), etc. Probably the first applications of the adjoint method in 3D date back to 2009 when Fichtner; Kennett et al., 2009 applied it for upper-mantle tomography in the Australasian region, Tape et al., 2009 used it for developing of a 3D seismological model of the southern California crust and Stich et al., 2009 made use of it for localization of coda reflectors beneath northern Apennines. Other studies followed immediately, e. g. Tape et al., 2010 (improvement of a 3D tomographic model of the southern California crust), Fichtner; Kennett et al., 2010 (tomography for the Australasian region with incorporated radial anisotropy), Zhu; Bozdağ; Peter et al., 2012 (structure of the European upper mantle), Fichtner; Trampert et al., 2013 (inversion for Eurasia with focus on Anatolia using multiscale approach), P. Chen, 2013 (inversion for the southern California using combined adjoint – scattering-integral method), Rickers et al., 2013 (a high-resolution S-velocity model down to 1300 km depth revealing a presence of two plumes beneath Iceland and Jan Mayen), Zhu; Bozdağ; Duffy et al., 2013 (attenuation beneath Europe and the North Atlantic), Lee; P. Chen et al., 2014 (refinement of a crustal structure of the southern California using combined adjoint – scattering-integral method obtained from almost 5.9 million measurements), M. Chen; Niu; Q. Liu; Tromp, 2015 (shear wave speed model of Mongolia with focus on Hangai Dome), M. Chen; Niu; Q. Liu; Tromp; Zheng, 2015 (radially anisotropic seismic model of East Asia down to 900 km depth), Zhu; Bozdağ; Tromp, 2015 (detailed three-stage inversion for the crust and upper mantle beneath Europe and North Atlantic with attenuation and anisotropy), Lee; P. Chen, 2016 (improvement in determination of basin structures in the southern California), M. Chen; Niu; Tromp et al., 2017 (image of lithospheric foundering and underthrusting beneath Tibet), Y. Liu et

al., 2017 (crustal and uppermost mantle structure beneath NE China obtained by ambient noise adjoint tomography), etc.

Notice that the adjoint method has been used over a great range of scales – from hundreds of kilometres (e.g. southern California) up to a continental scale. Even the first inversions on a global scale were performed (e.g. Bozdağ et al., 2016), despite their high computational costs. However, their resolution does not reach the one of continental-scale inversions yet, but it is close to it in areas with dense coverage of data.

One could assume that it cannot be difficult to employ the adjoint method on a local scale. But it is a big blunder because of several reasons.

First of all, one must realise how a typical local surface sedimentary structure looks like. It is usually several kilometres wide and hundreds of metres deep and it is characterised by strong heterogeneity. Material parameters often vary over a range of several orders of magnitude and a typical length of variation is very short.

That implies that waves used for an inversion in the local surface sedimentary structures must be of short wavelengths, simply because if they had longer wavelengths than the typical length of variation, they would not "see" those short-length heterogeneities. This "invisibility" is exploited in a multiscale approach to minimisation of misfit, however it has some serious drawbacks. If short wavelengths are to be used for the inversion, the model must be fine spatially sampled in order to simulate a propagation of such short-wavelength waves which leads to an increase of computational time.

However, a more serious problem is that only local earthquakes can be used for the inversion because short waves from distant earthquakes may be strongly attenuated. Furthermore, the use of distant sources would increase computational costs excessively.

On the other hand, there is usually a lack of local sources, they are often too weak and their distribution is uneven which results in severe ill-posedness of the problem and a careful preconditioning needs to be employed.

Another serious problem of application of the adjoint method for sedimentary structures is that an initial model is usually poorly determined which causes difficulties to converge to a global minimum of a misfit.

Considering all this together, it is not a surprise that the adjoint method has not been widely employed for the local surface sedimentary structures so far. One of the first attempts (if not the first at all) to change that can be found in PhD Thesis of Kubina, 2017. He proposed an algorithm for application of the adjoint method for the local surface sedimentary structures and performed a blind numerical test for 2D artificial model.

We build on a work of Kubina et al., 2018. We focus on kernel preconditioning, trying to generalise his ideas to 3D and investigating properties and applicability of the generalised methods for kernel preconditioning.

This thesis consists of two main parts. In the first one, we provide a brief insight to inverse problems and a review of the adjoint method. Then in the second part, we focus on kernel preconditioning. We investigate smoothing and applying of a mask in detail. We propose generalisation of Kubina's method for smoothing and of his design of mask to 3D and we perform a numerical tests of them.

Part I

A review of the adjoint tomography method

1 Inverse problems in geophysics

1.1 Forward and inverse problem

A typical problem in geophysics can be described as finding of a relation between observations and physical parameters of a geophysical system. Denote the observations (or data) as d and the physical parameters (or model) as m. Providing we know the physics of the system, we can formulate the relation as

$$G(m) = d, \tag{1.1}$$

where G is an operator/function which relates the data to the model. We can look at it as a mapping from a model space to a data space.

G may be of various forms, such as an ordinary differential equation, a partial differential equation, a system of algebraic equations or something else, m can be a vector of parameters or a continuous function of space and/or time and d can be a vector of discrete data or a continuous function. If m and d are continuous functions, G is called an operator, whilst if they are vectors, G itself is called a function.

If we refer to a problem, we have in mind determination of either m, d or even G in eq. (1.1), whilst the other two are known. Based on what is unknown, we distinguish three types of a problem.

If d is unknown, we talk about *forward problem*. It is usually the simplest case and it commonly requires solving a differential equation, evaluating an integral or performing some well-defined operations. Forward problems are typically well-posed and have a unique solution. It is a problem of predicting observations if we know the physical properties of the system.

A noticeably more difficult problem is determination of unknown m if d is given. It is called an *inverse problem*. It is problem of determination of physical parameters of the system, providing we have a set of collected data. Such a problem is often ill-posed, its solution may be non-unique and it does not have to exist at all due to inaccurate data or not exact physics.

For the sake of completeness, the third problem is a problem of determination of unknown G, providing we have a set of observations d and we know the model m. This kind of problem is called a *model identification problem*. We will not address it here.

We focus on inverse problems. We may need to determine a set of unknown parameters forming a vector $\mathbf{m}$, providing we have a set of measured data represented by vector $\mathbf{d}$. Then we refer to a *discrete inverse problem* (or parameter estimation problem) which can be written as a system of algebraic equations

$$G(\mathbf{m}) = \mathbf{d}. \tag{1.2}$$

And if those equations are linear, then it takes a form of

$$\mathbf{G}\,\mathbf{m} = \mathbf{d}, \tag{1.3}$$

where $\mathbf{G}$ is a matrix of the system of linear equations.

The other type of the inverse problems are so called *continuous inverse problems* when m and d are continuous functions of space and/or time. They are usually discretised and addressed as the discrete inverse problems with lots of unknown parameters $\mathbf{m}$.

1.2 Difficulties with solving of inverse problems

The most straightforward approach to solving of an inverse problem is the application of an inverse operator G^{-1} to the operator G. However, this approach can be used very rarely. Beside

the fact that the inverse operator does not have to exist, it has to cope with another serious problem.

Our measured data d_{observed} usually differ from expected data d_{true} which we would obtain if we applied the operator G to the true model m_{true} because our measured data are affected by noise. That means we are trying to resolve a problem of finding m_{true} from

$$\underbrace{G\left(m_{\text{true}}\right)}_{d_{\text{true}}} + \varepsilon = d_{\text{observed}}, \tag{1.4}$$

where ε stands for the noise and it is impossible by applying of the inverse operator G^{-1}. It is like that because the operator G usually cannot explain noise, nor all phenomena due to some simplifications in physics and therefore we cannot infer a true model m_{true} from noisy data d_{observed} using the inverse operator G^{-1}.

Instead of finding m from eq. (1.4), we rather address different problem – a minimization of an objective function

$$F\left(m\right) = \left\| G\left(m\right) - d_{\text{observed}} \right\|_p . \tag{1.5}$$

$\left\| f \right\|_p$ denotes p-norm of f. Typically 2-norm is used which corresponds to Euclidean length but sometimes other norms are used as well.

A difficulty of addressing inverse problems arises from the fact that the inverse problems are often ill-posed. Ill-posedness of the inverse problems means that they do not match one or more conditions which well-posed problems have to fulfil. A solution of a well-posed problem has to exist, be unique and stable.

Existence of a solution of an inverse problem is closely related to the fact that the observed data contain noise and/or physics of the system is only approximate. That may cause that no model m could fit the observed data exactly and therefore a solution would not exist. In such a case only an approximate solution could be found.

On the other hand, if a solution to the inverse problem exists, it might not be unique. There may be even an infinite number of distinct models which explain observed data. A good example from geophysics is a determination of mass distribution by measuring of external gravitational field in a spherically-symmetric situation.

Last but not least, a condition of stability requires that a small change in input data results to a small change of a solution. This is often not the case when dealing with an inverse problem. It happens quite often that slightly different observed data lead to a completely different inferred model.

Having said that, one can easily understand that instability of the inverse problem in connection with a presence of noise in measured data can result to an inverted model which is not related to a true model, as the inverted model may be dominated by the noise which is completely random and different noise would lead to a totally different inverted model.

In order to stabilise the process of inversion, we usually set additional constraints on a solution or make other modifications, so that we obtain a reasonable solution. We refer to it as regularisation and preconditioning.

1.3 Seismic tomography

We hereby address one particular inverse problem – a problem of imaging of the Earth's interior using seismic waves records. A technique addressing this problem is called seismic tomography.

Seismic tomography is based on a fact that seismic waves propagating through the Earth are affected by its physical properties. Therefore, seismic records reflect inner structure of the Earth.

Tomography can be based on P-waves, S-waves, surface waves or even on ambient noise.

The aim of seismic tomography is to design an Earth's model m represented by spatial variations of physical parameters. Those parameters are density and bulk and shear moduli (alternatively P and S wave velocities). In more advanced tomographies also anisotropy and anelasticity may be incorporated.

As data d, seismic records serve. For many years, only limited information from seismic records could have been utilised due to insufficient computational power of then computers. Mostly, only travel times to many seismic stations of a certain wave phase were used. Nowadays, almost complete records can be used.

A link between a model space and a data space is governed by an elastic wave equation. That implies that dealing with tomography involves solving of this equation.

We are interested in tomographic methods which utilise entire or nearly entire seismic records. Like this, we can gain more information, thus data from a fewer earthquakes and seismic stations are necessary. In such a case we refer to full-waveform inverse methods. They are of a particular importance in case of tomography of local structures where only a little data are available.

1.4 Full-waveform inversion

1.4.1 Principles of full-waveform inverse methods

Full-waveform inverse methods are based on a trial-and-error approach. In fact, strictly speaking, full-waveform inversion involves addressing rather a forward problem than an inverse problem. The idea behind full-waveform inversion is that we consider a lot of trial models m_{trial} and for each of them we predict data $d_{\text{modelled}} = G(m_{\text{trial}})$ which we should observe. These modelled data are then compared with true observed data d_{observed} and a goodness of the trial model is quantified by a *misfit* $\chi(m_{\text{trial}})$. We are trying to determine a model with the lowest possible misfit.

There are two fundamental categories of inverse problems. The first one belongs to a probabilistic inverse theory. Each model m_{trial} from a model space is assigned a probability $p\left(m_{\text{trial}}\right)$ of representing a true Earth's model m_{true}. The probability $p\left(m_{\text{trial}}\right)$ takes into account a misfit related to m_{trial} and a consistency of m_{trial} with our prior knowledge of the Earth's structure. A solution of the inverse problem is then a probability density function $p\left(m\right)$.

The advantage of the probabilistic approach is that we have the entire model space covered which allows us to identify really the best model. However, it requires too many forward computations to sample the entire model space which is computationally intractable and thus this approach is of no use.

The other category belongs to deterministic inverse theory. It is based on an iterative minimisation of the misfit. A computational feasibility is reached for a cost that not the entire model space is probed in detail which may results to a state that an iterative algorithm could converge only to a local minimum of misfit, thus not the optimal model would be found. Additionally, we cannot decide whether the obtained model is unique or whether there exists a different model which can explain the observations as well.

There are two distinct approaches to a deterministic solving of inverse problems. The first one starts with a set of different models. For each model, data are modelled and a corresponding misfit is calculated. The models with the least misfit are selected and combined, so that one model is obtained. This resulting model is then randomly perturbed multiple times by which a new generation of models is obtained. Members of this new generation are closer to the true model than the ones of former generation. After many generations, the resulting model approximates the true model well.

The other approach begins with a single initial model. Data are modelled for this model and corresponding misfit is calculated. Then other models in vicinity are probed in order to find a model with less misfit. It is usually done by calculating of the gradient of misfit and a new model is chosen in an opposite direction to the gradient of misfit. After several iterations, the algorithm should converge to a local minimum. This approach is called an iterative gradient method. An example of the iterative gradient method is a scattering integral method (P. Chen; Zhao et al., 2007, P. Chen; Jordan et al., 2007) or an adjoint method which is introduced in the chapter 2.

1.4.2 Misfit

A misfit $\chi(m)$ is a measure which accounts for discrepancy between modelled data d_{modelled} and observed data d_{observed}. It quantifies a goodness of the model.

We distinguish three levels of misfits:

- a *waveform misfit* is calculated from a single pair of recorded and modelled seismograms;
- an *event misfit* is a sum of all waveform misfits calculated from pairs of recorded and modelled seismograms of a single event;
- an *aggregate misfit* is a sum of the event misfits for a set of events.

It is recommended to minimise the aggregate misfit when searching for the optimal model. Other option is to minimise several waveform or event misfits in sequence. However, a minimisation of one misfit can lead to an increase of another, therefore the former is preferred.

The most commonly used misfit is an L2-misfit

$$\chi(m) = \int_{t_1}^{t_2} \frac{1}{2} \left[d_{\text{modelled}}(m; \mathbf{r}, t) - d_{\text{observed}}(\mathbf{r}, t) \right]^2 \mathrm{d}t, \tag{1.6}$$

where $\mathbf{r}$ is a position vector of the receiver. It is an integral of a half of a square of the difference between modelled and observed data over a certain time window. Presented formula is for a waveform misfit. Other levels of misfit can be easily obtained by summation of waveform misfits.

Another quite common misfit is a cross-correlation time-shift misfit. It is based on a cross-correlation function of modelled and recorded data

$$C(t) = \int_{-\infty}^{\infty} d_{\text{modelled}}(m; \mathbf{r}, \tau) \cdot d_{\text{observed}}(\mathbf{r}, t + \tau) \, \mathrm{d}\tau. \tag{1.7}$$

Maximum of the cross-correlation function corresponds to the difference of arrival times between modelled and observed seismograms

$$\Delta t = \arg\max_t C(t). \tag{1.8}$$

This time shift is then used to quantify the misfit. More precisely, a half of a square of the time shift serves as the misfit

$$\chi(m) = \frac{1}{2} (\Delta t)^2. \tag{1.9}$$

There is a variety of other misfits. Some of them are based on time-frequency analysis of data, e.g. envelope and phase misfit or instantaneous phase and envelope misfit. Others are based on amplitude characteristics.

Sometimes, it is useful to write misfit in a time-space integral form

$$\chi(m) = \int_T \int_V \tilde{\chi}\left[d_{\text{modelled}}(m; \mathbf{r}, t), d_{\text{observed}}(\mathbf{r}, t), \boldsymbol{\rho}\right] \mathrm{d}^3\rho \, \mathrm{d}t, \quad (1.10)$$

where $\tilde{\chi}(m)$ is a misfit function. E. g. for L2-misfit, we have

$$\tilde{\chi}(m) = \frac{1}{2}\left[d_{\text{modelled}}(m; \boldsymbol{\rho}, t) - d_{\text{observed}}(\boldsymbol{\rho}, t)\right]^2 \delta(\boldsymbol{\rho} - \mathbf{r}), \quad (1.11)$$

where $\delta(\boldsymbol{\rho} - \mathbf{r})$ is Dirac delta function. A misfit function for an event L2-misfit can be written as

$$\tilde{\chi}(m) = \sum_{i=1}^{n} \frac{1}{2}\left[d_{\text{modelled}}(m; \boldsymbol{\rho}, t) - d_{\text{observed}}(\boldsymbol{\rho}, t)\right]^2 \delta(\boldsymbol{\rho} - \mathbf{r_i}), \quad (1.12)$$

where $\mathbf{r_i}$ is a position vector of i-th receiver and a summation is performed over all receivers.

2 Adjoint tomography

2.1 Concept of adjoint method

The adjoint tomography is an iterative gradient full-waveform inverse method. It is based on a minimisation of misfit $\chi(\mathbf{m})$ defined in the section 1.4.2. It is achieved by computation of a misfit gradient with respect to model parameters $\boldsymbol{\nabla}_{\mathbf{m}}\chi(\mathbf{m})$. A differential of misfit in the direction $\boldsymbol{\delta}\mathbf{m}$ is defined as

$$\delta\chi(\mathbf{m}) = \boldsymbol{\nabla}_{\mathbf{m}}\chi(\mathbf{m}) \cdot \boldsymbol{\delta}\mathbf{m} = \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left[\chi(\mathbf{m} + \varepsilon\boldsymbol{\delta}\mathbf{m}) - \chi(\mathbf{m})\right]. \qquad (2.1)$$

However, a straightforward computation of the gradient is practically not possible due to the size of a model space. Here the adjoint method comes in handy. It allows us to compute the misfit gradient for large model spaces.

The adjoint method requires a computation of two wavefields. The first wavefield is a regular wavefield $\mathbf{u}(\mathbf{r}, t)$. It is a wavefield corresponding to an initial model $\mathbf{m}$, which is generated by a source which is similar to a true source the most. This wavefield is compared with an observed wavefield $\mathbf{u_0}(\mathbf{r}, t)$ and a corresponding misfit is calculated.

The second wavefield is an adjoint wavefield $\mathbf{u}^{\dagger}(\mathbf{r}, t)$. It is generated by adjoint sources located at the places of receivers., whose source-time functions reflect differences between modelled and observed wavefields. The adjoint wavefield is propagated back in time, so it can be viewed like a propagation of residuals from place where they were recorded to the place of their origin. The adjoint wavefield focuses at the place where the initial model $\mathbf{m}$ differs from the true model.

Regular and adjoint wavefields are used to calculate the misfit gradient. Consequently, the model is updated in an opposite direction of the misfit gradient. As the adjoint wavefield focuses at

the location of discrepancy between initial and true model, the gradient is biggest at those places, thus the model is updated the most therein.

2.2 Regular and adjoint wavefield

Consider a volume V of continuum. A model $\mathbf{m}$ is given by some initial spatial variations of physical parameters within V. We are interested in a computation of a regular displacement wavefield $\mathbf{u}(\mathbf{m}; \mathbf{r}, t)$ due to external forces $\mathbf{f}(\mathbf{r}, t)$ acting as a source. A link between the model $\mathbf{m}$ and the regular wavefield $\mathbf{u}$ is provided by an elastic wave equation together with a set of initial and boundary conditions.

The wave equation can be written symbolically as

$$\boldsymbol{\mathcal{L}}[\mathbf{m}; \mathbf{u}(\mathbf{r}, t)] = \mathbf{f}(\mathbf{r}, t), \tag{2.2}$$

where $\boldsymbol{\mathcal{L}}(\mathbf{m}; \mathbf{u})$ is an operator of wave equation. More precisely, $\boldsymbol{\mathcal{L}}(\mathbf{m}; \cdot)$ is the operator and the former is the operator applied on a function $\mathbf{u}$, but for simplicity, we will refer to the former as the operator, whilst we will stress it if we mean the bare operator.

We require zero initial conditions

$$\mathbf{u}(\mathbf{r}, t)|_{t=t_0} = 0; \tag{2.3a}$$

$$\dot{\mathbf{u}}(\mathbf{r}, t)|_{t=t_0} = 0. \tag{2.3b}$$

and a free-surface boundary condition

$$\mathbf{n} \cdot \boldsymbol{\sigma}(\mathbf{r}, t)|_{\mathbf{r} \in \partial V} = 0. \tag{2.4}$$

Then the regular wavefield is completely determined.

A difference between the regular wavefield $\mathbf{u}(\mathbf{r}, t)$ and the observed wavefield $\mathbf{m_0}(\mathbf{r}, t)$ is quantified by misfit $\chi(\mathbf{m})$ as defined by eq. (1.10). We adopt notation $\langle \cdot \rangle$ for time space integral.

Then misfit can be written in the form

$$\chi\left[\mathbf{u}\left(\mathbf{m};\mathbf{r},t\right),\mathbf{u_0}\left(\mathbf{r},t\right)\right]=\left\langle\tilde{\chi}\left[\mathbf{u}\left(\mathbf{m};\mathbf{r},t\right),\mathbf{u_0}\left(\mathbf{r},t\right)\right]\right\rangle. \tag{2.5}$$

We are interested in the differential of misfit $\delta\chi\left(\mathbf{m}\right)$. It can be written as

$$\delta\chi\left(\mathbf{m}\right)=\boldsymbol{\nabla}_{\mathbf{m}}\chi\left(\mathbf{m}\right)\cdot\boldsymbol{\delta}\mathbf{m}=\boldsymbol{\nabla}_{\mathbf{m}}\left\langle\tilde{\chi}\left(\mathbf{m}\right)\cdot\boldsymbol{\delta}\mathbf{m}\right\rangle=\left\langle\boldsymbol{\nabla}_{\mathbf{m}}\tilde{\chi}\left(\mathbf{m}\right)\cdot\boldsymbol{\delta}\mathbf{m}\right\rangle, \tag{2.6}$$

as order of differentiation and integration can be interchanged. Notice that misfit depends on the model $\mathbf{m}$ only indirectly via the regular wavefield $\mathbf{u}\left(\mathbf{m};\mathbf{r},t\right)$. That means the differential of misfit must be calculated by the chain rule

$$\begin{aligned}\delta\chi\left[\mathbf{u}\left(\mathbf{m}\right)\right] &= \boldsymbol{\nabla}_{\mathbf{m}}\chi\left(\mathbf{m}\right)\cdot\boldsymbol{\delta}\mathbf{m}=\boldsymbol{\nabla}_{\mathbf{u}}\chi\left(\mathbf{u}\right)\cdot\boldsymbol{\nabla}_{\mathbf{m}}\mathbf{u}\left(\mathbf{m}\right)\cdot\boldsymbol{\delta}\mathbf{m}=\\ &= \boldsymbol{\nabla}_{\mathbf{u}}\chi\left(\mathbf{u}\right)\cdot\boldsymbol{\delta}\mathbf{u}=\left\langle\boldsymbol{\nabla}_{\mathbf{u}}\tilde{\chi}\left(\mathbf{u}\right)\cdot\boldsymbol{\delta}\mathbf{u}\right\rangle,\end{aligned} \tag{2.7}$$

where $\boldsymbol{\delta}\mathbf{u}=\boldsymbol{\nabla}_{\mathbf{m}}\mathbf{u}\cdot\boldsymbol{\delta}\mathbf{m}$ is a small change of the regular wavefield corresponding to a small change of the model $\boldsymbol{\delta}\mathbf{m}$.

It is obvious that the most problematic term is $\boldsymbol{\delta}\mathbf{u}$, as it requires to compute a gradient of displacement field with respect to model parameters. Therefore, we try to express this term using something which is not so computationally demanding.

Let $\boldsymbol{\nabla}_{\mathbf{u}}\tilde{\chi}^{\dagger}$ be an adjoint operator to $\boldsymbol{\nabla}_{\mathbf{u}}\tilde{\chi}$ satisfying relation

$$\left\langle\boldsymbol{\delta}\mathbf{u}\cdot\boldsymbol{\nabla}_{\mathbf{u}}\tilde{\chi}^{\dagger}\right\rangle=\left\langle\boldsymbol{\nabla}_{\mathbf{u}}\tilde{\chi}\cdot\boldsymbol{\delta}\mathbf{u}\right\rangle \tag{2.8}$$

for any $\boldsymbol{\delta}\mathbf{u}$. Then eq. (2.7) can be rewritten as

$$\boldsymbol{\nabla}_{\mathbf{m}}\chi\left(\mathbf{m}\right)\cdot\boldsymbol{\delta}\mathbf{m}=\left\langle\boldsymbol{\delta}\mathbf{u}\cdot\boldsymbol{\nabla}_{\mathbf{u}}\tilde{\chi}^{\dagger}\right\rangle. \tag{2.9}$$

Next we differentiate eq. (2.2) with respect to $\mathbf{m}$, multiply it with an arbitrary function $\mathbf{u}^{\dagger}\left(\mathbf{r},t\right)$ and integrate it over time and

space. We obtain

$$\left\langle \mathbf{u}^{\dagger} \cdot \boldsymbol{\nabla}_{\mathbf{m}} \boldsymbol{\mathcal{L}} \left(\mathbf{m}; \mathbf{u} \right) \cdot \boldsymbol{\delta} \mathbf{m} \right\rangle + \left\langle \mathbf{u}^{\dagger} \cdot \boldsymbol{\nabla}_{\mathbf{u}} \boldsymbol{\mathcal{L}} \left(\mathbf{m}; \mathbf{u} \right) \cdot \boldsymbol{\delta} \mathbf{u} \right\rangle = 0. \tag{2.10}$$

Let $\boldsymbol{\nabla}_{\mathbf{u}} \boldsymbol{\mathcal{L}}^{\dagger} \left(\mathbf{m}; \mathbf{u} \right)$ be an adjoint operator to $\boldsymbol{\nabla}_{\mathbf{u}} \boldsymbol{\mathcal{L}} \left(\mathbf{m}; \mathbf{u} \right)$ satisfying

$$\left\langle \boldsymbol{\delta} \mathbf{u} \cdot \boldsymbol{\nabla}_{\mathbf{u}} \boldsymbol{\mathcal{L}}^{\dagger} \cdot \mathbf{u}^{\dagger} \right\rangle = \left\langle \mathbf{u}^{\dagger} \cdot \boldsymbol{\nabla}_{\mathbf{u}} \boldsymbol{\mathcal{L}} \cdot \boldsymbol{\delta} \mathbf{u} \right\rangle. \tag{2.11}$$

Then summing of eq. (2.9) and eq. (2.10) using eq. (2.11) gives

$$\begin{aligned} \boldsymbol{\nabla}_{\mathbf{m}} \chi \cdot \boldsymbol{\delta} \mathbf{m} &= \left\langle \boldsymbol{\delta} \mathbf{u} \cdot \boldsymbol{\nabla}_{\mathbf{u}} \tilde{\chi}^{\dagger} \right\rangle + \left\langle \mathbf{u}^{\dagger} \cdot \boldsymbol{\nabla}_{\mathbf{m}} \boldsymbol{\mathcal{L}} \cdot \boldsymbol{\delta} \mathbf{m} \right\rangle + \left\langle \boldsymbol{\delta} \mathbf{u} \cdot \boldsymbol{\nabla}_{\mathbf{u}} \boldsymbol{\mathcal{L}}^{\dagger} \cdot \mathbf{u}^{\dagger} \right\rangle = \\ &= \left\langle \boldsymbol{\delta} \mathbf{u} \cdot \left(\boldsymbol{\nabla}_{\mathbf{u}} \tilde{\chi}^{\dagger} + \boldsymbol{\nabla}_{\mathbf{u}} \boldsymbol{\mathcal{L}}^{\dagger} \cdot \mathbf{u}^{\dagger} \right) \right\rangle + \left\langle \mathbf{u}^{\dagger} \cdot \boldsymbol{\nabla}_{\mathbf{m}} \boldsymbol{\mathcal{L}} \cdot \boldsymbol{\delta} \mathbf{m} \right\rangle. \end{aligned} \tag{2.12}$$

If

$$\boldsymbol{\nabla}_{\mathbf{u}} \boldsymbol{\mathcal{L}}^{\dagger} \left(\mathbf{m}; \mathbf{u} \right) \cdot \mathbf{u}^{\dagger} \left(\mathbf{m} \right) = - \boldsymbol{\nabla}_{\mathbf{u}} \tilde{\chi}^{\dagger} \left(\mathbf{m} \right), \tag{2.13}$$

$\boldsymbol{\delta} \mathbf{u}$ is eliminated from eq. (2.12) and we obtain a formula for the differential of misfit independent of $\boldsymbol{\delta} \mathbf{u}$

$$\delta \chi \left(\mathbf{m} \right) = \boldsymbol{\nabla}_{\mathbf{m}} \chi \left(\mathbf{m} \right) \cdot \boldsymbol{\delta} \mathbf{m} = \left\langle \mathbf{u}^{\dagger} \left(\mathbf{m} \right) \cdot \boldsymbol{\nabla}_{\mathbf{m}} \boldsymbol{\mathcal{L}} \left(\mathbf{m}; \mathbf{u} \right) \cdot \boldsymbol{\delta} \mathbf{m} \right\rangle. \tag{2.14}$$

It allows us to compute a change of misfit associated with a change of model $\boldsymbol{\delta} \mathbf{m}$ without explicit knowledge of $\boldsymbol{\delta} \mathbf{u}$. eq. (2.13) is called an *adjoint equation* and the right-hand-side term

$$\mathbf{f}^{\dagger} \left(\mathbf{m} \right) = - \boldsymbol{\nabla}_{\mathbf{u}} \tilde{\chi}^{\dagger} \left(\mathbf{m} \right) \tag{2.15}$$

is a so-called *adjoint source*. In fact, it is a sum of all adjoint sources situated at the places of receivers. If $\boldsymbol{\mathcal{L}} \left(\mathbf{m}; \cdot \right)$ is a linear operator, which is the case, it is applied on a function $\mathbf{u}$ like

$$\boldsymbol{\mathcal{L}} \left(\mathbf{m}; \mathbf{u} \right) = \boldsymbol{\mathcal{L}} \left(\mathbf{m}; \cdot \right) \mathbf{u}. \tag{2.16}$$

The left-hand-side term of eq. (2.13) can be modified using definition of directional derivative

$$\begin{aligned}
&\boldsymbol{\nabla}_{\mathbf{u}}\boldsymbol{\mathcal{L}}^{\dagger}(\mathbf{m};\mathbf{u})\cdot\mathbf{u}^{\dagger}=\boldsymbol{\nabla}_{\mathbf{u}}\boldsymbol{\mathcal{L}}^{\dagger}(\mathbf{m};\mathbf{u})\cdot\frac{\mathbf{u}^{\dagger}}{\|\mathbf{u}^{\dagger}\|}\cdot\|\mathbf{u}^{\dagger}\|=\\
&=\lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\boldsymbol{\mathcal{L}}^{\dagger}\left(\mathbf{m};\mathbf{u}+\varepsilon\frac{\mathbf{u}^{\dagger}}{\|\mathbf{u}^{\dagger}\|}\right)-\boldsymbol{\mathcal{L}}^{\dagger}(\mathbf{m};\mathbf{u})\right]\cdot\|\mathbf{u}^{\dagger}\|=\\
&=\lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\boldsymbol{\mathcal{L}}^{\dagger}(\mathbf{m};\mathbf{u})+\boldsymbol{\mathcal{L}}^{\dagger}\left(\mathbf{m};\varepsilon\frac{\mathbf{u}^{\dagger}}{\|\mathbf{u}^{\dagger}\|}\right)-\boldsymbol{\mathcal{L}}^{\dagger}(\mathbf{m};\mathbf{u})\right]\cdot\|\mathbf{u}^{\dagger}\|=\\
&=\lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\boldsymbol{\mathcal{L}}^{\dagger}\left(\mathbf{m};\varepsilon\frac{\mathbf{u}^{\dagger}}{\|\mathbf{u}^{\dagger}\|}\right)\cdot\|\mathbf{u}^{\dagger}\|=\\
&=\lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\boldsymbol{\mathcal{L}}^{\dagger}(\mathbf{m};\cdot)\cdot\varepsilon\frac{\mathbf{u}^{\dagger}}{\|\mathbf{u}^{\dagger}\|}\cdot\|\mathbf{u}^{\dagger}\|=\\
&=\boldsymbol{\mathcal{L}}^{\dagger}(\mathbf{m};\cdot)\cdot\mathbf{u}^{\dagger}=\boldsymbol{\mathcal{L}}^{\dagger}\left(\mathbf{m};\mathbf{u}^{\dagger}\right).
\end{aligned}\tag{2.17}$$

Then the adjoint equation can be written in the form

$$\boldsymbol{\mathcal{L}}^{\dagger}\left(\mathbf{m};\mathbf{u}^{\dagger}\right)=\mathbf{f}^{\dagger}(\mathbf{m}),\tag{2.18}$$

which is formally the same equation as eq. (2.2) for calculation of the regular wavefield.

To conclude, we have shown that the adjoint wavefield $\mathbf{u}^{\dagger}(\mathbf{r},t)$ can be calculated using the adjoint equation eq. (2.18). The operator of elastic wave equation is self-adjoint

$$\boldsymbol{\mathcal{L}}^{\dagger}=\boldsymbol{\mathcal{L}},\tag{2.19}$$

provided some conditions are met, which means the adjoint wavefield $\mathbf{u}^{\dagger}(\mathbf{r},t)$ can be computed by same methods as the regular wavefield $\mathbf{u}(\mathbf{r},t)$.

The adjoint source is given by eq. (2.15). For L2-misfit defined by eq. (1.12), it is calculated simply as residuals between the regular wavefield $\mathbf{u}(\mathbf{r},t)$ and the observed wavefield $\mathbf{u_0}(\mathbf{r},t)$ recorded on seismograms

$$\mathbf{f}^{\dagger}(\mathbf{r},t)=\sum_{i=1}^{n}\left[\mathbf{u_0}(\mathbf{r},t)-\mathbf{u}(\mathbf{m};\mathbf{r},t)\right]\delta(\mathbf{r}-\mathbf{r_i}).\tag{2.20}$$

The adjoint sources are located at the places of receivers $\mathbf{r_i}$.

In the process of derivation of the adjoint operator $\boldsymbol{\mathcal{L}}^\dagger$, some additional constraints are set on the adjoint wavefield $\mathbf{u}^\dagger(\mathbf{r},t)$ (see Fichtner, 2011). Those are zero terminal conditions

$$\mathbf{u}^\dagger(\mathbf{r},t)\big|_{t=t_1} = 0; \tag{2.21a}$$

$$\dot{\mathbf{u}}^\dagger(\mathbf{r},t)\big|_{t=t_1} = 0, \tag{2.21b}$$

where t_1 is the end of the time window over which time integrals are calculated, and a free-surface boundary condition

$$\mathbf{n}\cdot\boldsymbol{\sigma}^\dagger(\mathbf{r},t)\big|_{\mathbf{r}\in\partial V} = 0. \tag{2.22}$$

Hence, the adjoint wavefield is fully determined.

Calculation of the adjoint wavefield requires one modification comparing to calculation of the regular wavefield. In order to fulfil zero terminal conditions, we need to start a simulation at time t_1 and propagate waves backward in time. As the source-time function is given by residuals of the regular wavefield and the observed wavefield, it can be viewed as a back propagation of the residuals in time.

The reverse time propagation of the adjoint wavefield yields a serious complication. To calculate the differential of misfit according to eq. (1.12), we need know both the regular and the adjoint wavefield at the same time level. That means we must store entire history of the regular wavefield, which is usually beyond the bounds of possibility. This problem can be avoided for a price of increased computational demands. For instance, if the continuum is perfectly elastic, we can replace storing of the regular wavefield by a simulation of its reverse time propagation along with the adjoint wavefield. Other methods applicable also in cases when energy is not conserved are checkpointing or data compression.

2.3 Misfit sensitivity kernels

Recall eq. (2.14). We perform the integration over time, so that we obtain

$$\begin{aligned}
\delta\chi(\mathbf{m}) &= \boldsymbol{\nabla}_{\mathbf{m}}\chi(\mathbf{m}) \cdot \boldsymbol{\delta}\mathbf{m} = \\
&= \int_T \int_V \mathbf{u}^\dagger(\mathbf{m};\mathbf{r},t) \cdot \boldsymbol{\nabla}_{\mathbf{m}} \boldsymbol{\mathcal{L}}\left[\mathbf{m};\mathbf{u}(\mathbf{m};\mathbf{r},t)\right] \cdot \boldsymbol{\delta}\mathbf{m}(\mathbf{r}) \,\mathrm{d}^3 r \,\mathrm{d}t = \\
&= \int_V \left\{ \int_T \mathbf{u}^\dagger(\mathbf{m};\mathbf{r},t) \cdot \boldsymbol{\nabla}_{\mathbf{m}} \boldsymbol{\mathcal{L}}\left[\mathbf{m};\mathbf{u}(\mathbf{m};\mathbf{r},t)\right] \mathrm{d}t \right\} \cdot \boldsymbol{\delta}\mathbf{m}(\mathbf{r}) \,\mathrm{d}^3 r = \\
&= \int_V \mathbf{K}_{\mathbf{m}}(\mathbf{r}) \cdot \boldsymbol{\delta}\mathbf{m}(\mathbf{r}) \,\mathrm{d}^3 r, \qquad (2.23)
\end{aligned}$$

where

$$\mathbf{K}_{\mathbf{m}}(\mathbf{r}) = \int_T \mathbf{u}^\dagger(\mathbf{m};\mathbf{r},t) \cdot \boldsymbol{\nabla}_{\mathbf{m}} \boldsymbol{\mathcal{L}}\left[\mathbf{m};\mathbf{u}(\mathbf{m};\mathbf{r},t)\right] \mathrm{d}t \qquad (2.24)$$

is a so-called misfit sensitivity kernel. It can be viewed as a volumetric density of misfit gradient with respect to the model parameters. It expresses how much the misfit functional $\chi(\mathbf{m})$ is affected by the change of model parameters $\boldsymbol{\delta}\mathbf{m}$ at position $\mathbf{r}$.

Kernels can be calculated for each level of misfit defined on page 24. The associated kernels are:

- a *source-receiver kernel* – the kernel between a source and a receiver which is calculated using the regular wavefield and the adjoint wavefield excited by one adjoint source;

- an *event kernel* – the kernel between a source and all receivers which is calculated as a sum of all source-receivers kernels for single event, or alternatively, thanks to linearity of the wave equation, it can be calculated using the regular wavefield and the adjoint wavefield excited by all adjoint sources;

- an *aggregate (total) kernel* – a sum of the event kernels for all sources used in inversion.

It is recommended to use the aggregate kernels for updating the model in order to ensure convergence. If we chose a different type of kernels, we might have to face problems as discussed for different levels of misfit.

The aggregate kernels can be calculated effectively from superposition of the regular wavefields of all events and the adjoint wavefield excited by all adjoint sources of all events. This is the idea behind a so-called *source-stacking technique* (e.g. Capdeville et al., 2005). Such an approach requires a calculation of only one regular and one adjoint wavefield which makes it really effective. However, it leads to the appearance of additional artefacts due to the interaction of non-related wavefields.

Exact expressions for kernel components depend on model parametrisation. One such parametrisation is

$$\mathbf{m}(\mathbf{r}) = [\rho(\mathbf{r}), \lambda(\mathbf{r}), \mu(\mathbf{r})], \tag{2.25}$$

in which kernel components are given by

$$K_\rho(\mathbf{r}) = -\int_T \dot{\mathbf{u}}^\dagger(\mathbf{r},t) \cdot \dot{\mathbf{u}}(\mathbf{r},t)\, \mathrm{d}t; \tag{2.26a}$$

$$\begin{aligned} K_\lambda(\mathbf{r}) &= \int_T \left[\boldsymbol{\nabla}\cdot\mathbf{u}^\dagger(\mathbf{r},t)\right] \cdot \left[\boldsymbol{\nabla}\cdot\mathbf{u}(\mathbf{r},t)\right] \mathrm{d}t = \\ &= \int_T \mathrm{Tr}\,\boldsymbol{\varepsilon}^\dagger(\mathbf{r},t) \cdot \mathrm{Tr}\,\boldsymbol{\varepsilon}(\mathbf{r},t)\, \mathrm{d}t; \end{aligned} \tag{2.26b}$$

$$\begin{aligned} K_\mu(\mathbf{r}) &= \int_T \boldsymbol{\nabla}\mathbf{u}^\dagger(\mathbf{r},t) : \boldsymbol{\nabla}\mathbf{u}(\mathbf{r},t) + \boldsymbol{\nabla}\mathbf{u}^\dagger(\mathbf{r},t) : \boldsymbol{\nabla}\mathbf{u}^{\mathrm{T}}(\mathbf{r},t)\, \mathrm{d}t = \\ &= 2\int_T \boldsymbol{\varepsilon}^\dagger(\mathbf{r},t) : \boldsymbol{\varepsilon}(\mathbf{r},t)\, \mathrm{d}t. \end{aligned} \tag{2.26c}$$

2.3.1 Kernel preconditioning

Kernels, as calculated according to eq. (2.26), cannot be used directly to update model. Firstly, they have to be modified in order to get rid of unpleasant features like singularities, high frequency oscillations, additional artefacts, or to make corrections for uneven source and receiver coverage, radiation patterns or many others. A specific set of necessary modifications depends on the particular problem and used method.

Everyone agrees that preconditioning is an important part of inversion procedure, thus everyone uses it to some extent. However only a few describe what preconditioning they use and what techniques they apply. Studies with detailed description of applied preconditioning like Zhu; Bozdağ; Tromp, 2015 or Bozdağ et al., 2016 are rather exception.

Standard preconditioning includes clipping and smoothing. Clipping is a technique for removal of singularities. The singularities appear around sources – both regular and adjoint – because energy is focussed in their vicinity which results to high amplitudes of wavefields.

Clipping is based on a truncation of kernels. However, the proper maximum value must be guessed. As a consequence of clipping, areas of constant kernel value appear, which can introduce sharp edges into model. Therefore, smoothing must be applied afterwards. Another purpose of smoothing is suppressing high-frequency content and removing additional artefacts.

There is a dissonance among authors concerning the terminology. Kubina, 2017 uses term *preconditioning* for all modifications performed with kernel between its computation and applying for model update. Other authors are more specific.

Zhu; Bozdağ; Tromp, 2015 use term *pre-conditioning* only for spatially-dependent normalization by pseudo-Hessian, the aim of which is to remove singularities. The pseudo-Hessian is calcu-

lated using regular and adjoint accelerations and it has high values just around sources, thus the kernel is lowered the most therein. It also serves as a correction to an uneven source and receivers coverage. Consequently, they apply smoothing by Gauss function, however they classify it rather as regularisation technique than pre-conditioning.

Bozdağ et al., 2016 are even more specific. They distinguish *pre-processing* and *post-processing*. The former includes operations preceding the kernel computation, which we are not interested in, the later names operations with calculated kernels – summation of event kernels, smoothing, pre-conditioning (normalisation by pseudo-Hessian), optimisation (use of conjugate-gradient method) and determining of step length.

So far, we adopted the terminology used by Kubina, 2017, though it is not entirely correct. The term preconditioning is associated with a set of modifications whose purpose is to ensure stability of the solution and fast convergence. However, he uses it also for application of mask which is rather regularisation than preconditioning, thus we prefer to stick with the term *kernel preprocessing*. Contrary to Bozdağ et al., 2016, we perceive it as a set of modifications of kernels which are to be performed before the kernel can be used for updating the model. Therefore from here on, through the Part II, we will use the term kernel preprocessing.

2.4 Model update

Once we have calculated the kernel, we can proceed to updating of the model. We do it iteratively. Let's assume that we operate with a model $\mathbf{m_i}$ on i-th iteration. A subsequent model $\mathbf{m_{i+1}}$ will be obtained based on linear approximation

$$\mathbf{m_{i+1}} = \mathbf{m_i} + \gamma_i \mathbf{h_i}. \tag{2.27}$$

For that purpose we need to determine a search direction $\mathbf{h_i}$ and a step length γ_i.

The most intuitive choice of the search direction is to pick out the steepest descent direction given by the negative misfit gradient. Recall that kernel was defined as a density of the misfit gradient, thus it has its direction. Therefore, we can opt

$$\mathbf{h_i} = -\mathbf{K_i}(\mathbf{r}). \tag{2.28}$$

Note, that the raw kernel had the direction of the steepest ascent. The direction of the kernel after preprocessing is slightly deviated from that direction. However, this deviation is in our favour.

Determination of the step length is more complicated. If we chose a sufficiently short step length, we would obtain a new model associated with lower misfit, so that a convergence to a local minimum would be guaranteed. However, it might require too many iterations to reach it and the convergence would be too slow. Sometimes we can use a far longer step. Therefore, it is advantageous to determine the step length for each iteration separately. It is done by a few trial computations. We choose a few different step lengths and compute a model and a corresponding misfit for each of them. Then we construct a function $\chi(\gamma_i)$ by interpolating misfits obtained for trial computations. Finally, we determine optimal step length as

$$\gamma_i^{\mathrm{opt}} = \arg\min_{\gamma_i} \chi(\gamma_i). \tag{2.29}$$

The new model is then calculated as

$$\mathbf{m_{i+1}}(\mathbf{r}) = \mathbf{m_i}(\mathbf{r}) - \gamma_i^{\mathrm{opt}} \mathbf{K_i}(\mathbf{r}). \tag{2.30}$$

This approach is used by Kubina, 2017. Some authors rather use a conjugate gradient method for updating of the model, which determines the search direction as a linear combination of a direction of the misfit gradient and the search direction from the previous iteration. The conjugate gradient method is suitable when the problem is not strongly nonlinear, which is not often the case in local surface sedimentary structures.

2.4.1 Multiscale approach

The algorithm described in this section converges always to the closest local minimum. However, we are interested in finding of the global minimum. It happens only if the initial model is sufficiently close to the true model. However, it is rarely the case. We usually have little information about the true model, which means that the convergence to the global minimum is not guaranteed.

A multiscale approach to the minimisation process can overcome this problem. It is based on the idea, that long waves are not sensitive to short-length structures. Therefore, if we filter data by a low-pass filter and use these filtered data, we obtain a relatively simple misfit functional $\chi(\mathbf{m})$.

It is like that because long waves "see" only rough structures, which leads to a simple wavefields and thus to the simple misfit. In other words, the initial and the true model are very similar on this scale, therefore the misfit is small and simple. Such a simple misfit has a wider region, from which the algorithm converges to the global minimum, therefore the initial model does not have to be very similar to the true model.

Another advantage is that thanks to the misfit simplicity, we do not have to do many trial computations when determining the optimal step length for model update. Even a quadratic approximation to the $\chi(\gamma_i)$ may be sufficient which means that only two

trial computation are necessary, as one value we already have (for $\gamma_i = 0$).

The first iteration on the highest scale finishes with a rough resulting model. This model is used as the initial model for the second iteration which is performed for wider frequency range. Although, the misfit is now more complex, the initial model is close enough for the algorithm to converge to the global minimum.

A gradual widening of the frequency range covering also lower wavelength leads to the well-determined resulting model.

Part II

Kernel preprocessing

3 Smoothing

3.1 Overview of commonly used smoothing techniques of 3D scalar fields

3.1.1 Smoothing as convolution with smoothing function

The most straightforward approach to smoothing is to calculate value of $\mathbf{K}^{\mathrm{smoothed}}(\mathbf{r})$ as a weighted average of values $\mathbf{K}(\boldsymbol{\rho})$. The values $\mathbf{K}(\boldsymbol{\rho})$ can be taken either from the entire volume of continuum or from only a small space window centred on $\mathbf{r}$.

The weights are given by a smoothing function $w(\mathbf{r}, \boldsymbol{\rho})$. The smoothing function is sometimes called also a window function or a kernel. However, we will not use the term "kernel" for the smoothing function to avoid confusion with misfit sensitivity kernels.

The weights, in fact, depend only on a relative position of $\mathbf{r}$ and $\boldsymbol{\rho}$, thus

$$w(\mathbf{r}, \boldsymbol{\rho}) = w(\mathbf{r} - \boldsymbol{\rho}). \tag{3.1}$$

Therefore, the corresponding weighted average can be calculated as

$$\mathbf{K}^{\mathrm{smoothed}}(\mathbf{r}) = \frac{1}{W(\mathbf{r})} \int_V w(\mathbf{r} - \boldsymbol{\rho}) \cdot \mathbf{K}(\boldsymbol{\xi}) \, \mathrm{d}^3\rho, \tag{3.2}$$

where

$$W(\mathbf{r}) = \int_V w(\mathbf{r} - \boldsymbol{\rho}) \, \mathrm{d}^3\rho. \tag{3.3}$$

The eq. (3.2) is a convolution integral, thus we can write it as

$$\mathbf{K}^{\mathrm{smoothed}}(\mathbf{r}) = \frac{1}{W(\mathbf{r})} \cdot w(\mathbf{r}) * \mathbf{K}(\mathbf{r}). \tag{3.4}$$

A common choice of the smoothing function is a Gauss function. An example can be found for instance in Zhu; Bozdağ; Tromp, 2015, though they do not specify what is an integration domain. Although many authors do not state explicitly what smoothing

technique they use, we assume this is the choice of majority of studies.

3.1.1.1 Time complexity We will now estimate time complexity of a numerical computation of the convolution integral (3.2). In a discretised form, it translates to a triple sum

$$\mathbf{K}^{\mathrm{smoothed}}_{i\,j\,k} = \frac{1}{W_{i\,j\,k}} \sum_{\iota=0}^{N_x-1} \sum_{\gamma=0}^{N_y-1} \sum_{\kappa=0}^{N_z-1} w^{\iota\,\gamma\,\kappa}_{i\,j\,k} \mathbf{K}_{\iota\,\gamma\,\kappa}. \tag{3.5}$$

Let N be a geometric mean of numbers of grid points in respective directions. The sum (3.5) has to be calculated for each combination of (i, j, k), thus it is N^3 calculations of the triple sum. The sum itself requires addition of N^3 summands. Therefore, the overall time complexity of calculation of the sum (3.5) is $\mathcal{O}\left(N^6\right)$.

Fortunately, some reductions of time complexity are possible. First of all, we can realise that the weights of distant points are negligible, thus we can leave them out. That effectively means that we perform summation over smaller subdomain of size $n_x \times n_y \times n_z$. If we denote the geometric mean of the dimensions of this spatial widow as n, the number of summands to be added together when calculating the sum (3.5) is n^3, thus the overall time complexity is reduced to $n^3\mathcal{O}\left(N^3\right)$, where n is a constant independent of the dimensions of the entire integration domain. However, the integration domains for local surface structures are usually not very large, thus the spatial window covers a significant portion of the domain and the reduction of the computational time is only slight.

Another cut of computational time can be reached by parallelisation of the computation. We divide the computational domain into several smaller subdomains and calculate the sum separately in each of them. At the end, the partial sums are shared among

subdomains and the final sum is calculated. A reasonable number of subdomains for local surface structures is about $4^3 = 64$, thus the computational time is further reduced to $\frac{n^3}{64}\mathcal{O}\left(N^3\right)$.

For one special choice of the smoothing function, it is possible to use an efficient algorithm for a computation of the sum (3.5). It was developed by Kubina, 2017. We will discuss it in the section 3.2.

3.1.2 Smoothing as low-pass filtering

The second approach to the calculation of the convolution integral (3.2) is to perform the calculation in a frequency domain. First of all, we need express the kernel and the smoothing function in frequency domain. It is done by performing of Fourier transform

$$\hat{\mathbf{K}}\left(\mathbf{k}\right) = \mathscr{F}\left[\mathbf{K}\left(\mathbf{r}\right)\right] = \int_V \mathbf{K}\left(\mathbf{r}\right) \mathrm{e}^{-\mathrm{i}\mathbf{k}\cdot\mathbf{r}}\, \mathrm{d}^3 r; \tag{3.6a}$$

$$\hat{w}\left(\mathbf{k}\right) = \mathscr{F}\left[w\left(\mathbf{r}\right)\right] = \int_V w\left(\mathbf{r}\right) \mathrm{e}^{-\mathrm{i}\mathbf{k}\cdot\mathbf{r}}\, \mathrm{d}^3 r. \tag{3.6b}$$

Next, we make use of a property of Fourier transform that a Fourier transform of a convolution is a product of Fourier transforms of convolved functions

$$\mathscr{F}\left[\mathbf{K}\left(\mathbf{r}\right) * w\left(\mathbf{r}\right)\right] = \mathscr{F}\left[\mathbf{K}\left(\mathbf{r}\right)\right] \cdot \mathscr{F}\left[w\left(\mathbf{r}\right)\right]. \tag{3.7}$$

Using inverse Fourier transform, we can express the sought convolution as

$$\mathbf{K}\left(\mathbf{r}\right) * w\left(\mathbf{r}\right) = \mathscr{F}^{-1}\left[\hat{\mathbf{K}}\left(\mathbf{k}\right) \cdot \hat{w}\left(\mathbf{k}\right)\right]. \tag{3.8}$$

Notice that we can look at it as on a filtering. In that case, the function $\hat{w}\left(\mathbf{k}\right)$ serves as a transfer function of the filter. The transfer function completely determines properties of the filter. It describes which frequencies are suppressed and which are al-

lowed. In case of smoothing, only low frequencies are allowed, thus $\hat{w}(\mathbf{k})$ effectively equals zero for high wave numbers $\|\mathbf{k}\|$.

3.1.2.1 Time complexity Smoothing in frequency domain consists of three steps:

1. computation of Fourier transform of the kernel;
2. application of the filter;
3. computation of inverse Fourier transform of the filtered kernel.

The first step requires calculation of the integrals (3.6). In fact, only the integral (3.6a) needs to be calculated. If a standard smoothing function is used, a corresponding transfer function is known, and even if it were not, the integral would have to be calculated only once.

We have already shown that evaluation of a general integral in 3D is of complexity $\mathcal{O}(N^6)$. However, eq. (3.6a) is a very special integral – it is an integral of Fourier transform, thus it can be calculated by the algorithm of Fast Fourier Transform (FFT) which has time complexity $3\mathcal{O}(N^3 \log N)$ in a three-dimensional case. The best choice is to use the algorithm of "the Fastest Fourier Transform in the West" (FFTW3) developed by Frigo et al., 2005, which is not limited by any conditions on numbers of grid points in respective directions, contrary to standard implementations of FFT.

The second step is the application of the filter which is, in fact, only N^3 multiplications of two numbers.

The last step is the computation of inverse Fourier transform which can be performed again by FFTW, thus the time complexity of this step is $3\mathcal{O}(N^3 \log N)$ too.

If we add all contributions together, the overall time complexity will be $\mathcal{O}(N^3(6 \log N + 1)) \sim 6\mathcal{O}(N^3 \log N)$. For small compu-

tational domains, it is more effective than a calculation of the convolution integral in space domain. However, the FFTW3 algorithm must be used.

It is questionable whether it is worth implementing smoothing algorithm in frequency domain. We were unable to find out whether there is someone who uses it, as no study provides such details.

3.1.3 Bessel smoothing filters

A novel idea to the smoothing is presented independently by Wellington, 2016 and Trinh et al., 2017. Although they used different approaches, they obtained same results. Instead of direct calculation of eq. (3.4), they rather solve a linear system

$$w^{-1}(\mathbf{r}) * \mathbf{K}^{\text{smoothed}}(\mathbf{r}) = w^{-1}(\mathbf{r}) * [w(\mathbf{r}) * \mathbf{K}(\mathbf{r})] = \mathbf{K}(\mathbf{r}), \quad (3.9)$$

where $w^{-1}(\mathbf{r})$ is an inverse operator to $w(\mathbf{r})$ satisfying

$$w^{-1}(\mathbf{r}) * w(\mathbf{r}) = \delta(\mathbf{r}). \quad (3.10)$$

A matrix associated with $w^{-1}(\mathbf{r})$ is sparse, thus a significant reduction of computational time is achieved.

3.1.3.1 Trinh's approach

Trinh et al., 2017 start with Bessel filters

$$B_{2\text{D}} = \frac{1}{2\pi L_x L_y} K_0(r_{2\text{D}}); \quad (3.11a)$$

$$B_{3\text{D}} = \frac{1}{\sqrt{2\pi}^3 L_x L_y L_z} r_{3\text{D}}^{-1/2} K_{1/2}(r_{3\text{D}}), \quad (3.11b)$$

where $r_{2\text{D}} = \sqrt{\frac{x^2}{L_x^2} + \frac{y^2}{L_y^2}}$ and $r_{3\text{D}} = \sqrt{\frac{x^2}{L_x^2} + \frac{y^2}{L_y^2} + \frac{z^2}{L_z^2}}$. K_ν is a modified Bessel function of the second kind. The Bessel filters are solu-

tions of partial differential equations

$$\left[1 - \left(L_x^2 \frac{\partial^2}{\partial x^2} + L_y^2 \frac{\partial^2}{\partial y^2}\right)\right] B_{2\mathrm{D}}(\mathbf{r}) = \delta(\mathbf{r}); \quad (3.12a)$$

$$\left[1 - \left(L_x^2 \frac{\partial^2}{\partial x^2} + L_y^2 \frac{\partial^2}{\partial y^2} + L_z^2 \frac{\partial^2}{\partial z^2}\right)\right] B_{2\mathrm{D}}(\mathbf{r}) = \delta(\mathbf{r}). \quad (3.12b)$$

The inverse Bessel filters must satisfy eq. (3.10), thus

$$B_{2\mathrm{D}}^{-1}(\mathbf{r}) = \delta(\mathbf{r}) - \left[L_x^2 \delta^{(2)}(x)\,\delta(y) + L_y^2 \delta(x)\,\delta^{(2)}(y)\right]; \quad (3.13a)$$

$$B_{3\mathrm{D}}^{-1}(\mathbf{r}) = \delta(\mathbf{r}) - \left[L_x^2 \delta^{(2)}(x)\,\delta(y)\,\delta(z) + L_y^2 \delta(x)\,\delta^{(2)}(y)\,\delta(z) + L_z^2 \delta(x)\,\delta(y)\,\delta^{(2)}(z)\right]. \quad (3.13b)$$

If we apply them on eq. (3.9), we obtain

$$\left[1 - \left(L_x^2 \frac{\partial^2}{\partial x^2} + L_y^2 \frac{\partial^2}{\partial y^2}\right)\right] \mathbf{K}^{\mathrm{smoothed}}(\mathbf{r}) = \mathbf{K}(\mathbf{r}); \quad (3.14a)$$

$$\left[1 - \left(L_x^2 \frac{\partial^2}{\partial x^2} + L_y^2 \frac{\partial^2}{\partial y^2} + L_z^2 \frac{\partial^2}{\partial z^2}\right)\right] \mathbf{K}^{\mathrm{smoothed}}(\mathbf{r}) = \mathbf{K}(\mathbf{r}). \quad (3.14b)$$

A discretised version of these equations is represented by a sparse operator, thus the corresponding system can be solved efficiently.

Let's note that the Bessel filter decays more rapidly than a Laplace filter. In order to mimic a decay of the Laplace filter, we apply the Bessel filter twice, or equivalently, we apply a filter $B(\mathbf{r}) * B(\mathbf{r})$. To do so, we use the same approach – we find an inverse operator $B^{-1}(\mathbf{r}) * B^{-1}(\mathbf{r})$ and solve eq. (3.9). It can be shown that the Laplace filter

$$L(\mathbf{r}) = A\mathrm{e}^{-r} \quad (3.15)$$

is an approximate solution of

$$\left[B^{-1}(\mathbf{r}) * B^{-1}(\mathbf{r})\right] * L(\mathbf{r}) \approx \delta(\mathbf{r}), \quad (3.16)$$

thus

$$L(\mathbf{r}) \approx B(\mathbf{r}) * B(\mathbf{r}). \quad (3.17)$$

3.1.3.2 Wellington's approach Wellington, 2016 took it from the other end. He started with an 1D Laplacian operator

$$L_{1\mathrm{D}}(x,\xi) = \frac{1}{2L_x}\mathrm{e}^{-\frac{|x-\xi|}{L_x}} \tag{3.18}$$

and a well-known corresponding inverse Laplacian operator (see Tarantola, 2005)

$$L_{1\mathrm{D}}^{-1}(x,\xi) = \delta(x-\xi) - L_x^2\delta^{(2)}(x-\xi), \tag{3.19}$$

where $\delta^{(2)}(x-\xi)$ is the second derivative of the Dirac delta function defined as

$$\int \delta^{(n)}(x-\xi)\, f(\xi)\, \mathrm{d}\xi = (-1)^n f^{(n)}(x). \tag{3.20}$$

A finite-difference approximation of the Dirac function and its derivative is

$$\delta(x-\xi) = \begin{cases} {}^1/_h, & x=\xi \\ 0, & x\neq\xi \end{cases}; \tag{3.21}$$

$$\delta^{(2)}(x-\xi) = \begin{cases} -{}^2/_h, & x=\xi \\ {}^1/_h, & x=\xi\pm h \\ 0, & |x-\xi|>h \end{cases}. \tag{3.22}$$

That means we need only three adjacent points, thus the operator is really sparse.

Wellington, 2016 compared results obtained by this technique with results obtained by standard computation of spatial convolution and he found out a great agreement. It encouraged him to generalised this approach to higher dimensions. He guessed a form of the inverse operators in 2D and 3D and verified their correctness.

In 2D, he proposed an operator

$$\begin{aligned}\tilde{L}_{2\mathrm{D}}^{-1}(\mathbf{r},\boldsymbol{\rho}) &= \frac{1}{2}\left[\delta(x-\xi)-L_x^2\delta^{(2)}(x-\xi)\right]+\\ &+\frac{1}{2}\left[\delta(y-\eta)-L_y^2\delta^{(2)}(y-\eta)\right].\end{aligned}\tag{3.23}$$

He named it as a 2D additive inverse Laplacian operator, because he found out that it is not an inverse operator to the Laplacian operator. It decays too rapidly. However, he found out that if he applied the operator twice on the same data, the resulting scalar field would look like it had been smoothed by the true 2D Laplacian operator. The computation requires only five closest points, so the corresponding matrix is sparse.

The 3D case is similar. He proposed an operator

$$\begin{aligned}\tilde{L}_{3\mathrm{D}}^{-1}(\mathbf{r},\boldsymbol{\rho}) &= \frac{1}{3}\left[\delta(x-\xi)-L_x^2\delta^{(2)}(x-\xi)\right]+\\ &+\frac{1}{3}\left[\delta(y-\eta)-L_y^2\delta^{(2)}(y-\eta)\right]+\\ &+\frac{1}{3}\left[\delta(z-\zeta)-L_z^2\delta^{(2)}(z-\zeta)\right],\end{aligned}\tag{3.24}$$

which turned out not to be an inverse operator to the Laplacian operator. However, if it was applied consequently three times, the result was very similar to smoothing by the true 3D Laplacian operator. He called it a 3D additive inverse Laplacian operator. The computation requires only seven closest points comparing to a relatively large 3D subdomain for direct calculation of spatial convolution.

The method allows incorporation of a prior dip. That means if we use anisotropic smoothing, the axes do not have to be aligned with coordinate axes. It costs only a slight increase of computational time, as it is necessary to consider 9 neighbouring points in 2D and 19 points in 3D, compared to 5 and 7 respectively.

Wellington, 2016 did not know the nature of his additive inverse Laplacian operators. Only Trinh et al., 2017 showed later

that they were related to the Bessel filters in a finite-difference representation.

3.1.3.3 Time complexity A presented smoothing technique attributed to the significant time complexity reduction compared to the direct computation of spatial convolution. The complexity of the direct computation is $\mathcal{O}(N^4)$ in 2D and $\mathcal{O}(N^6)$ in 3D. In case the smoothing function can be tensorised (i. e. it is tensorial product of one-dimensional operators), the time complexity is reduced to $\mathcal{O}(N^2)$ in 2D and $\mathcal{O}(N^3)$ in 3D.

The inverse operator approach requires solving of very sparse linear systems. It can be done very efficiently. A computational time depends mainly on number of grid points, thus the time complexity is $\mathcal{O}(N^2)$ in 2D and $\mathcal{O}(N^3)$ in 3D. However, we cannot forget, that we have to repeat calculation twice in 2D and three times in 3D.

Wellington, 2016 compared all three mentioned approaches. Numerical tests revealed that the tensorial approach is the most efficient. However, it is not possible to incorporate a prior dip to that approach, so it should be chosen only if dip is not necessary. Otherwise, the inverse operator approach should be picked out.

3.1.4 Topology-based smoothing

A completely different approach to smoothing is presented by Weinkauf et al., 2010. It utilises a topological analysis and topological simplification of scalar fields.

The first step is an identification of critical points of a scalar field – minima, maxima and saddle points. It is done based on Morse theory (Forman, 1998). Its result is a so-called Morse-Smale complex (Edelsbrunner; Harer et al., 2003) which provides complete information on topology. A practical approach to a com-

putation of the Morse-Smale complex is presented in Gyulassy; Peer-Timo Bremer et al., 2008.

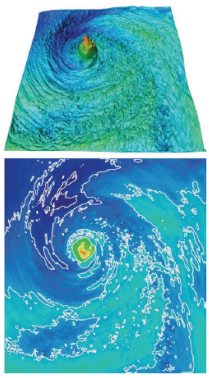
Consequently, the Morse-Smale complex is simplified. A simplification is performed by removing of pairs of critical points (minimum-saddle or maximum-saddle) based on their significance. A measure of significance is a so-called persistence (Edelsbrunner; Letscher et al., 2002). The pairs with little persistence are iteratively removed and only those with the highest persistence are preserved.

Finally, the scalar field is reconstructed from the simplified Morse-Smale complex. As the simplified Morse-Smale complex contains only a few critical points, the resulting field is smooth and simple, whilst the most dominant features are preserved.

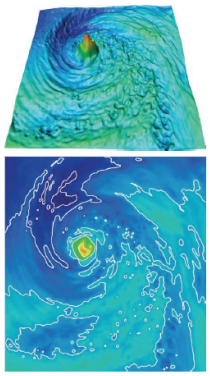
Weinkauf et al., 2010 build on the earlier work of Peer-Time Bremer et al., 2004. They were able to smooth a mid-sized 2D scalar field, however smoothing of 3D fields was yet too computationally demanding to be carried out. Nevertheless, they indicated it would be technically possible.

A generalisation to 3D is based on the work of Gyulassy; Natarajan et al., 2005. For the first time, it was accomplished by Günther et al., 2014. They developed an effective algorithm and parallelised the computation, so that they were able to smooth 3D scalar mid-sized fields in reasonable computational times. Furthermore, they introduced an option to manually pick out critical points which are to be removed, which allowed to selectively remove undesirable features. An example of an application of the method is given on Fig. 3.1.

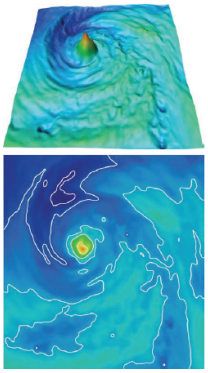
Despite of the recent improvement of the method, it is still too computationally demanding to be applied for tomographic purposes. Furthermore, it is questionable whether it is suitable. Undoubtedly, it is a great method for denoising of data. However, it perfectly preserves main structures. Paradoxically, this biggest



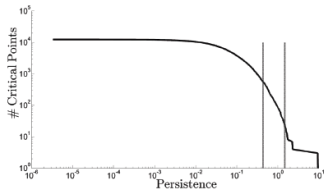
(a) Original data, 14 492 critical points



(b) Persistence level $P = 0.4$, 279 critical points



(c) Persistence level $P = 1.5$, 12 critical points



(d) Number of critical points vs. persistence, vertical lines indicate persistence values of (3.1b) and (3.1c)

Figure 3.1: A slice through 3D data set of temperature in the hurricane Isabel for different persistence thresholds (taken from Günther et al., 2014)

advantage of the method can be a huge problem in our case. We often need blur undesirable structures like artefacts which might have a high persistence, so the method would not remove them. Theoretically, there is an option to select them manually, if we knew where they are located. In any case, we assume it is worth trying, although the application of the method is not feasible with its current computational demands.

3.2 Efficient smoothing algorithm

3.2.1 Theoretical analysis

3.2.1.1 Basic idea behind the algorithm We are interested in calculation of integral (3.2). It can be done efficiently if we use

$$w\left(\mathbf{r}-\boldsymbol{\rho}\right)=\mathrm{e}^{-\left\|\boldsymbol{\Lambda}^{-\mathbf{1}}\cdot\left(\mathbf{r}-\boldsymbol{\rho}\right)\right\|_{1}}=\mathrm{e}^{-\frac{|x-\xi|}{\Lambda_x}}\mathrm{e}^{-\frac{|y-\eta|}{\Lambda_y}}\mathrm{e}^{-\frac{|z-\zeta|}{\Lambda_z}} \tag{3.25}$$

as a smoothing function. A vector

$$\boldsymbol{\Lambda}^{-\mathbf{1}}=\left(\frac{1}{\Lambda_x},\frac{1}{\Lambda_y},\frac{1}{\Lambda_z}\right) \tag{3.26}$$

provides information about smoothing intensities in respective directions. The smoothing intensities are expressed as characteristic smoothing lengths.

Generally, smoothing lengths are different. In such a case we refer to an anisotropic smoothing. Otherwise, the smoothing is called isotropic. The anisotropic smoothing is important, as we usually require less intense smoothing in a vertical direction. The characteristic smoothing lengths are chosen such that a ratio of characteristic lengths is equal to a ratio of model dimensions.

In a discrete case, we are to calculate a triple sum (3.5). It takes a form

$$\mathbf{K}_{i\,j\,k}^{\mathrm{smoothed}}=\frac{1}{W_{i\,j\,k}}\sum_{\iota=0}^{N_x-1}\sum_{\gamma=0}^{N_y-1}\sum_{\kappa=0}^{N_z-1}C_x^{|i-\iota|}C_y^{|j-\gamma|}C_z^{|k-\kappa|}\mathbf{K}_{\iota\,\gamma\,\kappa} \tag{3.27}$$

for our choice of the smoothing function. In order to keep expressions simple, we denote $C=\mathrm{e}^{-\frac{h}{\Lambda}}$, where h is spatial grid step.

If we want to remove an absolute value from the expression, each sum splits into two, one for $\iota\leq i$ and the second for $\iota>i$, and analogously other two sums, thus we need to evaluate $2^3=8$ sums in total. Here, we show how to calculate one such sum.

Consider the octant determined by inequalities $\iota \leq i$, $\gamma \leq j$ and $\kappa \leq k$. A contribution to the total sum from this octant can be calculated as

$$\mathbf{O}_{i\,j\,k} = \sum_{\kappa=0}^{k} C_z^{k-\kappa} \sum_{\gamma=0}^{j} C_y^{j-\gamma} \sum_{\iota=0}^{i} C_x^{i-\iota} \mathbf{K}_{\iota\,\gamma\,\kappa}. \tag{3.28}$$

Let k be an index in a vertical direction. Then terms with same k lie in a single plane $\mathbf{P}^k$. Additionally, the terms with same j within a single plane lie in a row $\mathbf{R}^j$.

Now, consider the octant determined by a point $(i, j, k+1)$. We can express a contribution to the sum from this octant as

$$\begin{aligned}
\mathbf{O}_{i\,j\,k+1} &= \sum_{\kappa=0}^{k+1} C_z^{k+1-\kappa} \sum_{\gamma=0}^{j} C_y^{j-\gamma} \sum_{\iota=0}^{i} C_x^{i-\iota} \mathbf{K}_{\iota\,\gamma\,\kappa} = \\
&= C_z \sum_{\kappa=0}^{k} C_z^{k-\kappa} \sum_{\gamma=0}^{j} C_y^{j-\gamma} \sum_{\iota=0}^{i} C_x^{i-\iota} \mathbf{K}_{\iota\,\gamma\,\kappa} + C_z^0 \sum_{\gamma=0}^{j} C_y^{j-\gamma} \sum_{\iota=0}^{i} C_x^{i-\iota} \mathbf{K}_{\iota\,\gamma\,\kappa} = \\
&= C_z \cdot \mathbf{O}_{i\,j\,k} + \sum_{\gamma=0}^{j} C_y^{j-\gamma} \sum_{\iota=0}^{i} C_x^{i-\iota} \mathbf{K}_{\iota\,\gamma\,\kappa} = \\
&= C_z \cdot \mathbf{O}_{i\,j\,k} + \mathbf{P}_{i\,j}^{k+1},
\end{aligned} \tag{3.29}$$

where $\mathbf{P}_{i\,j}^{k+1}$ is a contribution to the sum from a quadrant $\iota \leq i$ and $\gamma \leq j$ within the plane determined by an index $k+1$. We see that we can use a previously calculated sum $\mathbf{O}_{i\,j\,k}$, so that we do not need to calculate another triple sum, but instead we calculate only a double sum in order to obtain $\mathbf{P}_{i\,j}^{k+1}$.

Let's see whether another reduction of time complexity is possible. Consider the octant determined by a point $(i, j+1, k+1)$.

A contribution from this octant can be calculated as

$$\begin{aligned}
\mathbf{O}_{i\,j+1\,k+1} &= C_z \cdot \mathbf{O}_{i\,j+1\,k} + \mathbf{P}^{k+1}_{i\,j+1} = \\
&= C_z \cdot \mathbf{O}_{i\,j+1\,k} + \sum_{\gamma=0}^{j+1} C_y^{j+1-\gamma} \sum_{\iota=0}^{i} C_x^{i-\iota} \mathbf{K}_{\iota\,\gamma\,\kappa} = \\
&= C_z \cdot \mathbf{O}_{i\,j+1\,k} + C_y \sum_{\gamma=0}^{j} C_y^{j-\gamma} \sum_{\iota=0}^{i} C_x^{i-\iota} \mathbf{K}_{\iota\,\gamma\,\kappa} + C_y^0 \sum_{\iota=0}^{i} C_x^{i-\iota} \mathbf{K}_{\iota\,\gamma\,\kappa} = \\
&= C_z \cdot \mathbf{O}_{i\,j+1\,k} + C_y \cdot \mathbf{P}^{k+1}_{i\,j} + \sum_{\iota=0}^{i} C_x^{i-\iota} \mathbf{K}_{\iota\,\gamma\,\kappa} = \\
&= C_z \cdot \mathbf{O}_{i\,j+1\,k} + C_y \cdot \mathbf{P}^{k+1}_{i\,j} + \mathbf{R}^{j+1}_{i}. \qquad (3.30)
\end{aligned}$$

We have successfully substituted a calculation of a double sum $\mathbf{P}^{k+1}_{i\,j+1}$ with a calculation of a single sum $\mathbf{R}^{j+1}_{i}$ by utilising previously calculated value $\mathbf{P}^{k+1}_{i\,j}$.

Finally, let's take a closer look at a calculation of contribution from the octant determined by a point $(i+1, j+1, k+1)$. We derive that

$$\begin{aligned}
\mathbf{O}_{i+1\,j+1\,k+1} &= C_z \cdot \mathbf{O}_{i+1\,j+1\,k} + C_y \cdot \mathbf{P}^{k+1}_{i+1\,j} + \mathbf{R}^{j+1}_{i+1} = \\
&= C_z \cdot \mathbf{O}_{i+1\,j+1\,k} + C_y \cdot \mathbf{P}^{k+1}_{i+1\,j} + \sum_{\iota=0}^{i+1} C_x^{i+1-\iota} \mathbf{K}_{\iota\,\gamma\,\kappa} = \\
&= C_z \cdot \mathbf{O}_{i+1\,j+1\,k} + C_y \cdot \mathbf{P}^{k+1}_{i+1\,j} + C_x \sum_{\iota=0}^{i} C_x^{i-\iota} \mathbf{K}_{\iota\,\gamma\,\kappa} + C_x^0 \mathbf{K}_{\iota\,\gamma\,\kappa} = \\
&= C_z \cdot \mathbf{O}_{i+1\,j+1\,k} + C_y \cdot \mathbf{P}^{k+1}_{i+1\,j} + C_x \mathbf{R}^{j+1}_{i} + \mathbf{K}_{\iota\,\gamma\,\kappa}. \qquad (3.31)
\end{aligned}$$

Now, it is clear that we can avoid a calculation of a triple sum and substitute it with a summation of three previously calculated, rescaled values with an actual value.

Let's note that an application of this algorithm is limited only for the exponential smoothing function (3.25), as it is the only function that satisfy $f(x+1) = cf(x)$. Additionally, we stress that there is a necessity to keep the order of summation as presented, because the algorithm supposes that previous values are known. The indicated order is as follows:

- We start with the first value in the first row of the first plane.
- Firstly, we calculate entire first row.
- Then we continue with the second row, the third row, etc. within the first plane.
- Subsequently, we carry on with the second plane, the third plane, etc.
- Once we have calculated the contributions for each point of the octant, we continue with another.

3.2.1.2 Numerical implementation We have shown that a calculation of the sum (3.27) decays into a calculation of eight sums (3.28), one for each octant determined by a point (i, j, k). An increased attention must be paid to interfaces between octants in order not to include points from the interfaces twice.

In 2D case, it is not a big problem. We have four quadrants and four interfaces between them, thus we can attach just one interface to each quadrant, as it is depicted on Fig. 3.2. Consequently, we can use exactly same algorithm for calculating the sum within each quadrant. Only difference is that it has to be applied from a different direction in each quadrant.

We will now present the algorithm for calculation of the contribution from one quadrant. We have slightly modified the original version proposed by Kubina, 2017. He attached an interface at the bottom of the quadrant. We instead attach it on the right side of the quadrant, which effectively means that we add a contribution from the j-th column, instead of a contribution from the i-th row, as Kubina, 2017 does. We prefer this version because it allows more straightforward generalisation to 3D.

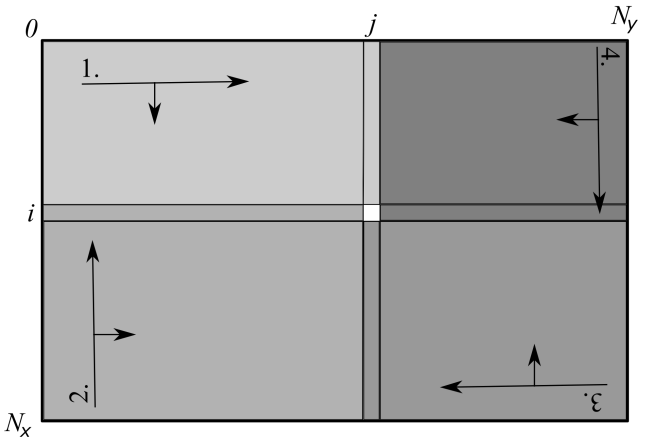


Figure 3.2: Four quadrants with attached interfaces with indicated directions of application of the algorithm within respective quadrants. The point (i, k) does not belong to any quadrant

The modified algorithm is as follows:

```
#a[nx][ny]...input 2D array
#b[nx][ny]...output 2D array
#tmp1[ny]...auxiliary 1D array
#tmp0...auxiliary variable
#cx,cy...multiplicative constants in x/y direction

b=copy(a)
set_to_zeroes(tmp1)
for i in range(nx):
    tmp0=0
    for j in range(ny):
        tmp1[j]*=cx
        b[i][j]+=tmp1[j]
        tmp0+=a[i][j]
        tmp1[j]+=tmp0
        tmp0*=cy
```

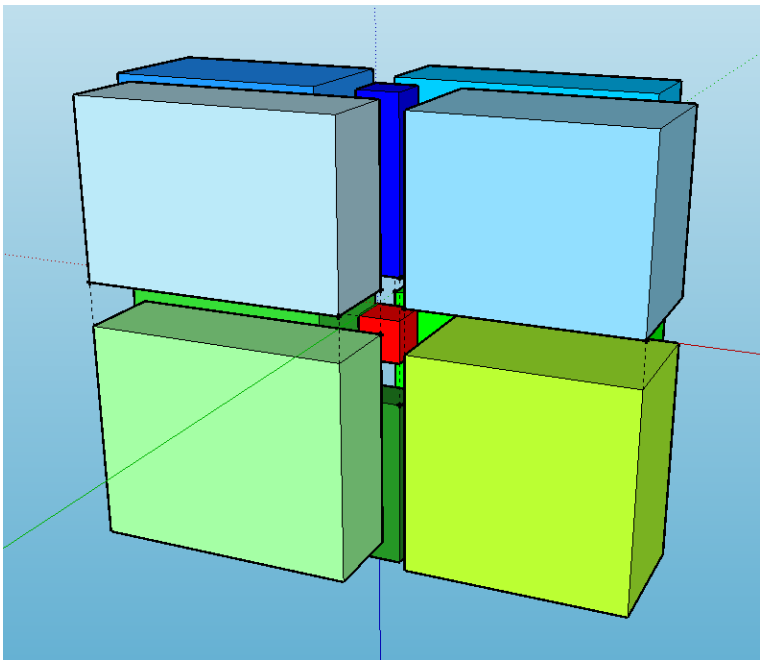
A contribution from the i-th row in the first quadrant is omitted, as it will be added as a contribution from the last column in the next quadrant. We cannot forget to add a contribution from the point (i, j).

Dealing with interfaces is more difficult in 3D. In this case, there are eight octants and twelve interfaces between them, thus it is not possible to attach interfaces to the quadrants in such a way that all octants are same. Therefore, we cannot use a same algorithm for each octant but some adjustments of the algorithm among different octants are needed.

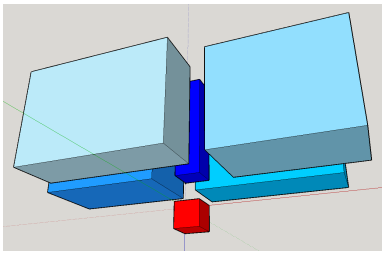
There are many solutions for this problem. We present one which requires application of only two different modifications of the algorithm and which is the most straightforward generalisation of the 2D algorithm.

Vertical interfaces are attached in the same way as they are in the 2D case. Horizontal interfaces are all attached to bottom octants. Therefore, we do not add a contribution from the actual plane in upper octants, whilst we do in bottom octants. It is depicted on Fig. 3.3.

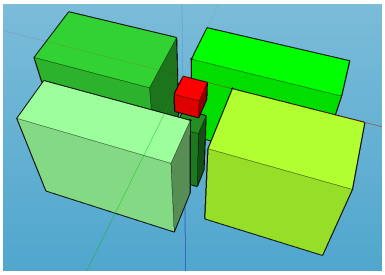
A summation is performed plane-wise and rescaled contributions from planes are added together. A contribution from a single plane is calculated in the same way, as it was in 2D. However, it causes that points exactly above or below the point (i, j, k) are not involved, thus a contribution from these points has to be calculated additionally.



(a) Overall view



(b) Top octants. Points situated exactly above the point(i, j, k) do not belong to any octant



(c) Bottom octants. Points situated exactly below the point(i, j, k) do not belong to any octant

Figure 3.3: Visualisation of octants. Gaps are inserted between blocks for better clarity

The algorithm for upper octants is:

```
#a[nz][nx][ny]...input 3D array
#b[nz][nx][ny]...output 3D array
#tmp1[ny]...auxiliary 1D array
#tmp2[nx][ny]...auxiliary 2D array
#tmp0...auxiliary variable
#cx,cy,cz...multiplicative constants in x/y/z direction
```

```
b=copy(a)
set_to_zeroes(tmp2)
for k in range(nz):
    set_zeroes(tmp1)
    for i in range(nx):
        tmp0=0
        for j in range(ny):
            tmp2[i][j]*=cz
            b[k][i][j]+=tmp2[i][j]
            tmp2[i][j]+=tmp1[j]
            tmp0+=a[k][i][j]
            tmp1[j]+=tmp0
            tmp0*=cy
            tmp1[j]*=cx
```

It is applied from different directions in other upper octants, analogously to the 2D case.

The algorithm for bottom octants is:

```
set_to_zeroes(tmp2)
for k in range(nz-1,-1,-1):
    set_zeros(tmp1)
    for i in range(nx):
        tmp0=0
        for j in range(ny):
            tmp2[i][j]=cz*tmp2[i][j]+tmp1[j]
            b[k][i][j]+=tmp2[i][j]
            tmp0=cy*tmp0+a[k][i][j]
            tmp1[j]=cx*(tmp1[j]+tmp0)
```

Finally, a contribution from points exactly above/below the point (i, j, k) is calculated as follows:

```
for i in range(nx):
    for j in range(ny):
        tmp0=0
        for k in range(nz):
            b[k][i][j]+=tmp0
            tmp0+=a[k][i][j]
            tmp0*=cz
        tmp0=0
        for k in range(nz-1,-1,-1):
            b[k][i][j]+=tmp0
            tmp0+=a[k][i][j]
            tmp0*=cz
```

3.2.1.3 Normalisation So far, we have found an algorithm for efficient computation of convolution of an arbitrary function with an exponential function. In fact, it is not a correct result yet – it has to be multiplied by grid steps in respective directions. We intentionally omitted this step, as we will perform it together with normalisation.

A smoothing function has one important property. An integral of the smoothing function is equal to one. It can be achieved in two different ways. We can either normalise the smoothing function which means that we divide it by an integral of it what secures that the integral of the function is equal to one, or we can firstly calculate a convolution integral with non-normalised function and then divide it by a sum of weights.

The presented algorithm cannot be applied with a normalised exponential smoothing function because a normalisation is space-dependent, therefore a result has to be divided by a sum of weights at the end. The sum of weights can be calculated in two different ways.

Kubina, 2017 applied exactly the same algorithm also for a computation of the sum of weights. He only replaced the term $\mathbf{K}_{\iota\gamma\kappa}$ in eq. (3.28) by 1 and computed it along with a convolution. A drawback of this approach is that it requires an additional 3D array for storing of the sums of weights.

The second option is to calculate the sum of weights analytically. We notice that the weights forms a geometric series

$$W_{i\,j\,k} = \sum_{\iota=0}^{N_x-1} C_x^{|i-\iota|} \sum_{\gamma=0}^{N_y-1} C_y^{|j-\gamma|} \sum_{\kappa=0}^{N_z-1} C_z^{|k-\kappa|}, \tag{3.32}$$

thus it can be summed directly

$$\sum_{\iota=0}^{N_x-1} C_x^{|i-\iota|} = \sum_{\iota=0}^{i} C_x^{i-\iota} + \sum_{\iota=i+1}^{N_x-1} C_x^{\iota-i} = \frac{1 - C_x^{N_x-i} + C_x\,(1 - C_x^{i})}{1 - C_x}. \tag{3.33}$$

Therefore

$$W_{i\,j\,k} = \frac{1 - C_x^{N_x - i} + C_x \left(1 - C_x^i\right)}{1 - C_x} \cdot \frac{1 - C_y^{N_y - j} + C_y \left(1 - C_y^j\right)}{1 - C_y} \cdot \cdot \frac{1 - C_z^{N_z - k} + C_z \left(1 - C_z^k\right)}{1 - C_z}. \tag{3.34}$$

Let's note that the sum of weights is not equal to the integral of the smoothing function. It can be easily demonstrated. Consider cuboidal domain $\langle 0; L_x \rangle \times \langle 0; L_y \rangle \times \langle 0; L_z \rangle$. Then according to eq. (3.3),

$$\mathbf{W}\left(\mathbf{r}\right) = \int_0^{L_x} \mathrm{e}^{-\frac{|x-\xi|}{\Lambda_x}} \,\mathrm{d}\xi \int_0^{L_y} \mathrm{e}^{-\frac{|y-\eta|}{\Lambda_y}} \,\mathrm{d}\eta \int_0^{L_z} \mathrm{e}^{-\frac{|z-\zeta|}{\Lambda_z}} \,\mathrm{d}\zeta. \tag{3.35}$$

It can be easily calculated that

$$\int_0^{L_x} \mathrm{e}^{-\frac{|x-\xi|}{\Lambda_x}} \,\mathrm{d}\xi = 2\Lambda_x \left[1 - \mathrm{e}^{-\frac{L_x}{2\Lambda_x}} \cosh\left(\frac{x}{\Lambda_x} - \frac{L_x}{2\Lambda_x}\right)\right]. \tag{3.36}$$

Same results are obtained also in the other dimensions, thus

$$\mathbf{W}\left(\mathbf{r}\right) = 8\Lambda_x\Lambda_y\Lambda_z \left[1 - \mathrm{e}^{-\frac{L_x}{2\Lambda_x}} \cosh\left(\frac{x}{\Lambda_x} - \frac{L_x}{2\Lambda_x}\right)\right] \left[1 - \mathrm{e}^{-\frac{L_y}{2\Lambda_y}} \cosh\left(\frac{y}{\Lambda_y} - \frac{L_y}{2\Lambda_y}\right)\right] \left[1 - \mathrm{e}^{-\frac{L_z}{2\Lambda_z}} \cosh\left(\frac{z}{\Lambda_z} - \frac{L_z}{2\Lambda_z}\right)\right] \tag{3.37}$$

We can discretise the right-hand side of eq. (3.36) using $x = ih$ and $L_x = (N_x - 1)\,h$. Furthermore $C_x = \mathrm{e}^{-\frac{h}{\Lambda_x}} \approx 1 - \frac{h}{\Lambda_x}$, thus $\Lambda_x \approx \frac{h}{1 - C_x}$. We obtain

$$\int_0^{L_x} \mathrm{e}^{-\frac{|x-\xi|}{\Lambda_x}} \,\mathrm{d}\xi = \Lambda_x \left(2 - C_x^i - C_x^{N_x - i - 1}\right) \approx h\frac{2 - C_x^i - C_x^{N_x - i - 1}}{1 - C_x}. \tag{3.38}$$

Comparing with eq. (3.33) which can be rewritten in the form

$$\sum_{\iota=0}^{N_x - 1} C_x^{|i-\iota|} = \frac{1 + C_x \left(1 - C_x^i - C_x^{N_x - i - 1}\right)}{1 - C_x}, \tag{3.39}$$

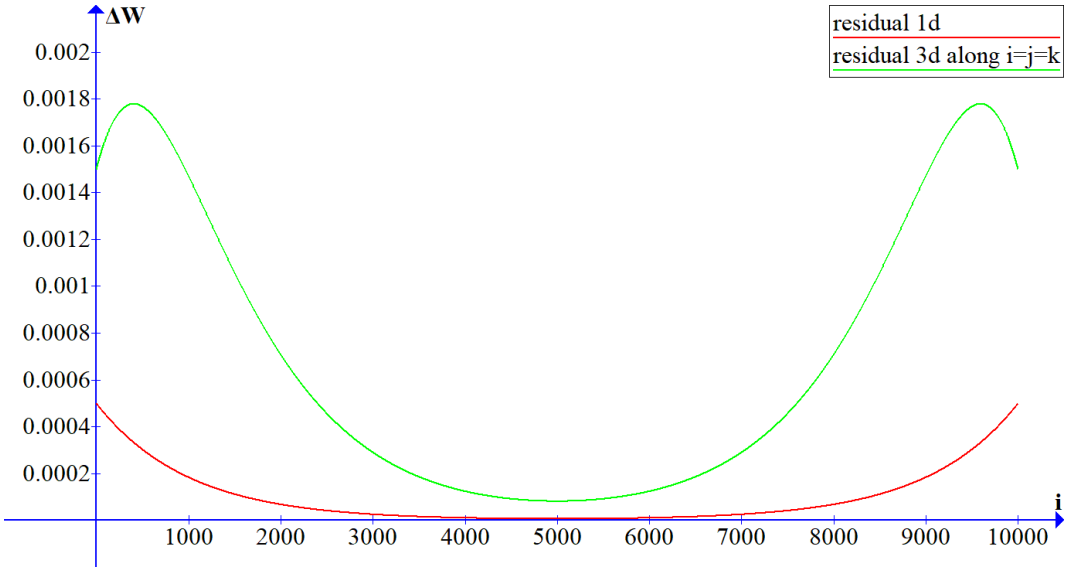


Figure 3.4: Residuals between a sum of weights and an integral of the smoothing function. A red line represents a 1D case and a green line corresponds to a cut through a 3D plot along $i = j = k$

we conclude that it differs only by a factor C_x which is anyway close to 1 as $\frac{h}{C_x} \ll 1$. (We remind that we omitted multiplication by grid step h when we calculated the sum.)

Residuals between the sum of weights and the discretised integral of the smoothing function are depicted on Fig. 3.4. Although they are small, we prefer a calculation of the sum of weights according eq. (3.34). We prefer a direct calculation of the sum to Kubina's approach, so that we do not need to store an additional 3D array. We already use two 3D arrays – one for an original kernel and one for a smoothed one – and the third 3D array would increase memory requirements significantly. A number of grid points is quite large in 3D. It can be of order $N^3 \sim 10^7 - 10^8$. A required memory for storing of one number of type float is 4 MB, therefore such a 3D array would require additional $\sim$ 380 MB of memory. The spared memory allows us to smooth larger kernels.

3.2.1.4 Time complexity Recall that a time complexity of a direct computation of convolution in 3D is $\mathcal{O}(N^6)$. If a smoothing function is effectively non-zero only in a smaller spatial window,

it can be reduced to $n^3\mathcal{O}\left(N^3\right)$, where n^3 is a multiplicative constant. Further reduction of a computational time can be achieved with a parallelisation of computation. However, the multiplicative constant remains large.

The presented efficient algorithm consists of three-level nested loops. They appear ten times – once in each octant, once when calculating a contribution from points above/below the actual point and once when carrying out normalisation. Therefore, a time complexity of the efficient smoothing algorithm is $10\mathcal{O}\left(N^3\right)$. In fact the multiplicative constant is larger than 10, as a single three-level nested loop consist of more than one operation.

One drawback of the algorithm is that it cannot be effectively parallelised. It is because the algorithm has to be performed in a special order, as described on page 55, which disallows any kind of parallelisation. Despite of it, it is still significantly more efficient than even a parallelised version of a direct computation of convolution over a smaller spatial window.

Let's note that a time complexity of $\mathcal{O}\left(N^3\right)$ is optimal. It is not possible to design a better algorithm in terms of time complexity. We need to calculate a smoothed value at N^3 grid points, thus the time complexity has to be at least $\mathcal{O}\left(N^3\right)$.

3.2.1.5 Directional dependence of smoothing We will now focus on a shape of the exponential smoothing function (3.43). Isosurfaces of the smoothing function satisfy condition

$$\left\| \mathbf{\Lambda^{-1}} \cdot (\mathbf{r} - \mathbf{r}) \right\|_1 = \text{const.} \tag{3.40}$$

It is the equation of an octahedron. If the smoothing is isotropic, it is a regular octahedron, otherwise it is a stretched one.

Diagonals of the octahedrons are aligned with Cartesian axes. It has a severe consequence – smoothing in the direction of Cartesian

axes is more intense than in any other direction. In other words, smoothing is strongly angle-dependent.

Wellington, 2016 considers it to be a severe problem and states that such a smoothing function is not suitable because it introduces characteristics that are not geologically justified. However, one must realise that it does not introduce any octahedral-shaped structures into the smoothed kernel, as each point of the kernel is smoothed in the same way. Furthermore, it is a common property of FD methods that just points in directions of Cartesian axes are used to approximate derivatives, thus it completely makes a sense that those points have higher weight than equally distant points in any other direction.

More severe limitation of the efficient smoothing algorithm is that main axes of anisotropic smoothing have to be aligned with Cartesian axes. It is not possible to adjust it to be able to perform the anisotropic smoothing with the main axes rotated with respect to the Cartesian axes. It is a trade-off between a reduction of computational time and a complexity of the algorithm.

We usually do not have prior information on a direction of the main axes and we usually require that the main smoothing axes are aligned with edges of the domain and their ratio is equal to the ratio of lengths of domain edges.

3.2.1.6 Spectrum of smoothing function – transfer function

More important characteristics of a smoothing function is its spectrum. A smoothing operator serves as a low-pass filter. It suppresses high frequencies. That means it is effectively zero for high wave numbers.

Let's find a transfer function corresponding to an exponential smoothing function

$$\hat{w}\left(\mathbf{k}\right) = \mathscr{F}\left[\mathrm{e}^{-\left\|\boldsymbol{\Lambda}^{-1}\cdot\left(\mathbf{r}-\mathbf{r}\right)\right\|_1}\right] = \int_{-\infty}^{\infty} \mathrm{e}^{-\left\|\boldsymbol{\Lambda}^{-1}\cdot\left(\mathbf{r}-\mathbf{r}\right)\right\|_1} \mathrm{e}^{-\mathrm{i}\mathbf{k}\cdot\mathbf{r}}\, \mathrm{d}^3 r. \quad (3.41)$$

It can be easily calculated that

$$\int_{-\infty}^{\infty} \mathrm{e}^{-\frac{|x|}{\Lambda_x}} \mathrm{e}^{-\mathrm{i}k_x x}\,\mathrm{d}x = \frac{2\Lambda_x}{1+\left(\Lambda_x k_x\right)^2}, \tag{3.42}$$

thus

$$\hat{w}\left(\mathbf{k}\right) = \frac{8\Lambda_x\Lambda_y\Lambda_z}{\left[1+\left(\Lambda_x k_x\right)^2\right]\left[1+\left(\Lambda_y k_y\right)^2\right]\left[1+\left(\Lambda_z k_z\right)^2\right]}. \tag{3.43}$$

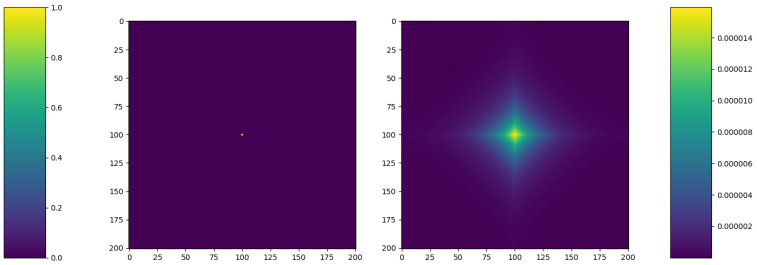
It is a Lorentzian-like function. Its slice is a typical bell-shaped function, thus it meets a requirement of effectively zero values for high wave numbers. However, the smoothing function maintains its strong directional dependence also in a wave-number domain.

Let's note that (3.43) is not exactly a transfer function of the applied filter. It is because we use a normalised filter and a normalisation is not constant but it is a spatially dependent (see (3.37)). Therefore, a spectrum of the normalised smoothing function is slightly different.

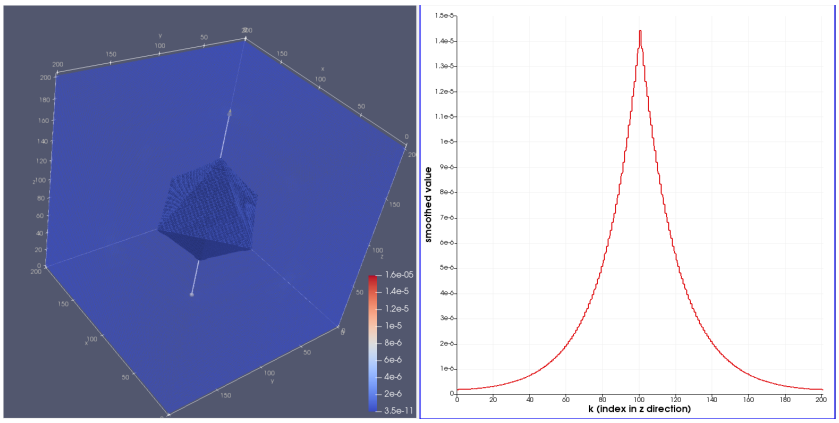
It is important to select smoothing intensities properly. In a wave-number domain, it corresponds to choosing of cut-off wave numbers. The cut-off wave numbers are usually selected such that an amplitude at the cut-off wave number is a half of a maximum amplitude (or equivalently lower by 3 dB), thus it has to satisfy a condition $\Lambda \cdot k = 1$. Therefore

$$\mathbf{k}^{\mathrm{cut-off}} = \left(\frac{1}{\Lambda_x}, \frac{1}{\Lambda_y}, \frac{1}{\Lambda_z}\right) = \boldsymbol{\Lambda}^{-\mathbf{1}}. \tag{3.44}$$

eq. (3.44) allows us to relate the smoothing intensities to the cut-off wave number and hence to a cut-off frequency of input data. Assume that the input data has been filtered with a cut-off frequency ω. Therefore the highest reasonable wave number is $k = \frac{\omega}{v}$, where v is an estimation of velocity of the slowest wave, and hence the lowest smoothing intensity is $\Lambda = \frac{v}{\omega}$.



(a) Horizontal slices through the centres of original and smoothed array



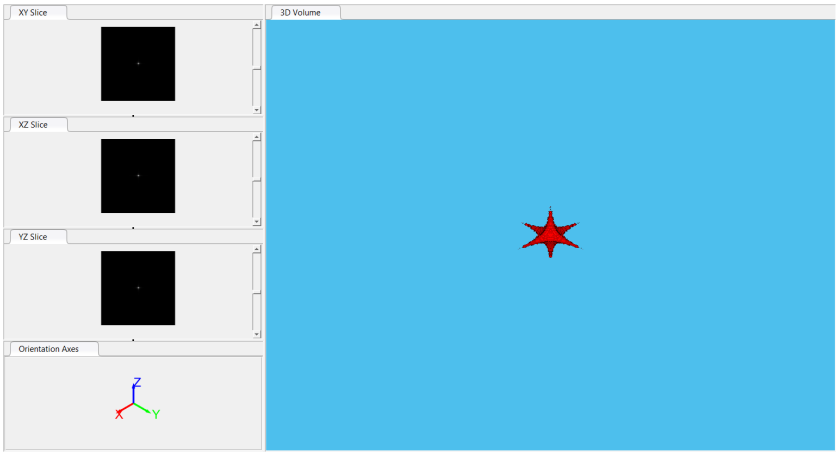
(b) An isosurface of smoothed array and a profile along a vertical line through the centre

Figure 3.5: A spike test

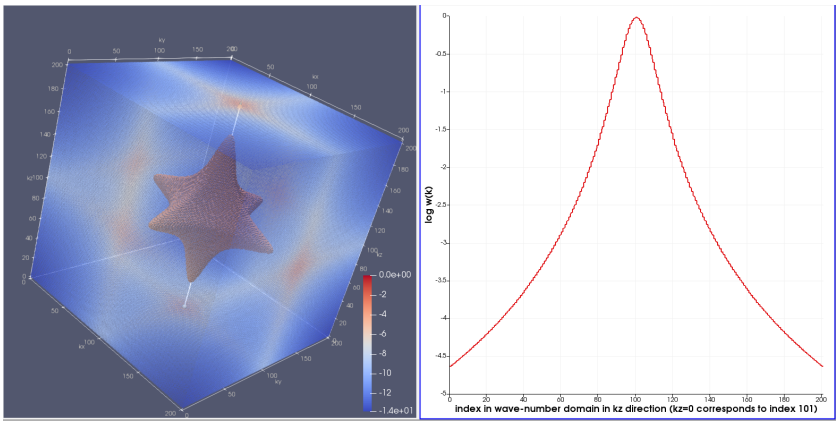
3.2.2 Numerical tests

3.2.2.1 Spike test We start with a verification of the algorithm. To do so, we create a 3D array with all but one points set to zero. Only a central point is equal to one. We expect to obtain a smoothed array in shape of a smoothing function.

A visualisation of the smoothed array is depicted on Fig. 3.5. Fig. 3.5a shows horizontal slices through the array before and after smoothing. A slice through the original array contains single non-zero point of the array. A same slice through the smoothed array shows a typical square shape of a cross-section of the smoothing function. Fig. 3.5b depicts an octahedral isosurface of the smoothing function. A blue cube is an entire domain. The right-



(a) A smoothed array in wave-number domain



(b) A predicted spectrum of the smoothing function, e. i. a transfer function

Figure 3.6: A spectrum of the smoothed array

hand-side plot of Fig. 3.5b is a cut through the smoothed array along a z axis through the centre. It shows an exponential decay of values.

Let's look on a spectrum of the smoothed array. The spectrum of the original array is constant as it contains only a Delta-like peak, thus the spectrum of the smoothed array corresponds to a spectrum of the smoothing function. Therefore, we can compare it with a theoretical prediction based on eq. (3.43). Fig. 3.6 shows an agreement between the theoretical predictions (Fig. 3.6b) and a spectrum obtained by performing of discrete fast Fourier transform of the smoothed array (Fig. 3.6a). The spectrum was calculated by `fftn` function in Matlab. The resulted array

was transformed in such a way that $\mathbf{k} = 0$ was mapped into the centre of the domain.

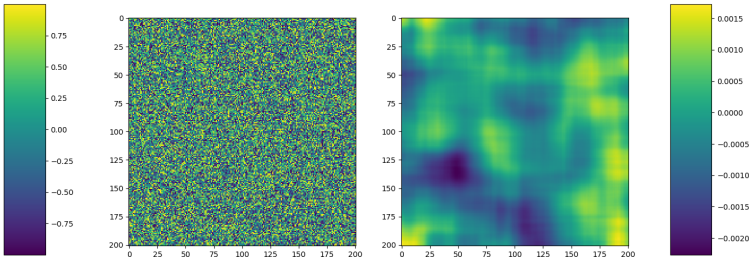
3.2.2.2 Smoothing of white noise Now we verify the ability of the algorithm to smooth out a random noise. We generated a white noise by attributing a random value from the uniform distribution on interval $\langle -1; 1 \rangle$ to each point. Then we applied the smoothing algorithm. The results are presented on Fig. 3.7.

First of all, we notice that smoothed values are 3 orders of magnitude lower, thus we can conclude that the noise was successfully removed. Only deficiency is that the effect of dominant smoothing in the directions of the Cartesian axes can be observed in the structure of the smoothed array.

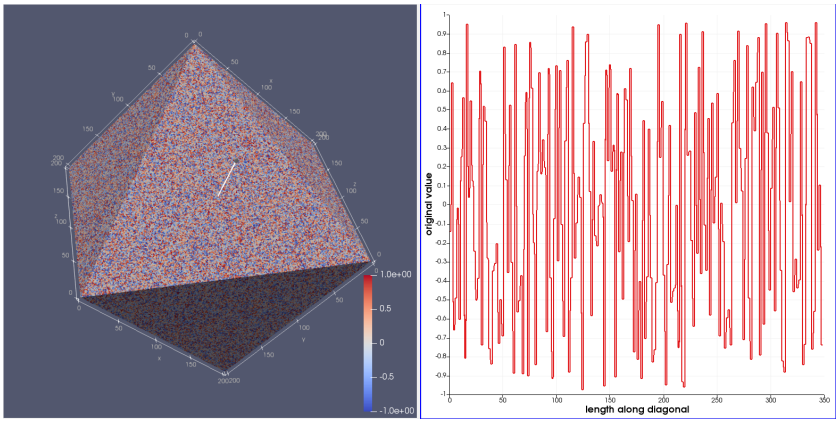
We used isotropic smoothing with a characteristic length corresponding to 20 grid points. We can notice on the cut along diagonal through the smoothed array that it agrees with a characteristic length of oscillations. Note that 20 grid points correspond to the length of $20\sqrt{3} \doteq 35$ along diagonal.

Finally, let's look at spectra of original and smoothed array (Fig. 3.8). A spectrum of the original array contains all wave numbers with approximately equal amplitudes. A spectrum of the smoothed array has a characteristic radial shape. It may be a bit counter-intuitive as the smoothing is stronger right in directions of Cartesian axes. However, realise that the Cartesian axes in the space domain and in the wave-number domain are not the same. Whilst the Cartesian axes in the space domain are directions of stronger smoothing, the Cartesian axes in the wave-number domain are places of long waves in respective directions which are thus smoothed more weakly.

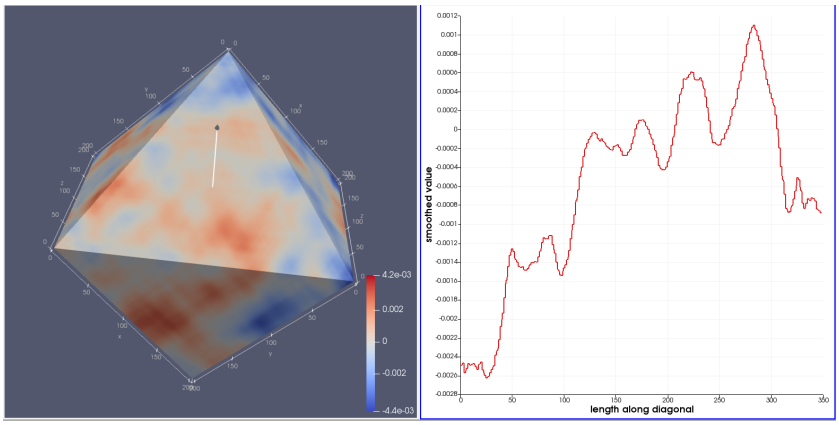
3.2.2.3 Signal overlaid by noise Next, we verify the ability of the algorithm to smooth out a random noise whilst preserving a signal. As a signal, we used a Gaussian peak of unitary height



(a) Slices through original and smoothed array

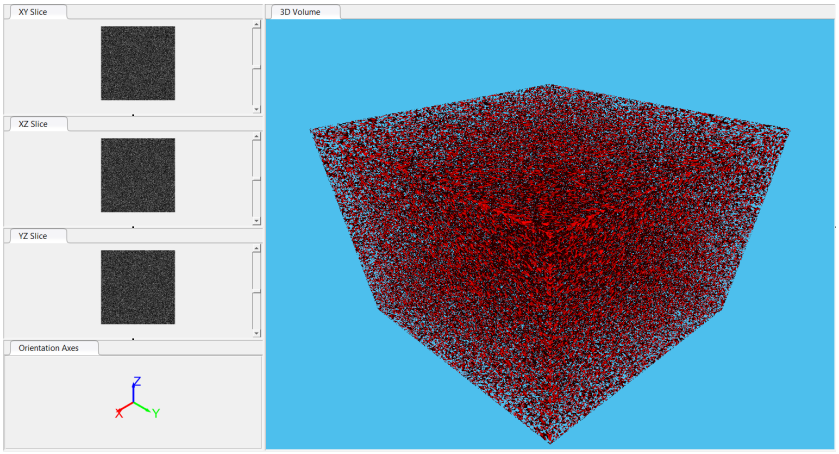


(b) A clipped 3D outlook of the original array and a profile along its diagonal

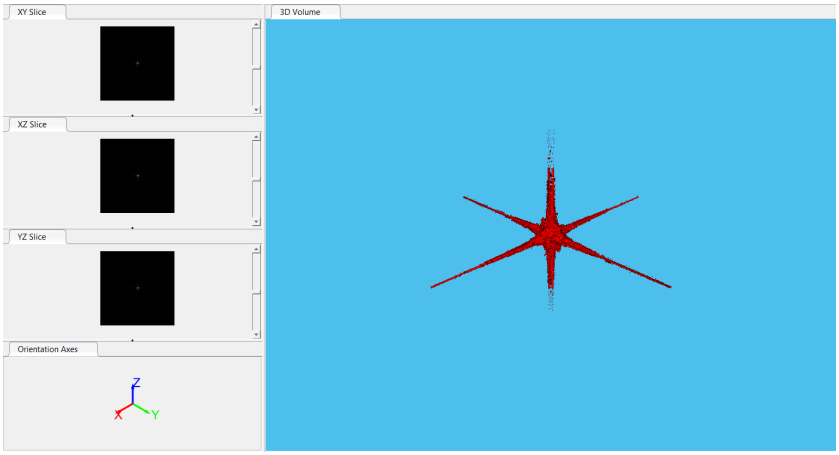


(c) A clipped 3D outlook of the smoothed array and a profile along its diagonal

Figure 3.7: Smoothing of white noise

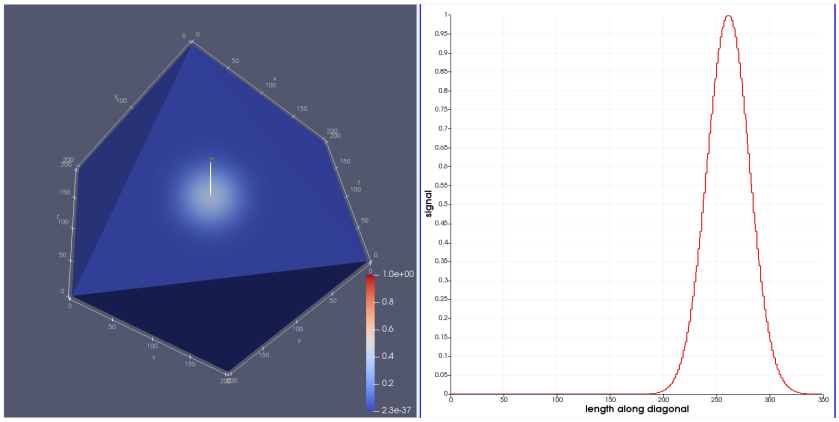


(a) A spectrum of the original array contains all frequencies

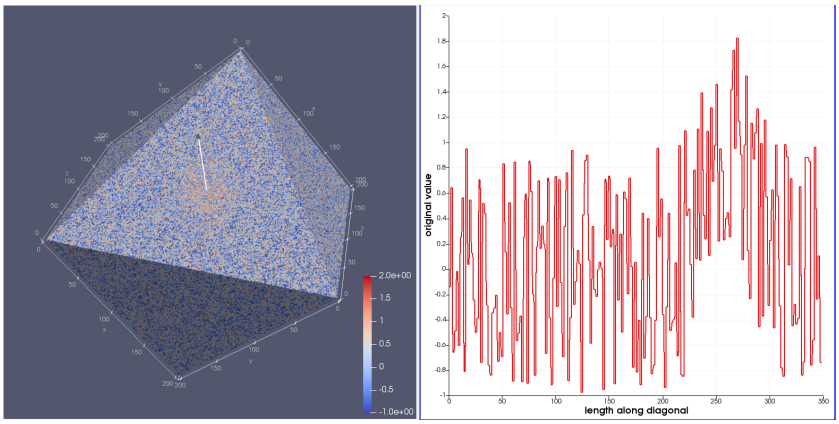


(b) A spectrum of the smoothed array has a typical radial shape

Figure 3.8: Spectra of original and smoothed array



(a) A Gaussian signal

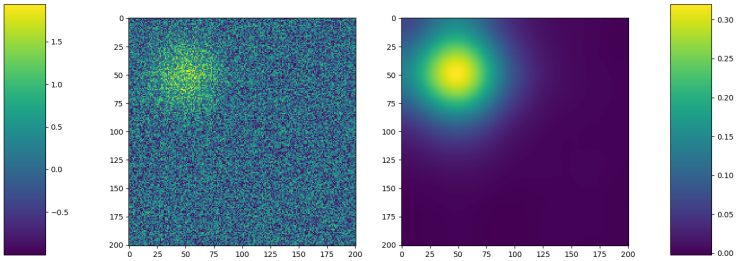


(b) The signal overlaid by a noise

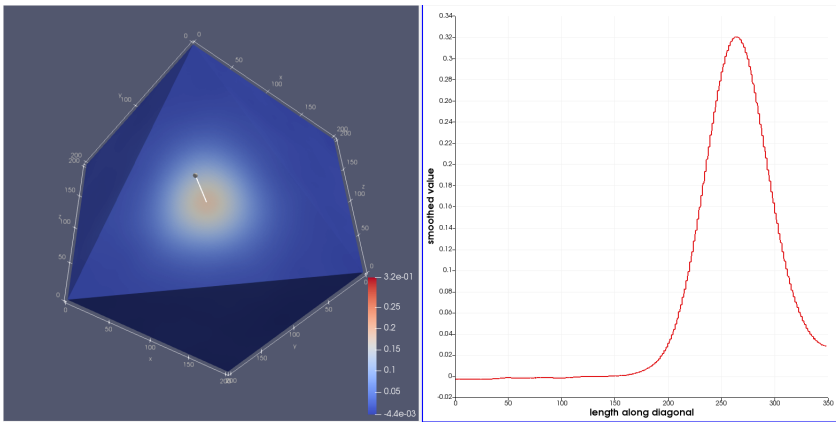
Figure 3.9: Input data

with $\sigma/h = 20$ grid points, located in the middle of an octant (Fig. 3.9a). Then we overlaid it by a random noise of same amplitude. The input array is illustrated on Fig. 3.9b.

Consequently, we performed an isotropic smoothing with three distinct smoothing intensities. We started with a characteristic smoothing length of $\Lambda/h = 20$ grid points, what is a standard deviation of the signal (Fig. 3.10). We observe that the signal is quite well-preserved, whilst the noise is removed. However, an amplitude of the signal decreased approximately to a third of the original amplitude. It is like that due to a fact that the signal contained a considerable amount of energy also in frequencies higher than a cut-off frequency of the filter.



(a) Slices through a centre of the Gaussian signal before and after application of smoothing

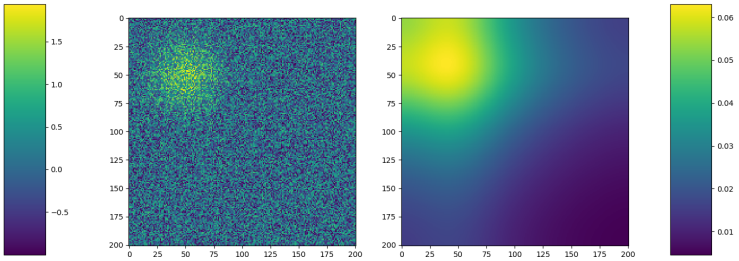


(b) A clipped 3D outlook of the smoothed array and a profile along diagonal

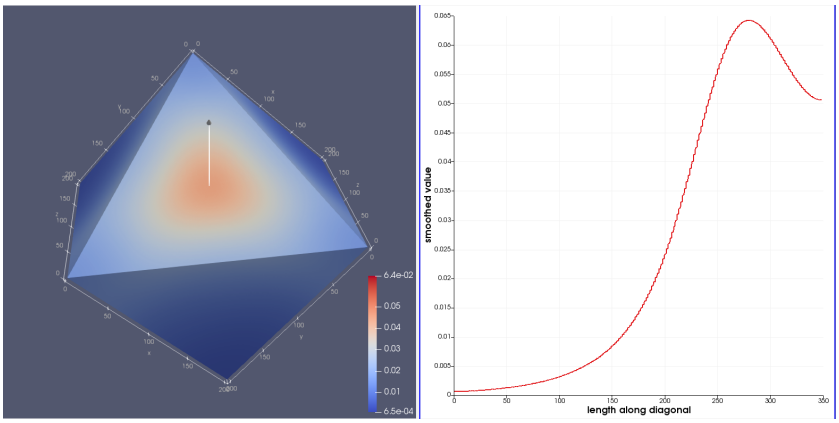
Figure 3.10: Smoothed data with $\Lambda = 20$ grid points

However, we rarely know a characteristic length of the signal, thus we cannot design the filter for a particular situation. Instead, we use a multiscale approach presented in the section 2.4.1. That means we start at the highest scale, thus using a very strong filter. Let's test such a strong filter with $\Lambda/h = 80$ grid points. The obtained results are depicted on Fig. 3.11. We see that the noise is completely removed, but the signal is significantly suppressed. However, the data are smooth and simple, which is important for updating the model on the highest scales in the multiscale approach.

Finally, let's test out a smoothing on the lowest scales. The smoothing on the lowest scales has to be weak in order to pre-



(a) Slices through a centre of the Gaussian signal before and after application of smoothing

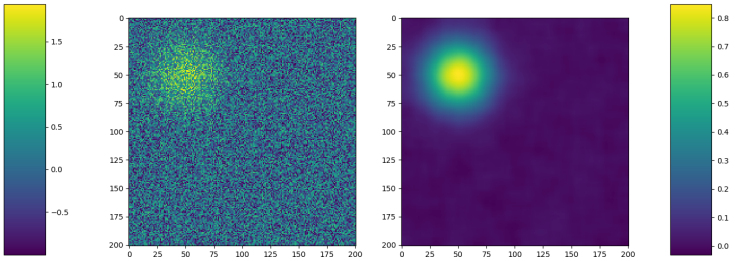


(b) A clipped 3D outlook of the smoothed array and a profile along diagonal

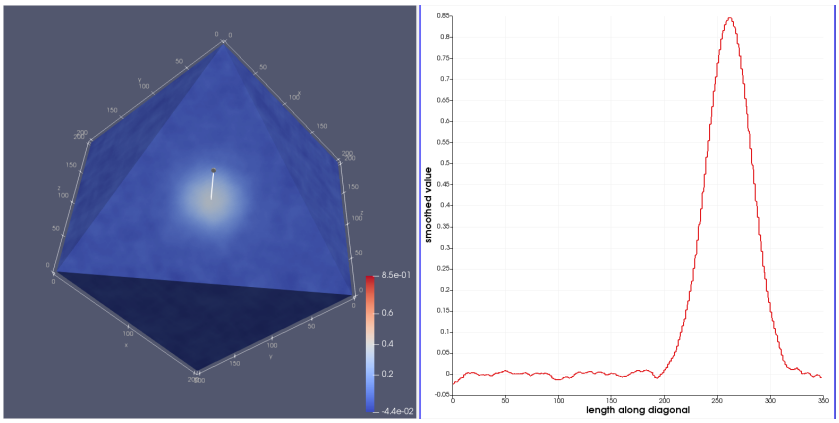
Figure 3.11: Smoothed data with $\Lambda = 80$ grid points

serve fine structures. The results of the smoothing with $\Lambda/h = 5$ grid points are shown on Fig. 3.12. We see that the noise is quite well suppressed, but it is still clearly visible. On the other hand, the signal is only slightly affected by the smoothing. Therefore, we can conclude that it is crucial to find a trade-off between suppressing of the noise and preserving of the signal.

3.2.2.4 Test of time complexity At the end, we tested a time complexity of the algorithm. We used a same input array as for a spike test, in order to minimise a time necessary for creating/loading of the input array. We kept smoothing intensities constant, as a number of computational operations is independent of them, and we were changing a size of the array.



(a) Slices through a centre of the Gaussian signal before and after application of smoothing



(b) A clipped 3D outlook of the smoothed array and a profile along diagonal

Figure 3.12: Smoothed data with $\Lambda = 5$ grid points

We compare two versions of the smoothing algorithm which differs by a way of computation of sums of weights for normalisation. In one version, the sums are calculated directly using eq. (3.34), in the other, they are computed by the efficient smoothing algorithm itself along with a computation of convolution.

The test was carried out on my laptop with 4 cores (Intel Core i5 2.5 GHz) and 8 GB RAM. We carried it out for 6 sizes of the array selected logarithmically in a range between $N = 15$ and $N = 511$ and we repeated it five times for each size. The reason for repeating was that there were some other system processes running on the laptop which were varying over time and we needed

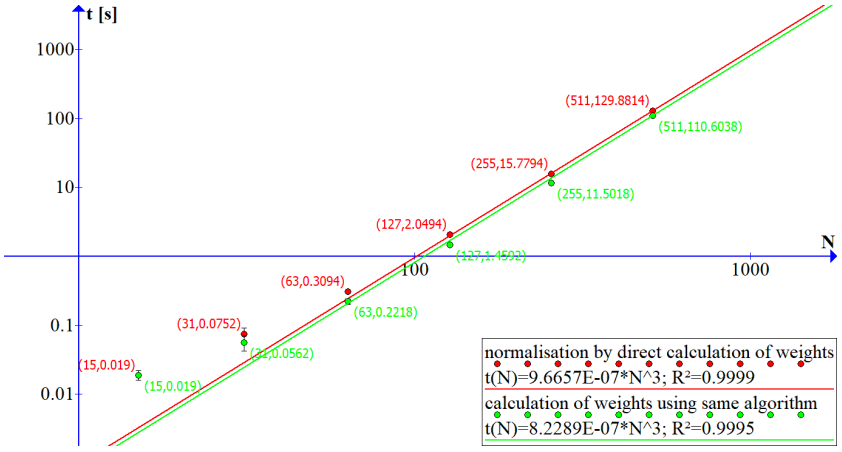


Figure 3.13: Test of time complexity of the smoothing algorithm

to average their effect. A test for $N = 1023$ could not have been carried out due to a lack of available memory in my laptop.

Results are illustrated on Fig. 3.13. A red sequence of points represents averages of computational times of the algorithm with a direct calculation of sums of weights. It is fitted by a function $t(N) = c \cdot N^3$. The best fit is found for $c = 9.6657 \cdot 10^{-7}$ with coefficient of determination $R^2 = 0.9999$. A green sequence represents averages of computational times of the algorithm with sums of weights computed by the same algorithm along with a computation of convolution. It is fitted by the same function with $c = 8.2289 \cdot 10^{-7}$ and with a coefficient of determination $R^2 = 0.9995$. In both cases, a good fit is obtained what confirms that a time complexity of the algorithm is $\mathcal{O}(N^3)$.

The ratio of constants is $\frac{9.6657 \cdot 10^{-7}}{8.2289 \cdot 10^{-7}} \doteq 1.1746$, thus there is only a slight difference between two versions. If we consider an array with $N = 1000$, a computational time is about 15 minutes and the difference between versions is only 2 minutes which is negligible. On the other hand, the version with a direct calculation of the sums of weights requires significantly less memory, thus we prefer that one.

4 Mask

4.1 Purpose of mask

An application of a mask is both regularisation and preconditioning technique. It prevents a model from being updated in its certain parts by modifying a kernel. There are several reasons for that.

As a regularisation technique, it ensures that a model is updated only in its unknown parts. We assume that we know a bedrock properly, thus we want the kernel to be modified only in a sedimentary basin. Therefore, the mask should null the kernel everywhere outside the basin but keep it unchanged inside. It means that in an ideal case, the mask should have a shape of the basin. However, it is usually unknown, hence we need to design the mask in a such way that the entire basin lies in a so-called full-gain zone of the mask.

As a preconditioning technique, the mask is used to improve a stability of an inversion. The inverse problem is often ill-conditioned in the deep parts of the model due to little data, therefore any attempts to update the deep parts of the model are unlikely to be correct and can destabilise the inversion. Therefore, the mask should null the kernel in the depth.

The application of the mask allows a computational time of the inversion to be reduced significantly. If we restrict the model to being updated only in its upper regions, we do not have to simulate propagation of seismic waves in the deep parts repeatedly, but instead we can use an excitation box (see e. g. Oprsal et al., 2002). Then the propagation has to be simulated outside the box only once and we restrict ourselves to updating of the model only inside the box which means that the mask should null everything outside the box and preserve everything inside.

4.2 Mask construction

4.2.1 Mask design

The excitation box divides the model into two parts. The outer part should not be modified, thus the corresponding part of the mask is a so-called no-gain zone. In the ideal case, the entire inner part of the model would belong to a full-gain zone of the mask. However, such a mask design would cause a sharp artificial discontinuity in the model. Therefore, a transitional zone has to be present in between, in which a nature of the mask changes from no-gain to full-gain smoothly. The transitional zone should lie inside the box in a vicinity of its walls.

We will consider a kernel component $K(\mathbf{r})$. A mask can be easily implemented as a multiplication of the mask $M(\mathbf{r})$ with the kernel component

$$K^{\text{masked}}(\mathbf{r}) = M(\mathbf{r}) \cdot K(\mathbf{r}). \tag{4.1}$$

In this case, the mask is formed by zeroes in the no-gain zone and by ones in full-gain zone. Values in transitional zone gradually increase from 0 to 1.

An important property of the mask is a change of a spectrum of the kernel. The mask definitely affects the spectrum, but ideally it should not introduce higher wave numbers into the spectrum of the kernel. And that is the problem. We need to not only design the mask, so that its spectrum does not contain higher wave numbers than a certain cut-off wave number, but also a spectrum of the modified kernel is calculated as a convolution of a spectrum of the original kernel with a spectrum of the mask and a convolution in a wave-number domain always introduces higher wave numbers. Moreover, we have not mentioned yet that we need a different mask on each scale.

The other option is to design the mask $\delta K(\mathbf{r})$ in such a way that it is added to the kernel

$$K^{\text{masked}}(\mathbf{r}) = K(\mathbf{r}) + \delta K(\mathbf{r}). \tag{4.2}$$

In a no-gain zone, the mask satisfies $\delta K(\mathbf{r}) = -K(\mathbf{r})$ and in a full-gain zone $\delta K(\mathbf{r}) = 0$. An advantage of this approach is that a spectrum of the modified kernel is a sum of the original spectrum and a spectrum of the mask, which can be easily controlled.

Realise that there are no specific requirements on a shape of the transition zone. Only limitation is the spectrum. Therefore, we can choose any reasonable shape of the transition zone.

Kubina, 2017 suggested an interested way how to construct the mask. He proposed that it can be find as a solution of Laplace equation

$$\triangle \delta K = 0. \tag{4.3}$$

He took inspiration from a potential of an electric field. He argues that kernel values decrease as $1/r$ which is the same dependence on the distance as for the potential of an electric source. Then he finds the mask as a potential of a source which is located outside the box with such boundary conditions that it nulls the kernel at walls of the excitation box. The satisfactory boundary conditions are

$$\delta K(\mathbf{r}) = -K(\mathbf{r}) \tag{4.4a}$$

at all faces except the top one and

$$\delta K(\mathbf{r}) = 0 \tag{4.4b}$$

on a free surface.

We stress that there is no relation between an electric potential and the kernel or the mask. We calculate the mask using Laplace equation only because we can. We can choose any reas-

onable shape of the mask, but it turns out that it is handy to find the mask as a solution of the Laplace equation, because it allows us to directly control the spectrum of the mask and to construct similar masks at different scales easily.

4.2.2 Solution of Laplace equation

We are interested in solving of eq. (4.3) in a cuboid $\langle 0; L_x \rangle \times \langle 0; L_y \rangle \times \langle 0; L_z \rangle$ with boundary conditions (4.4). We will seek a solution in a form

$$\delta K(\mathbf{r}) = X(x) \cdot Y(y) \cdot Z(z). \tag{4.5}$$

Inserting (4.5) into eq. (4.3) and dividing it by δK gives

$$\frac{\triangle \delta K}{K} = \frac{X''}{X} + \frac{Y''}{Y} + \frac{Z''}{Z} = 0. \tag{4.6}$$

Notice that each summand is dependent on a different variable, thus if it is to be equal to zero for each combination of variables, each summand has to be constant. Therefore eq. (4.6) is split into a set of three ordinary differential equations

$$\frac{X''}{X} = A; \qquad \frac{Y''}{Y} = B; \qquad \frac{Z''}{Z} = C \tag{4.7}$$

tied by an algebraic equation

$$A + B + C = 0. \tag{4.8}$$

A solution of an equation

$$X'' - A \cdot X = 0 \tag{4.9}$$

depends on a sign of the constant A. Depending on the sign, three distinct solutions are possible:

$$A > 0: \qquad X(x) = c_1 \mathrm{e}^{\sqrt{A}x} + c_2 \mathrm{e}^{-\sqrt{A}x}; \tag{4.10a}$$

$$A < 0: \qquad X(x) = c_1 \mathrm{e}^{\mathrm{i}\sqrt{|A|}x} + c_2 \mathrm{e}^{-\mathrm{i}\sqrt{|A|}x}; \tag{4.10b}$$

$$A = 0: \qquad X(x) = c_1 x + c_2. \tag{4.10c}$$

Analogous solutions can be find also for $Y(y)$ and $Z(z)$. Any combination of solutions for $X(x)$, $Y(y)$ and $Z(z)$ satisfying the condition (4.8) is then a solution of eq. (4.3).

A solution of a particular problem is determined by boundary conditions and it is usually a linear combination of several distinct solutions of this kind. We are going to find a solution satisfying the boundary conditions (4.4). It will be a linear combination of solutions of three types. Each type is a different combination of solutions (4.10) and is derived either from corners, edges or faces. Therefore, the final solution can be written as

$$\delta K(\mathbf{r}) = \delta K_C(\mathbf{r}) + \sum_{e \in \mathfrak{E}} \delta K_e(\mathbf{r}) + \sum_{f \in \mathfrak{F}} \delta K_f(\mathbf{r}), \tag{4.11a}$$

where $\delta K_C(\mathbf{r})$ is a solution derived from corners, $\delta K_e(\mathbf{r})$ are solutions derived from edges and $\delta K_f(\mathbf{r})$ solutions derived from faces. The set

$$\mathfrak{E} = \{FL, FR, BL, BR, UL, UR, DL, DR, FU, FD, BU, BD\} \tag{4.11b}$$

contains all edges and the set

$$\mathfrak{F} = \{F, B, R, L, U, D\} \tag{4.11c}$$

contains all faces.

4.2.2.1 Corners We start with a partial solution $\delta K_C(\mathbf{r})$ which satisfies the boundary conditions at corners of the cuboid. Only solution which can satisfy any conditions at the corners together with the bond (4.8) has to have a trilinear form

$$\delta K_C(x,y,z) = a + bx + cy + dz + exy + fxz + gyz + hxyz. \quad (4.12)$$

Coefficients are determined from eight boundary conditions:

$$\delta K_C(0,0,0) = a \stackrel{!}{=} -K(0,0,0); \quad (4.13a)$$

$$\delta K_C(L_x,0,0) = a + bL_x \stackrel{!}{=} -K(L_x,0,0)$$
$$\Longrightarrow\ b = -\tfrac{K(L_x,0,0)+a}{L_x}; \quad (4.13b)$$

$$\delta K_C(0,L_y,0) = a + cL_y \stackrel{!}{=} -K(0,L_y,0)$$
$$\Longrightarrow\ c = -\tfrac{K(0,L_y,0)+a}{L_y}; \quad (4.13c)$$

$$\delta K_C(0,0,L_z) = a + dL_z \stackrel{!}{=} -K(0,0,L_z)$$
$$\Longrightarrow\ d = -\tfrac{K(0,0,L_z)+a}{L_z}; \quad (4.13d)$$

$$\delta K_C(L_x,L_y,0) = a + bL_x + cL_y + eL_xL_y \stackrel{!}{=} -K(L_x,L_y,0)$$
$$\Longrightarrow\ e = -\tfrac{K(L_x,L_y,0)+a+bL_x+cL_y}{L_xL_y}; \quad (4.13e)$$

$$\delta K_C(L_x,0,L_z) = a + bL_x + dL_z + fL_xL_z \stackrel{!}{=} -K(L_x,0,L_z)$$
$$\Longrightarrow\ f = -\tfrac{K(L_x,0,L_z)+a+bL_x+dL_z}{L_xL_z}; \quad (4.13f)$$

$$\delta K_C(0,L_y,L_z) = a + cL_y + dL_z + gL_yL_z \stackrel{!}{=} -K(0,L_y,L_z)$$
$$\Longrightarrow\ g = -\tfrac{K(0,L_y,L_z)+a+cL_y+dL_z}{L_yL_z}; \quad (4.13g)$$

$$\delta K_C(L_x,L_y,L_z) = a + bL_x + cL_y + dL_z + eL_xL_y + fL_xL_z +$$
$$+ gL_yL_z + hL_xL_yL_z \stackrel{!}{=} -K(L_x,L_y,L_z)$$
$$\Longrightarrow\ h = -\tfrac{K(L_x,L_y,L_z)+a+bL_x+cL_y+dL_z+eL_xL_y+fL_xL_z+gL_yL_z}{L_xL_yL_z} \quad (4.13h)$$

The trilinear component of the mask has a quite simple structure. Values of this component lie in a range between minimum and maximum value among all values at corners and they grow linearly between two neighbouring corners.

The trilinear component fulfils the boundary conditions at the corners, therefore all other components should be zero there. On the other hand, it is non-zero at edges and faces, thus it changes the boundary conditions for other mask components. The new boundary conditions are

$$\delta K(\mathbf{r}) = -K(\mathbf{r}) - \delta K_C(\mathbf{r}) \tag{4.14a}$$

on walls of the excitation box and

$$\delta K(\mathbf{r}) = -\delta K_C(\mathbf{r}) \tag{4.14b}$$

on the free surface.

4.2.2.2 Edges Now we proceed to mask components derived from edges. We are going to seek solutions which satisfy the boundary conditions on a single edge and are zero on all other edges. There are 12 edges, thus we will need to find 12 such solutions. We will demonstrate how to find such a solution for one edge.

Consider non-zero boundary conditions on the rear-left edge of the cuboid with coordinates $(0, 0, z)$. We denote a corresponding solution $\delta K_{BL}(\mathbf{r})$. We are going to seek it again in form of the product (4.5). It means that the solution will be composed of functions (4.5).

First of all, we realise that all corners have to be zero, and it can be fulfilled only by a solution of type (4.10b), thus

$$Z(z) = c_1 \mathrm{e}^{\mathrm{i}\sqrt{|C|}z} + c_2 \mathrm{e}^{-\mathrm{i}\sqrt{|C|}z}. \tag{4.15a}$$

The zero condition at the upper corner $Z(0) = 0$ yields $c_1 = -c_2$, thus

$$Z(z) = c \sin\left(\sqrt{|C|}z\right). \tag{4.15b}$$

The zero condition at the other corner $Z(L_z) = 0$ can be fulfilled only if $\sqrt{|A|}L_z$ is an integer multiple of π:

$$\sqrt{|C|}L_z = l\pi,\ l \in \mathbb{Z}. \tag{4.15c}$$

Therefore, the solution in z direction is

$$Z_l(z) = c_l \sin\frac{l\pi z}{L_z}. \tag{4.15d}$$

The given solution is associated with a negative constant, thus

$$C = -\left(\frac{l\pi}{L_z}\right)^2. \tag{4.15e}$$

In order to satisfy eq. (4.8), at least one of the constants A and B must be positive. One option is a choice $A = 0$ and $B = -C$. Let's find the solution corresponding to this choice.

A solution associated with a null constant is (4.10c). The zero boundary condition at the front-left edge $(L_x, 0, z)$ provides a boundary condition for function

$$X(x) = a_1 x + a_2, \tag{4.16a}$$

namely $X(L_x) = 0$. It yields $a_2 = -a_1 L_x$, therefore we obtain

$$X(x) = a(x - L_x). \tag{4.16b}$$

A solution associated with a positive constant has form (4.10a). The zero boundary condition at the rear-right edge $(0, L_y, z)$ provides a boundary condition for function

$$Y(y) = b_1 \mathrm{e}^{\sqrt{B}y} + b_2 \mathrm{e}^{-\sqrt{B}y}. \tag{4.17a}$$

The condition is $Y(L_y) = 0$ and it yields $b_2 = -b_1 \mathrm{e}^{2\sqrt{B}L_y}$. Considering eq. (4.15e) and that $B = -C$, we obtain

$$Y_l(y) = b_l \sinh \frac{l\pi(y - L_y)}{L_z}. \tag{4.17b}$$

Putting it all together, we obtain one possible solution

$$\delta K_{1l}(\mathbf{r}) = D_{1l}(x - L_x) \sinh \frac{l\pi(y - L_y)}{L_z} \sin \frac{l\pi z}{L_z}. \tag{4.18a}$$

However, we could have chosen constants A and B vice versa. Such a choice would lead to the solution

$$\delta K_{2l}(\mathbf{r}) = D_{2l}(y - L_y) \sinh \frac{l\pi(x - L_x)}{L_z} \sin \frac{l\pi z}{L_z}. \tag{4.18b}$$

Both solutions are appropriate, thus their arbitrary linear combination

$$\delta K_l(\mathbf{r}) = D'_l \left[d_{1l}(x - L_x) \sinh \frac{l\pi(y - L_y)}{L_z} + \right. \\ \left. + d_{2l}(y - L_y) \sinh \frac{l\pi(x - L_x)}{L_z} \right] \sin \frac{l\pi z}{L_z} \tag{4.18c}$$

is also a solution. There is usually no preferred orientation, thus we choose $d_{1l} = d_{2l} = \frac{1}{2}$. However, it is only one solution. In fact, there is a solution for each l, therefore

$$\delta K_l(\mathbf{r}) = \sum_{l=1}^{\infty} D_l \left[(x - L_x) \sinh \frac{l\pi(y - L_y)}{L_z} + \right. \\ \left. + (y - L_y) \sinh \frac{l\pi(x - L_x)}{L_z} \right] \sin \frac{l\pi z}{L_z}. \tag{4.19}$$

Theoretically, we could use the infinite series, in practice, however, we truncate it at some $L_{\max}$. There are two major reasons for doing it. Firstly, we want to avoid introducing high wave numbers into the kernel. If we smoothed the kernel with a cut-off wave number $k_z^{\mathrm{cut-off}}$, we truncate the series at the highest integer $L_{\max}$

which satisfies $\frac{L_{\max}\pi}{L_z} \leq k_z^{\mathrm{cut-off}}$, thus

$$L_{\max} = \left\lfloor \frac{k_z^{\mathrm{cut-off}} L_z}{\pi} \right\rfloor = \left\lfloor \frac{L_z}{\pi \Lambda_z} \right\rfloor . \tag{4.20}$$

The other reason for truncating the series is that we have a boundary condition only in some discrete points along the edge, thus we are able to determine only a limited number of coefficients D_l. It is impossible to properly determine coefficients corresponding to frequencies higher than a Nyquist frequency, which is given by a sampling rate. Nevertheless, this is much weaker criterion, thus $L_{\max}$ is determined by a maximum permissible wave number.

Speaking about the coefficients D_l, they follow from the nonzero boundary condition. Recall that we require (4.14a) at the rear-left edge. Denote

$$k_{BL}(z) = -K(0,0,z) - \delta K_C(0,0,z) . \tag{4.21}$$

Then the boundary condition yields

$$\delta K_{BL}(0,0,z) = \sum_{l=1}^{L_{\max}} D_l \left(L_x \sinh \frac{l\pi L_y}{L_z} + L_y \sinh \frac{l\pi L_x}{L_z} \right) \cdot \\ \cdot \sin \frac{l\pi z}{L_z} \stackrel{!}{=} k_{BL}(z) . \tag{4.22}$$

If we take a closer look, we realise that eq. (4.22) is a Fourier sine series with a coefficients

$$D_l \left(L_x \sinh \frac{l\pi L_y}{L_z} + L_y \sinh \frac{l\pi L_x}{L_z} \right) = \frac{2}{L_z} \int_0^{L_z} k_{BL}(z) \sin \frac{l\pi z}{L_z} \, \mathrm{d}z , \tag{4.23a}$$

thus

$$D_l = \frac{2}{L_z \left(L_x \sinh \frac{l\pi L_y}{L_z} + L_y \sinh \frac{l\pi L_x}{L_z} \right)} \int_0^{L_z} k_{BL}(z) \sin \frac{l\pi z}{L_z} \, \mathrm{d}z . \tag{4.23b}$$

The found solution has one shortcoming. It has a linear component which decreases much more slowly than hyperbolic term. As a result, the mask can have relatively high values also far from edges which is undesirable. Therefore, we attempt to find another solution for edges.

Recall that in order to satisfy condition (4.8), we had to choose constants A, B such that their sum was equal to (4.8). We met that by setting one constant to zero and the other was negative of C. However, it is not a sole way how to fulfil the condition (4.8). In general, we can set

$$A = \alpha \left(\frac{l\pi}{L_z} \right)^2, \qquad B = \beta \left(\frac{l\pi}{L_z} \right)^2 \qquad \text{with } \alpha + \beta = 1. \tag{4.24}$$

Using the same procedure as in the previous case, we find the corresponding solutions

$$X(x) = a \sinh \frac{\sqrt{\alpha}\, l\pi\,(x - L_x)}{L_z}; \tag{4.25a}$$

$$Y(y) = b \sinh \frac{\sqrt{\beta}\, l\pi\,(y - L_y)}{L_z}. \tag{4.25b}$$

There is usually no preferred direction, thus we choose $\alpha = \beta = \frac{1}{2}$. Then we obtain the solution

$$\delta K_l(\mathbf{r}) = D_l \sinh \frac{l\pi\,(x - L_x)}{\sqrt{2}\, L_z} \sinh \frac{l\pi\,(y - L_y)}{\sqrt{2}\, L_z} \sin \frac{l\pi z}{L_z}. \tag{4.26}$$

Such a solution has an advantage comparing to (4.18c), as it does not contain any linear component, thus it decays more rapidly and it is effectively equal to zero in smaller distance from edges. Furthermore, a hyperbolic sine is only a difference of two exponentials, what is also a shape of the applied smoothing function, thus the kernel remains smooth after application of the mask and its frequency content does not change a lot in sense that it does not acquire any frequency which it has not contained before.

Once again, the final solution is in form of Fourier sine series

$$\delta K_{BL}(\mathbf{r}) = \sum_{l=1}^{\infty} D_l \sinh \frac{l\pi (x - L_x)}{\sqrt{2} L_z} \sinh \frac{l\pi (y - L_y)}{\sqrt{2} L_z} \sin \frac{l\pi z}{L_z} \tag{4.27}$$

and it has to be truncated at some $L_{\max}$ due to the reasons discussed above. The $L_{\max}$ is again determined by (4.20). The Fourier coefficients D_l follow from the same boundary condition, therefore

$$D_l = \frac{2}{L_z \left(\sinh \frac{l\pi L_x}{\sqrt{2} L_z} \sinh \frac{l\pi L_y}{\sqrt{2} L_z} \right)} \int_0^{L_z} k_{BL}(z) \sin \frac{l\pi z}{L_z} \, \mathrm{d}z. \tag{4.28}$$

Finally, we present a summary of the mask components derived from the respective edges:

Front-left edge $(L_x, 0, z)$, $z \in \langle 0; L_z \rangle$

$$\delta K_{FL}(\mathbf{r}) = \sum_{l=1}^{\infty} D_l \sinh \frac{l\pi x}{\sqrt{2} L_z} \sinh \frac{l\pi (y - L_y)}{\sqrt{2} L_z} \sin \frac{l\pi z}{L_z}; \tag{4.29a}$$

$$D_l = -\frac{1}{\sinh \frac{l\pi L_x}{\sqrt{2} L_z} \sinh \frac{l\pi L_y}{\sqrt{2} L_z}} \frac{2}{L_z} \int_0^{L_z} k_{FL}(z) \sin \frac{l\pi z}{L_z} \, \mathrm{d}z; \tag{4.29b}$$

$$k_{FL}(z) = -K(L_x, 0, z) - \delta K_C(L_x, 0, z). \tag{4.29c}$$

Front-right edge (L_x, L_y, z), $z \in \langle 0; L_z \rangle$

$$\delta K_{FR}(\mathbf{r}) = \sum_{l=1}^{\infty} D_l \sinh \frac{l\pi x}{\sqrt{2} L_z} \sinh \frac{l\pi y}{\sqrt{2} L_z} \sin \frac{l\pi z}{L_z}; \tag{4.30a}$$

$$D_l = \frac{1}{\sinh \frac{l\pi L_x}{\sqrt{2} L_z} \sinh \frac{l\pi L_y}{\sqrt{2} L_z}} \frac{2}{L_z} \int_0^{L_z} k_{FR}(z) \sin \frac{l\pi z}{L_z} \, \mathrm{d}z; \tag{4.30b}$$

$$k_{FR}(z) = -K(L_x, L_y, z) - \delta K_C(L_x, L_y, z). \tag{4.30c}$$

Rear-left edge $(0,0,z)$, $z \in \langle 0; L_z \rangle$

$$\delta K_{BL}(\mathbf{r}) = \sum_{l=1}^{\infty} D_l \sinh \frac{l\pi(x-L_x)}{\sqrt{2}L_z} \sinh \frac{l\pi(y-L_y)}{\sqrt{2}L_z} \sin \frac{l\pi z}{L_z}; \quad (4.31a)$$

$$D_l = \frac{1}{\sinh \frac{l\pi L_x}{\sqrt{2}L_z} \sinh \frac{l\pi L_y}{\sqrt{2}L_z}} \frac{2}{L_z} \int_0^{L_z} k_{BL}(z) \sin \frac{l\pi z}{L_z} \,\mathrm{d}z; \quad (4.31b)$$

$$k_{BL}(z) = -K(0,0,z) - \delta K_C(0,0,z). \quad (4.31c)$$

Rear-right edge $(0,L_x,z)$, $z \in \langle 0; L_z \rangle$

$$\delta K_{BR}(\mathbf{r}) = \sum_{l=1}^{\infty} D_l \sinh \frac{l\pi(x-L_x)}{\sqrt{2}L_z} \sinh \frac{l\pi y}{\sqrt{2}L_z} \sin \frac{l\pi z}{L_z}; \quad (4.32a)$$

$$D_l = -\frac{1}{\sinh \frac{l\pi L_x}{\sqrt{2}L_z} \sinh \frac{l\pi L_y}{\sqrt{2}L_z}} \frac{2}{L_z} \int_0^{L_z} k_{BR}(z) \sin \frac{l\pi z}{L_z} \,\mathrm{d}z; \quad (4.32b)$$

$$k_{BR}(z) = -K(0,L_x,z) - \delta K_C(0,L_x,z). \quad (4.32c)$$

Top-left edge $(x,0,0)$, $x \in \langle 0; L_x \rangle$

$$\delta K_{UL}(\mathbf{r}) = \sum_{m=1}^{\infty} D_m \sinh \frac{m\pi(y-L_y)}{\sqrt{2}L_x} \sinh \frac{m\pi(z-L_z)}{\sqrt{2}L_x} \sin \frac{m\pi x}{L_x}; \quad (4.33a)$$

$$D_m = \frac{1}{\sinh \frac{m\pi L_y}{\sqrt{2}L_x} \sinh \frac{m\pi L_z}{\sqrt{2}L_x}} \frac{2}{L_x} \int_0^{L_x} k_{UL}(x) \sin \frac{m\pi x}{L_x} \,\mathrm{d}x; \quad (4.33b)$$

$$k_{UL}(x) = -K(x,0,0) - \delta K_C(x,0,0). \quad (4.33c)$$

Top-right edge $(x,L_y,0)$, $x \in \langle 0; L_x \rangle$

$$\delta K_{UR}(\mathbf{r}) = \sum_{m=1}^{\infty} D_m \sinh \frac{m\pi y}{\sqrt{2}L_x} \sinh \frac{m\pi(z-L_z)}{\sqrt{2}L_x} \sin \frac{m\pi x}{L_x}; \quad (4.34a)$$

$$D_m = -\frac{1}{\sinh \frac{m\pi L_y}{\sqrt{2}L_x} \sinh \frac{m\pi L_z}{\sqrt{2}L_x}} \frac{2}{L_x} \int_0^{L_x} k_{UR}(x) \sin \frac{m\pi x}{L_x} \,\mathrm{d}x; \quad (4.34b)$$

$$k_{UR}(x) = -K(x,L_y,0) - \delta K_C(x,L_y,0). \quad (4.34c)$$

Bottom-left edge $(x, 0, L_z)$, $x \in \langle 0; L_x \rangle$

$$\delta K_{DL}(\mathbf{r}) = \sum_{m=1}^{\infty} D_m \sinh \frac{m\pi (y - L_y)}{\sqrt{2} L_x} \sinh \frac{m\pi z}{\sqrt{2} L_x} \sin \frac{m\pi x}{L_x}; \quad (4.35a)$$

$$D_m = -\frac{1}{\sinh \frac{m\pi L_y}{\sqrt{2} L_x} \sinh \frac{m\pi L_z}{\sqrt{2} L_x}} \frac{2}{L_x} \int_0^{L_x} k_{DL}(x) \sin \frac{m\pi x}{L_x} \, \mathrm{d}x; \quad (4.35b)$$

$$k_{DL}(x) = -K(x, 0, L_z) - \delta K_C(x, 0, L_z). \quad (4.35c)$$

Bottom-right edge (x, L_y, L_z), $x \in \langle 0; L_x \rangle$

$$\delta K_{DR}(\mathbf{r}) = \sum_{m=1}^{\infty} D_m \sinh \frac{m\pi y}{\sqrt{2} L_x} \sinh \frac{m\pi z}{\sqrt{2} L_x} \sin \frac{m\pi x}{L_x}; \quad (4.36a)$$

$$D_m = \frac{1}{\sinh \frac{m\pi L_y}{\sqrt{2} L_x} \sinh \frac{m\pi L_z}{\sqrt{2} L_x}} \frac{2}{L_x} \int_0^{L_x} k_{DR}(x) \sin \frac{m\pi x}{L_x} \, \mathrm{d}x; \quad (4.36b)$$

$$k_{DR}(x) = -K(x, L_y, L_z) - \delta K_C(x, L_y, L_z). \quad (4.36c)$$

Front-top edge $(L_x, y, 0)$, $y \in \langle 0; L_y \rangle$

$$\delta K_{FU}(\mathbf{r}) = \sum_{n=1}^{\infty} D_n \sinh \frac{n\pi x}{\sqrt{2} L_y} \sinh \frac{n\pi (z - L_z)}{\sqrt{2} L_y} \sin \frac{n\pi y}{L_y}; \quad (4.37a)$$

$$D_n = -\frac{1}{\sinh \frac{n\pi L_x}{\sqrt{2} L_y} \sinh \frac{n\pi L_z}{\sqrt{2} L_y}} \frac{2}{L_y} \int_0^{L_y} k_{FU}(y) \sin \frac{n\pi y}{L_y} \, \mathrm{d}y; \quad (4.37b)$$

$$k_{FU}(y) = -K(L_x, y, 0) - \delta K_C(L_x, y, 0). \quad (4.37c)$$

Front-bottom edge (L_x, y, L_z), $y \in \langle 0; L_y \rangle$

$$\delta K_{FD}(\mathbf{r}) = \sum_{n=1}^{\infty} D_n \sinh \frac{n\pi x}{\sqrt{2} L_y} \sinh \frac{n\pi z}{\sqrt{2} L_y} \sin \frac{n\pi y}{L_y}; \quad (4.38a)$$

$$D_n = \frac{1}{\sinh \frac{n\pi L_x}{\sqrt{2} L_y} \sinh \frac{n\pi L_z}{\sqrt{2} L_y}} \frac{2}{L_y} \int_0^{L_y} k_{FD}(y) \sin \frac{n\pi y}{L_y} \, \mathrm{d}y; \quad (4.38b)$$

$$k_{FD}(y) = -K(L_x, y, L_z) - \delta K_C(L_x, y, L_z). \quad (4.38c)$$

Rear-top edge $(0, y, 0)$, $y \in \langle 0; L_y \rangle$

$$\delta K_{BU}(\mathbf{r}) = \sum_{n=1}^{\infty} D_n \sinh \frac{n\pi (x - L_x)}{\sqrt{2} L_y} \sinh \frac{n\pi (z - L_z)}{\sqrt{2} L_y} \sin \frac{n\pi y}{L_y}; \tag{4.39a}$$

$$D_n = \frac{1}{\sinh \frac{n\pi L_x}{\sqrt{2} L_y} \sinh \frac{n\pi L_z}{\sqrt{2} L_y}} \frac{2}{L_y} \int_0^{L_y} k_{BU}(y) \sin \frac{n\pi y}{L_y} \, \mathrm{d}y; \tag{4.39b}$$

$$k_{BU}(y) = -K(0, y, 0) - \delta K_C(0, y, 0). \tag{4.39c}$$

Rear-bottom edge $(0, y, L_z)$, $y \in \langle 0; L_y \rangle$

$$\delta K_{BD}(\mathbf{r}) = \sum_{n=1}^{\infty} D_n \sinh \frac{n\pi (x - L_x)}{\sqrt{2} L_y} \sinh \frac{n\pi z}{\sqrt{2} L_y} \sin \frac{n\pi y}{L_y}; \tag{4.40a}$$

$$D_n = -\frac{1}{\sinh \frac{n\pi L_x}{\sqrt{2} L_y} \sinh \frac{n\pi L_z}{\sqrt{2} L_y}} \frac{2}{L_y} \int_0^{L_y} k_{BD}(y) \sin \frac{n\pi y}{L_y} \, \mathrm{d}y; \tag{4.40b}$$

$$k_{BD}(y) = -K(0, y, L_z) - \delta K_C(0, y, L_z). \tag{4.40c}$$

4.2.2.3 Faces Before we will find mask components derived from the faces, we need to update boundary conditions, as those has been affected by the mask components derived from the edges. The new boundary conditions are

$$\delta K(\mathbf{r}) = -K(\mathbf{r}) - \delta K_C(\mathbf{r}) - \sum_{e \in \mathfrak{E}} \delta K_e(\mathbf{r}) \tag{4.41a}$$

at walls of the excitation box and

$$\delta K(\mathbf{r}) = -\delta K_C(\mathbf{r}) - \sum_{e \in \mathfrak{E}} \delta K_e(\mathbf{r}) \tag{4.41b}$$

at the free surface.

Now we can proceed to looking for the mask components derived from the faces. Let's just remind that we have already satisfied boundary conditions at corners and edges, thus the solution for faces has to be zero there. We will use the same procedure as

in case of edges. We will build the solution from 6 solutions – one for each face. Those individual solutions have a non-zero boundary condition at just one face and zero conditions at the others.

We are going to demonstrate the procedure of finding the solution for one face on example of the rear face $(0, y, z)$. We seek the solution $\delta K_B(\mathbf{r})$ in form (4.5) again, which means it consists of functions (4.10).

The solution for the face has to be zero at the adjacent edges. It can by satisfied only by functions of the type (4.10b) in y and z directions, therefore corresponding coefficients has to be negative. Hence we can write them in form $B = -q^2$ and $C = -r^2$.

Firstly, we are going to find the solution in y direction. We know, it has to have the form

$$Y(y) = b_1 \mathrm{e}^{\mathrm{i}qy} + b_2 \mathrm{e}^{-\mathrm{i}qy}. \tag{4.42a}$$

The zero boundary conditions at the left and right adjacent edges yield $Y(0) = 0$ and $Y(L_y) = 0$ respectively. The former implies $b_2 = -b_1$, thus

$$Y(y) = b \sin(qy), \tag{4.42b}$$

the later sets restriction on the coefficient

$$qL_y = n\pi,\ n \in \mathbb{Z}. \tag{4.42c}$$

Therefore the sought solution in y direction is

$$Y_n(y) = b \sin \frac{n\pi y}{L_y} \tag{4.42d}$$

and the corresponding constant is

$$B = -\left(\frac{n\pi}{L_y}\right)^2 = -q_n^2. \tag{4.42e}$$

Analogously, we can find a solution in z direction. It has the same form and the upper and bottom adjacent edges provide similar boundary conditions $Z(0) = 0$ and $Z(L_z) = 0$, therefore we obtain

$$Z_l(z) = c \sin \frac{l\pi z}{L_z} \tag{4.43a}$$

with the corresponding constant

$$C = -\left(\frac{l\pi}{L_z}\right)^2 = -r_l^2. \tag{4.43b}$$

Finally, we have to find the solution in x direction. The condition (4.8) implies that the constant A has to be positive

$$A = -B - C = \left(\frac{n\pi}{L_y}\right)^2 + \left(\frac{l\pi}{L_z}\right)^2 = p_{nl}^2, \tag{4.44a}$$

therefore the sought solution is of form (4.10a)

$$X_{nl}(x) = a_1 \mathrm{e}^{p_{nl}x} + a_2 \mathrm{e}^{-p_{nl}x}. \tag{4.44b}$$

The zero boundary condition at the opposite face yields $X(L_x) = 0$, thus $a_2 = -a_1 \mathrm{e}^{2p_{nl}L_x}$, hence the solution is

$$X_{nl}(x) = a \sinh\left[p_{nl}(x - L_x)\right]. \tag{4.44c}$$

Putting it all together, we obtain

$$\delta K_{nl}(\mathbf{r}) = D_{nl} \sinh\left[p_{nl}(x - L_x)\right] \sin \frac{n\pi y}{L_y} \sin \frac{l\pi z}{L_z}. \tag{4.45}$$

Actually, we have found an infinite number of solutions and their arbitrary linear combination

$$\delta K_B(\mathbf{r}) = \sum_{n=1}^{\infty} \sum_{l=1}^{\infty} D_{nl} \sinh\left[p_{nl}(x - L_x)\right] \sin \frac{n\pi y}{L_y} \sin \frac{l\pi z}{L_z}. \tag{4.46}$$

is also the solution. We obtained a two-dimensional Fourier sine series. In practice, we need to truncate it at some $N_{\max}$ and $L_{\max}$ because of the same reasons as for edges.

The coefficients D_{nl} are determined by the sole non-zero boundary condition. Denote

$$k_B(y,z) = -K(0,y,z) - \delta K_C(0,y,z) - \sum_{e\in\mathfrak{E}} \delta K_e(0,y,z), \quad (4.47)$$

where $\mathfrak{E}$ contains (4.11b). Then it yields

$$\delta K_B(0,y,z) = -\sum_{n=1}^{\infty}\sum_{l=1}^{\infty} D_{nl}\sinh(p_{nl}L_x)\sin\frac{n\pi y}{L_y}\sin\frac{l\pi z}{L_z} \stackrel{!}{=} k_B(y,z). \quad (4.48)$$

Consequently, it follows that

$$D_{nl} = \frac{4}{L_y L_z \sinh(p_{nl}L_x)}\int_0^{L_y}\left[\int_0^{L_z} k_B(y,z)\sin\frac{l\pi z}{L_z}\,\mathrm{d}z\right]\sin\frac{n\pi y}{L_y}\,\mathrm{d}y. \quad (4.49)$$

Finally, we present a summary of the mask components derived from respective faces:

Front face (L_x,y,z), $y\in\langle 0;L_y\rangle$, $z\in\langle 0;L_z\rangle$

$$\delta K_F(\mathbf{r}) = \sum_{n=1}^{\infty}\sum_{l=1}^{\infty} D_{nl}\sinh(p_{nl}x)\sin\frac{n\pi y}{L_y}\sin\frac{l\pi z}{L_z}; \quad (4.50a)$$

$$p_{nl} = \pi\sqrt{\left(\frac{n}{L_y}\right)^2 + \left(\frac{l}{L_z}\right)^2}; \quad (4.50b)$$

$$D_{nl} = \frac{1}{\sinh(p_{nl}L_x)}\frac{4}{L_y L_z}\int_0^{L_y}\left[\int_0^{L_z} k_F(y,z)\sin\frac{l\pi z}{L_z}\,\mathrm{d}z\right]\sin\frac{n\pi y}{L_y}\,\mathrm{d}y; \quad (4.50c)$$

$$k_F(y,z) = -K(L_x,y,z) - \delta K_C(L_x,y,z) - \sum_{e\in\mathfrak{E}}\delta K_e(L_x,y,z). \quad (4.50d)$$

Rear face $(0, y, z)$, $y \in \langle 0; L_y \rangle$, $z \in \langle 0; L_z \rangle$

$$\delta K_B(\mathbf{r}) = \sum_{n=1}^{\infty} \sum_{l=1}^{\infty} D_{nl} \sinh\left[p_{nl}(x - L_x)\right] \sin\frac{n\pi y}{L_y} \sin\frac{l\pi z}{L_z}; \quad (4.51a)$$

$$p_{nl} = \pi \sqrt{\left(\frac{n}{L_y}\right)^2 + \left(\frac{l}{L_z}\right)^2}; \quad (4.51b)$$

$$D_{nl} = -\frac{1}{\sinh(p_{nl} L_x)} \frac{4}{L_y L_z} \int_0^{L_y} \left[\int_0^{L_z} k_B(y, z) \sin\frac{l\pi z}{L_z} \,\mathrm{d}z \right] \sin\frac{n\pi y}{L_y} \,\mathrm{d}y; \quad (4.51c)$$

$$k_B(y, z) = -K(0, y, z) - \delta K_C(0, y, z) - \sum_{e \in \mathfrak{E}} \delta K_e(0, y, z). \quad (4.51d)$$

Top face $(x, y, 0)$, $x \in \langle 0; L_x \rangle$, $y \in \langle 0; L_y \rangle$

$$\delta K_U(\mathbf{r}) = \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} D_{mn} \sinh\left[r_{mn}(z - L_z)\right] \sin\frac{m\pi x}{L_x} \sin\frac{n\pi y}{L_y}; \quad (4.52a)$$

$$r_{mn} = \pi \sqrt{\left(\frac{m}{L_x}\right)^2 + \left(\frac{n}{L_y}\right)^2}; \quad (4.52b)$$

$$D_{mn} = -\frac{1}{\sinh(r_{mn} L_z)} \frac{4}{L_x L_y} \int_0^{L_x} \left[\int_0^{L_y} k_U(x, y) \sin\frac{n\pi y}{L_y} \,\mathrm{d}y \right] \sin\frac{m\pi x}{L_x} \,\mathrm{d}x; \quad (4.52c)$$

$$k_U(x, y) = -\delta K_C(x, y, 0) - \sum_{e \in \mathfrak{E}} \delta K_e(x, y, 0). \quad (4.52d)$$

Left face $(x, 0, z)$, $x \in \langle 0; L_x \rangle$, $z \in \langle 0; L_z \rangle$

$$\delta K_L(\mathbf{r}) = \sum_{m=1}^{\infty} \sum_{l=1}^{\infty} D_{ml} \sinh\left[q_{ml}(y - L_y)\right] \sin\frac{m\pi x}{L_x} \sin\frac{l\pi z}{L_z}; \quad (4.53a)$$

$$q_{ml} = \pi \sqrt{\left(\frac{m}{L_x}\right)^2 + \left(\frac{l}{L_z}\right)^2}; \quad (4.53b)$$

$$D_{ml} = -\frac{1}{\sinh(q_{ml} L_y)} \frac{4}{L_x L_z} \int_0^{L_x} \left[\int_0^{L_z} k_L(x, z) \sin\frac{l\pi z}{L_z} \,\mathrm{d}z \right] \sin\frac{m\pi x}{L_x} \,\mathrm{d}x; \quad (4.53c)$$

$$k_L(x, z) = -K(x, 0, z) - \delta K_C(x, 0, z) - \sum_{e \in \mathfrak{E}} \delta K_e(x, 0, z). \quad (4.53d)$$

Bottom face (x, y, L_z), $x \in \langle 0; L_x \rangle$, $y \in \langle 0; L_y \rangle$

$$\delta K_D(\mathbf{r}) = \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} D_{mn} \sinh(r_{mn} z) \sin \frac{m\pi x}{L_x} \sin \frac{n\pi y}{L_y}; \tag{4.54a}$$

$$r_{mn} = \pi \sqrt{\left(\frac{m}{L_x}\right)^2 + \left(\frac{n}{L_y}\right)^2}; \tag{4.54b}$$

$$D_{mn} = \frac{1}{\sinh(r_{mn} L_z)} \frac{4}{L_x L_y} \int_0^{L_x} \left[\int_0^{L_y} k_D(x, y) \sin \frac{n\pi y}{L_y} \, \mathrm{d}y \right] \sin \frac{m\pi x}{L_x} \, \mathrm{d}x; \tag{4.54c}$$

$$k_D(x, y) = -K(x, y, L_z) - \delta K_C(x, y, L_z) - \sum_{e \in \mathfrak{E}} \delta K_e(x, y, L_z). \tag{4.54d}$$

Right face (x, L_y, z), $x \in \langle 0; L_x \rangle$, $z \in \langle 0; L_z \rangle$

$$\delta K_R(\mathbf{r}) = \sum_{m=1}^{\infty} \sum_{l=1}^{\infty} D_{ml} \sinh(q_{ml} y) \sin \frac{m\pi x}{L_x} \sin \frac{l\pi z}{L_z}; \tag{4.55a}$$

$$q_{ml} = \pi \sqrt{\left(\frac{m}{L_x}\right)^2 + \left(\frac{l}{L_z}\right)^2}; \tag{4.55b}$$

$$D_{ml} = \frac{1}{\sinh(q_{ml} L_y)} \frac{4}{L_x L_z} \int_0^{L_x} \left[\int_0^{L_z} k_R(x, z) \sin \frac{l\pi z}{L_z} \, \mathrm{d}z \right] \sin \frac{m\pi x}{L_x} \, \mathrm{d}x; \tag{4.55c}$$

$$k_R(x, z) = -K(x, L_y, z) - \delta K_C(x, L_y, z) - \sum_{e \in \mathfrak{E}} \delta K_e(x, L_y, z). \tag{4.55d}$$

4.2.3 Problem with a free surface

We have found a solution of the Laplace equation (4.3) in the form of a sum of Fourier sine series. However, we do not use the entire series, instead we truncate them after several terms. It means that the solution does not satisfy the boundary conditions (4.4) exactly, but it has rather an oscillatory character.

If a function to be expanded does not contain any discontinuities, the Fourier series converges relatively fast, thus usually a

few first terms is enough. As we calculate the mask after the kernel has been smoothed, this requirement is met.

However, one must be careful, because we use the Fourier sine series, thus we expand an odd extension of the function. Therefore, the discontinuities appear if the end values of the function are non-zero. And it is often the case.

However, neither this causes a problem. Non-zero values at the corners are nulled by an application of the trilinear component of the mask, which naturally does not suffer from this problem. The trilinear component is smooth, so it lowers the values also in the vicinity of the corners. Consequently, the end points of the edges are nulled and the odd extensions of the edges do not contain discontinuities, thus the Fourier series for edges converge relatively fast and hence do not exhibit the oscillatory behaviour. Therefore, the boundary conditions on edges are satisfied quite well, which means that the boundary conditions for faces are zero at the adjacent edges, thus the series derived from the faces can be truncated as well. So where is the problem?

We must realise that we have a discontinuous boundary conditions at the upper edges. It comes from the fact that we require a no-gain zone of the mask at the walls of the excitation box, but a full-gain zone at the free surface. It would not be a problem if we dealt with a continuous problem. There are methods how to address it (e.g. Braverman et al., 1998). However, in the discrete case, we must decide which condition prevails on the edge where the discontinuity appears.

We prefer the no-gain condition at the edge. Therefore, the zero boundary condition at the free surface means that the mask component derived from the top face has to revert the changes in the kernel at the free surface caused by other mask components. However, it means that the values at the edges of the upper face are non-zero, thus there are discontinuities in odd extensions of

the function to be expanded and hence the Fourier series at the free surface converges very slowly.

Consequently, the truncated series exhibits an oscillatory behaviour. The oscillations diminish when more terms of the series are preserved which is the case at the lower scales. However, it is impossible to get rid of them completely. When we add another term to the truncated series, the oscillations lower slightly and shift towards the discontinuity but they never disappear. It is so-called Gibbs phenomenon. If we wanted to suppress the oscillations, so that they were shifted to the edge where they were not observable, we would need to go beyond Nyquist frequency.

We will be able to observe the oscillations at the numerical examples. Fortunately their amplitude is relatively small and it decreases with the depth rapidly. In any case, we should be very careful when applying the mask and we should always check them out.

4.2.4 Numerical implementation and time complexity

In this section, we briefly describe our numerical implementation of the algorithm for solving of the Laplace equation. It will allow us to estimate a time complexity of the mask computation.

First of all, we determine the number of terms in truncated Fourier series. Let's consider dimensions of the computational domain $N_x \times N_y \times N_z$ and smoothing intensities Λ_x, Λ_y and Λ_z. Then according to eq. (4.20), the corresponding numbers of terms are

$$M_{\max} = \left\lfloor \frac{(N_x - 1)\, h}{\pi \Lambda_x} \right\rfloor, \quad N_{\max} = \left\lfloor \frac{(N_y - 1)\, h}{\pi \Lambda_y} \right\rfloor, \quad L_{\max} = \left\lfloor \frac{(N_z - 1)\, h}{\pi \Lambda_z} \right\rfloor. \tag{4.56}$$

Secondly, we calculate the coefficients of the trilinear part of the mask. We use eq. (4.13) with one slight modification – in-

stead of lengths L, we use numbers of grid points in respective directions minus one $L \leftrightarrow (N-1)$. It will allow us to calculate mask values based on grid indices rather than on coordinates. We will use this interchange from now further on without notifying.

Beside that, we allocate an additional two-dimensional array, where we will store the mask values at the free surface. It will allow us to meet the zero boundary condition at the free surface later.

Consequently, we update the kernel with the trilinear mask components and we add values on the surface to that auxiliary array. A calculation of the coefficients takes a negligible amount of computational time, thus a time complexity of this step is determined by updating of the kernel. We need to update $N_x \cdot N_y \cdot N_z$ values, thus the time complexity of this step is $\mathcal{O}(N_x N_y N_z)$. Storing the mask values at the surface requires only $N_x \cdot N_y$ steps, thus it is only of $\mathcal{O}(N_x N_y)$.

Thirdly, we calculate the Fourier coefficients of the mask components derived from edges. As the trilinear part of the mask has been already added to the kernel, we can use the kernel values along edges as the boundary condition directly.

We use the Simpson rule for computation of the coefficients if a number of grid points along the respective edge is even, otherwise we use the trapezoidal rule. We do not need to use the Fast Fourier transform, as we are interested only in a few first coefficients. The FFT could bring some computational time reduction only on the lowest scales, on which we need to calculate more Fourier coefficients. However, the computational times are not too large, thus the potential reduction would not be significant in absolute values. A time complexity of the calculation of Fourier coefficients is hence $\mathcal{O}(N_x M_{\max})$ for an edge oriented in x direction and similar for other edges.

Consequently, we update the kernel with the mask components derived from edges and add values at the free surface to the auxiliary two-dimensional array. A time complexity of updating of the kernel is $12\mathcal{O}\left(N_x N_y N_z M_{\max}\right)$ for an edge oriented in x direction and $12\mathcal{O}\left(N_x N_y M_{\max}\right)$ of adding the values at the free surface to the auxiliary array. It is like that because there are $N_x \cdot N_y \cdot N_z$ points and for each of them, we need to calculate $M_{\max}$ Fourier terms.

Finally, we calculate Fourier coefficients of the mask components derived from faces. Once again, we can use the kernel values at the faces as a boundary condition, as it has been already updated by the mask components derived from corners and edges. The boundary condition for the top face is stored in the auxiliary array.

We use the two-dimensional Simpson rule if dimensions of the face are even, otherwise we use the two-dimensional trapezoidal rule. Neither here we use the Fast Fourier transform. For instance, a time complexity of the computation of the coefficients for the top face is $\mathcal{O}\left(M_{\max} N_{\max} N_X N_Y\right)$. Comparing it to the FFT, its time complexity is $\mathcal{O}\left(N_X N_Y \log\left(N_X N_Y\right)\right)$. Considering $N_X \cdot N_Y \sim 10^6$, a computational time of the FFT is $6c \cdot 10^6$. On the other hand $M_{\max} \cdot N_{\max} \sim 10^1$, thus it is not much more computational demanding than the FFT, except for the case of the lowest scales at which $M_{\max} \cdot N_{\max}$ can be of an order or two higher. Anyway, it is still less computationally demanding than updating the kernel.

Consequently, we update the kernel with the mask components derived from faces which is of $6\mathcal{O}\left(N_x N_y N_z M_{\max} N_{\max}\right)$ for a horizontally-oriented face. It is the most computationally demanding step of calculation and application of the mask. We do not have to add values at the free surface to that auxiliary array, as those values are per definition zero.

Let's note, that we have not calculated a spatial representation of the mask. Instead we represent it using the Fourier coefficients and the coefficients of the trilinear part, hence we know its (truncated) spectrum. Such a representation is very compact. We can store it with very low memory demands and reconstruct whenever we need it.

4.3 Numerical tests

4.3.1 Coefficients of mask components

First of all, we test out whether the Laplace equation solver solves the eq. (4.3) correctly. We filled a three-dimensional cubic array ($N = 201$) with a solution of the Laplace equation satisfying the boundary conditions (4.4). If the solver works correctly, it should return exactly the prescribed solution, hence the mask should effectively null the entire array.

Let's note that we implemented the mask as a function that has to be subtracted from the kernel, not added to it. It means that we changed the sign in eq. (4.2). Consequently, we had to change the sign in the boundary conditions (4.4a) from $-K(\mathbf{r})$ to $+K(\mathbf{r})$. Therefore, the mask should be same as the input array. If we implemented it the other way around, we should obtain exactly opposite solution.

The prescribed solution is

$$\delta K^{i\,j\,k} = \delta K_c^{i\,j\,k} + \delta K_{DL;1}^{i\,j\,k} + \delta K_{F;1,1}^{i\,j\,k}, \tag{4.57a}$$

$$\delta K_c^{i\,j\,k} = \frac{i \cdot j \cdot k}{(N_x - 1)(N_y - 1)(N_z - 1)}, \tag{4.57b}$$

$$\delta K_{DL;1}^{i\,j\,k} = \sinh \frac{\pi (j - N_y + 1)}{\sqrt{2}(N_x - 1)} \cdot \sinh \frac{\pi k}{\sqrt{2}(N_x - 1)} \cdot \sin \frac{\pi i}{N_x - 1}, \tag{4.57c}$$

$$\delta K_{F;1,1}^{i\,j\,k} = \sinh \left[\pi \sqrt{\frac{1}{(N_y - 1)^2} + \frac{1}{(N_z - 1)^2}}\, i \right] \cdot \sin \frac{\pi j}{N_y - 1} \cdot \sin \frac{\pi k}{N_z - 1}. \tag{4.57d}$$

It consists of the component derived from corners with a coefficient $h = {}^{1}/_{200^3} = 1.25 \cdot 10^{-7}$, a fundamental mode of the component derived from the bottom-left edge with a coefficient $D_1 = 1$ and a fundamental mode of the component derived from the front face with a coefficient $D_{1,1} = 1$. All other coefficients are zero. Thus we expect to obtain exactly these coefficients.

The characteristic length was $\Lambda = 20$ with a unit grid step, thus according to eq. (4.56), a number of terms in the Fourier series was 3. The calculated coefficients are presented in Tab. 4.1. They follow theoretical predictions very well. Discrepancies are caused by rounding within float precision.

4.3.2 Application of mask on smoothed Gaussian signal

As we have verified that our Laplace equation solver calculates mask correctly, we can perform numerical tests. We will find the mask for the smoothed array from the subsection 3.2.2.3 and will apply it. We use an array smoothed by a smoothing function with a smoothing intensity ${}^{\Lambda}/_{h} = 20$ grid points.

The results are presented on Fig. 4.1. The Fig. 4.1a shows the array before an application of the mask. We can observe non-zero values at the surface of domain. On the other hand, the Fig. 4.1b shows the array after the application of the mask. The non-zero values have been removed from the side faces, whilst they are still present at the free surface.

However, notice that the non-zero values have not been removed completely, but some lower-amplitude oscillations are still present. It is a consequence of truncating of the Fourier series. We discussed this phenomenon in the section 4.2.3. Another proofs are the dark blue areas around the positive anomalies which indicate negative values that were not present before the application of the mask.

h	$1.250000 \cdot 10^{-7}$

(a) Coefficients of a mask component derived from corners

harmonic	1st	2nd	3rd
FL edge	$-3.710221 \cdot 10^{-8}$	$2.841876 \cdot 10^{-10}$	$-2.374120 \cdot 10^{-12}$
FR edge	$-1.789501 \cdot 10^{-7}$	$-5.708514 \cdot 10^{-13}$	$-1.872145 \cdot 10^{-15}$
DL edge	$9.999996 \cdot 10^{-1}$	$-2.049828 \cdot 10^{-9}$	$-1.128682 \cdot 10^{-12}$
DR edge	$6.059649 \cdot 10^{-11}$	$-5.484305 \cdot 10^{-13}$	$4.110159 \cdot 10^{-15}$
FD edge	$-1.418135 \cdot 10^{-7}$	$2.830153 \cdot 10^{-10}$	$2.387933 \cdot 10^{-12}$

(b) Coefficients of mask components derived from edges

j \ k	1	2	3
1	$9.999940 \cdot 10^{-1}$	$-7.564761 \cdot 10^{-9}$	$1.194984 \cdot 10^{-10}$
2	$-1.473335 \cdot 10^{-8}$	$4.122872 \cdot 10^{-12}$	$-5.112367 \cdot 10^{-14}$
3	$-2.126567 \cdot 10^{-9}$	$-7.795322 \cdot 10^{-13}$	$-8.293764 \cdot 10^{-13}$

(c) Coefficients of mask components derived from the front face

i \ k	1	2	3
1	$3.238894 \cdot 10^{-8}$	$-1.328186 \cdot 10^{-9}$	$6.559744 \cdot 10^{-11}$
2	$-2.505720 \cdot 10^{-10}$	$-7.385819 \cdot 10^{-11}$	$7.807869 \cdot 10^{-12}$
3	$5.155900 \cdot 10^{-11}$	$-8.543731 \cdot 10^{-12}$	$7.195901 \cdot 10^{-13}$

(d) Coefficients of mask components derived from the left face

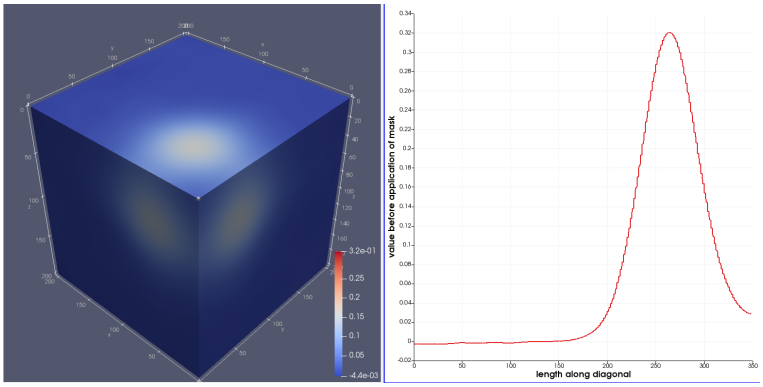
i \ k	1	2	3
1	$1.855614 \cdot 10^{-8}$	$8.013083 \cdot 10^{-14}$	$1.741980 \cdot 10^{-14}$
2	$-4.677217 \cdot 10^{-10}$	$-3.649220 \cdot 10^{-14}$	$-2.238311 \cdot 10^{-15}$
3	$9.890108 \cdot 10^{-12}$	$5.190340 \cdot 10^{-17}$	$3.019191 \cdot 10^{-16}$

(e) Coefficients of mask components derived from the right face

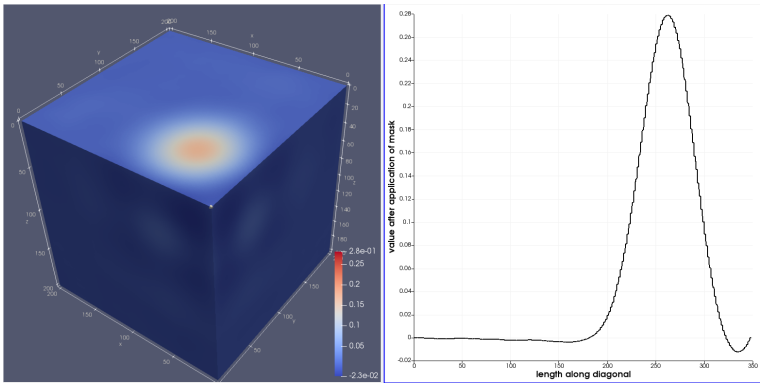
i \ j	1	2	3
1	$-1.582902 \cdot 10^{-8}$	$-1.465725 \cdot 10^{-9}$	$-7.367011 \cdot 10^{-11}$
2	$-2.631501 \cdot 10^{-10}$	$-6.473791 \cdot 10^{-11}$	$-7.722384 \cdot 10^{-12}$
3	$-4.161863 \cdot 10^{-11}$	$-9.131705 \cdot 10^{-12}$	$-8.727038 \cdot 10^{-13}$

(f) Coefficients of mask components derived from the bottom face

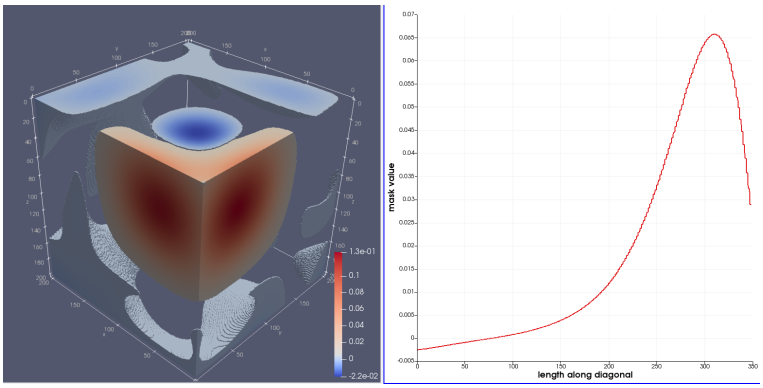
Table 4.1: Calculated coefficients of the mask. Zero coefficients are omitted. Green colour highlights the coefficients which should be non-zero



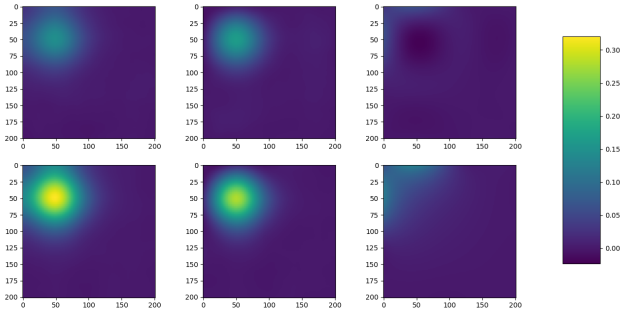
(a) A smoothed signal before an application of the mask



(b) The signal after the application of the mask



(c) Mask. Blue regions represent negative mask values, red regions represent mask values greater than 0.02



(d) A free surface and a horizontal slice before and after application of the mask and the mask itself

Figure 4.1: Application of mask on a smoothed Gaussian signal

Furthermore, the positive amplitude at the free surface has been enhanced by the application of the mask. It is another proof of this phenomenon. The reason for it is that an individual mask components have non-zero values at the surface. The mask component derived from the top face attempts to restore it, so that the zero boundary condition at the free surface is fulfilled. However, the series derived from the free surface has been truncated, thus it exhibits the oscillatory behaviour. What is more, the odd extension of the boundary condition has a discontinuity, thus the series converges slowly.

The changes caused by mask are best-illustrated by the mask itself (see Fig. 4.1c). The blue regions are the regions of negative mask values, thus the changes are positive there. A red colour indicates the regions where the negative change is greater than 0.02. The areas with mask values in between are transparent.

Fig. 4.1d shows a free surface and a horizontal slice through the Gaussian peak. The first column depicts the array before the application of the mask, the second column after the application and the third one illustrates the mask. We can see already mentioned negative mask anomaly surrounded by positive one at the free surface. We can also observe that the mask nulls the values only near the surface and modifies the interior only slightly.

4.3.3 Demonstration of Gibbs phenomenon

In the the section 4.2.3, we indicated that there is a problem at the free surface originated from a discontinuity in the boundary conditions. In this section, we are going to demonstrate how it is manifested in practice.

We expect the strongest Gibbs phenomenon if there is a signal close to any upper edge. Therefore, we use same Gaussian signal overlaid by noise as in the section 4.3.2, but centred at an upper corner. We added another Gaussian peak centred at the opposite

corner, so that we can compare the effect of the discontinuity in the boundary conditions with boundary conditions without any discontinuity.

First of all, we smoothed the signal with three different isotropic smoothing intensities $\frac{\Lambda}{h}$ – 40, 20 and 10. We used a cubic computational domain of size $N = 201$. Therefore, according to eq. (4.56), we truncated the Fourier series after 1st, 3rd and 6th term respectively.

Fig. 4.2 illustrates the effect of the mask at the highest scale. The Fourier series were truncated after the first term.

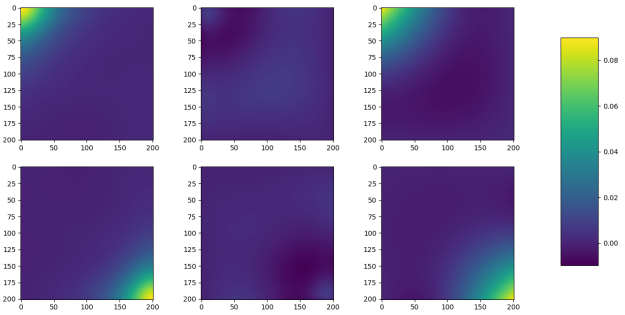
Fig. 4.2b shows the original smoothed signal before application of the mask. In the left picture, only values higher than 0.003 are depicted. We see that the signal is very simple.

Fig. 4.2c shows the signal after application of the mask. It is more complex. Red regions in the left picture have values higher than 0.003 and a blue regions values lower than -0.003. The mask influenced both upper and bottom face, but it was unable to completely remove signal from the bottom face and preserve it at the top face. The reason for it is that a single sine term cannot approximate signal situated close to the corner. Simply, higher harmonics are needed.

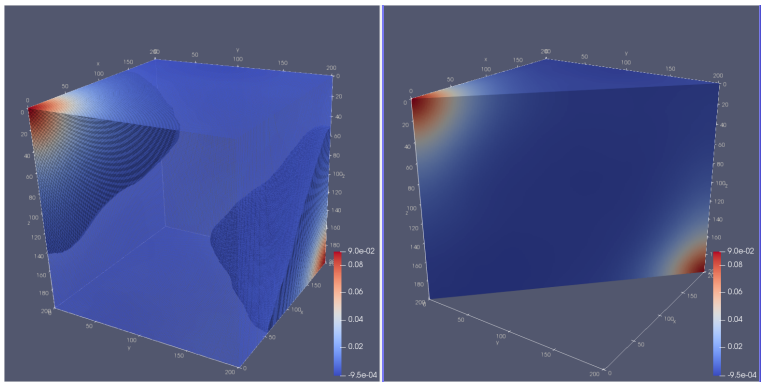
Fig. 4.2d shows the applied mask. We can observe that it affects the values in the entire volume. The highest values are at corners where the signal is located and along the diagonal in between. It comes from a mask component derived from the corners. The positive change in the middle is suppressed by sine terms, which are the source of the negative areas that have appeared.

Fig. 4.3 shows the same signal smoothed with $\frac{\Lambda}{h} = 20$. Fourier series were truncated after the third term.

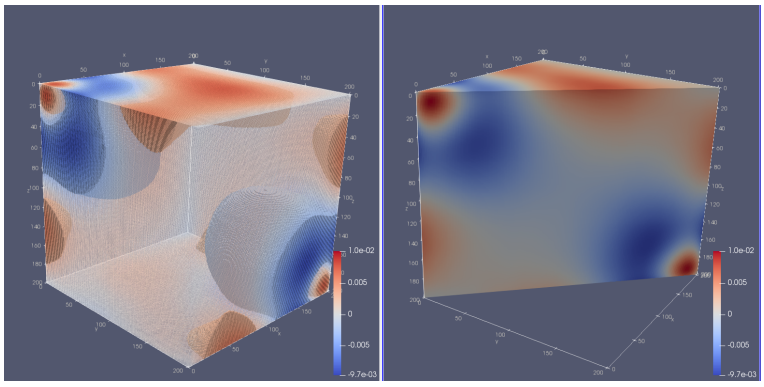
Fig. 4.3a shows upper and bottom face of the domain. The signal at the bottom face is quite well suppressed which confirms that if there is no discontinuity a fewer Fourier terms is enough.



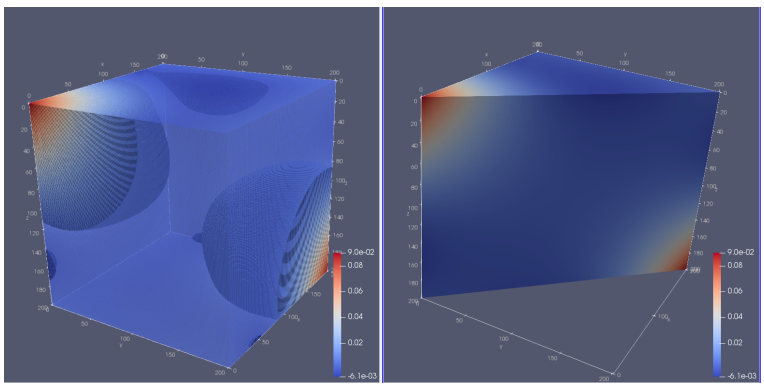
(a) Upper and bottom plane before and after application of a mask and the mask itself



(b) A signal before application of the mask

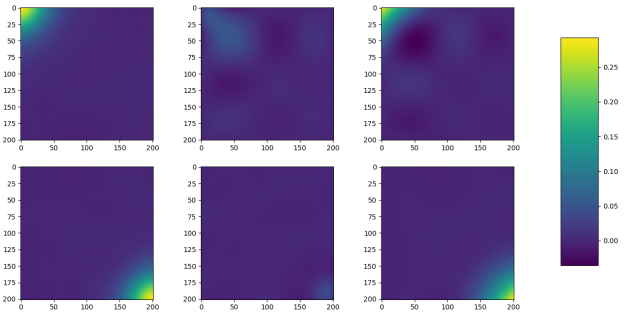


(c) A signal after application of the mask

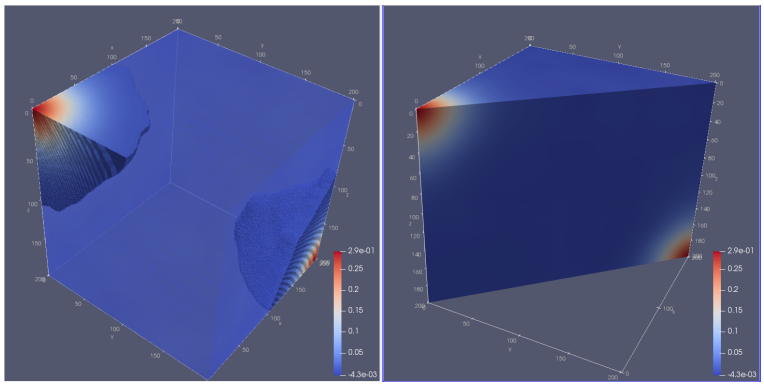


(d) An applied mask

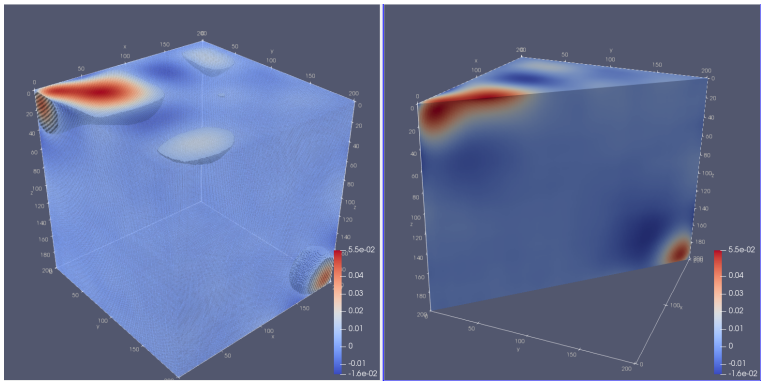
Figure 4.2: Application of mask to a pair of Gaussian peaks located at opposite corners with $\frac{\Lambda}{h} = 40$.



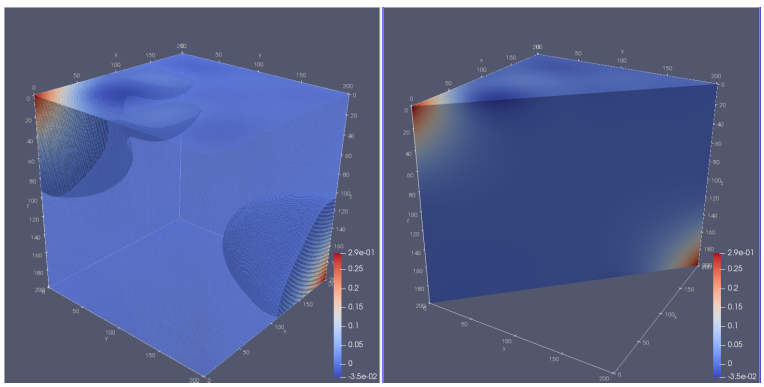
(a) Upper and bottom plane before and after application of a mask and the mask itself



(b) A signal before application of the mask



(c) A signal after application of the mask



(d) An applied mask

Figure 4.3: Application of mask to a pair of Gaussian peaks located at opposite corners with $\frac{\Lambda}{h} = 20$.

On the other hand, the signal at the upper face is still strongly affected by the mask. It is a nice demonstration of Gibbs phenomenon.

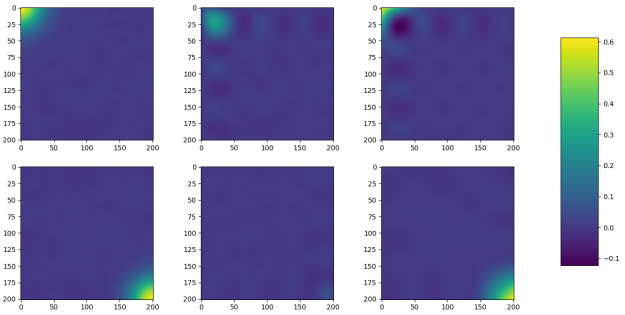
Fig. 4.3b and Fig. 4.3c depict the signal before and after the application of the mask. Values above 0.005 are depicted on the left pictures. The main features of the signal are preserved, but some additional perturbations have appeared. However, it is far better than in the previous case.

Finally, Fig. 4.4 shows the signal smoothed with $\frac{\Lambda}{h} = 10$. It means, that Fourier series were truncated after the sixth term. Therefore, the signal is affected by the mask only near the surface and it is well-preserved inside. However, we can still observe oscillations due to the Gibbs phenomenon. It is well demonstrated on a structure of the mask in the Fig. 4.4d, where we can clearly see the oscillatory character of the mask. A threshold on all left pictures is 0.01, which gives an insight to the amplitude of the oscillations.

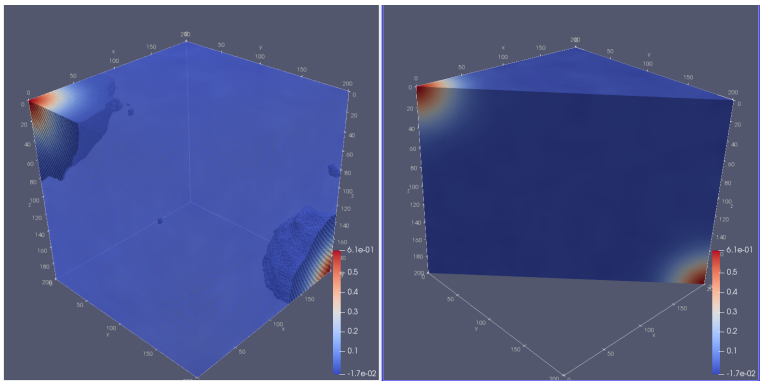
To conclude, we have demonstrated that the Gibbs phenomenon is a problem at the lowest scales if there is a signal close to the upper edges. In that case, the mask affects the signal also in greater depths. The Gibbs phenomenon can be observed also at the higher scales, but it affects only values near the surface, thus it is not a problem any more. However, we should be always careful when applying the mask and visually check how a kernel is affected by the mask.

4.3.4 Test of time complexity

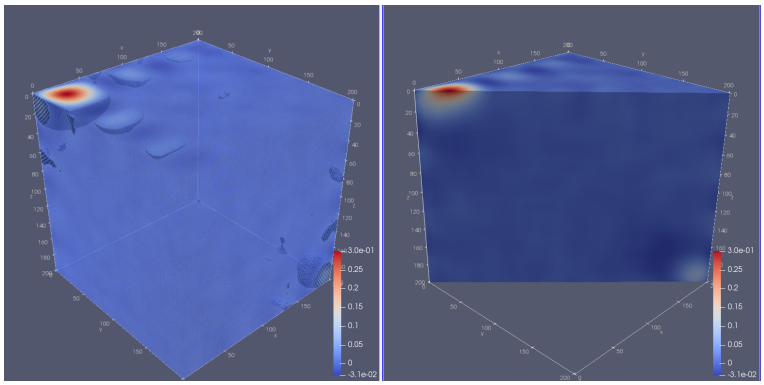
Last but not least, we tested a time complexity of an application of the mask. We were actuating the routine for the mask calculation and its application repeatedly with different dimensions of the computational domain and different smoothing intensities.



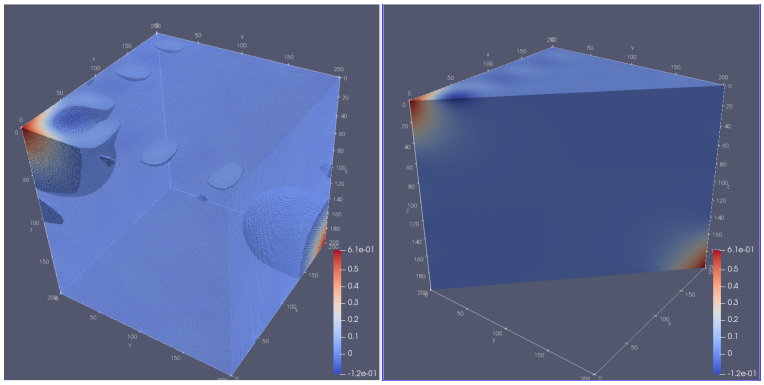
(a) Upper and bottom plane before and after application of a mask and the mask itself



(b) A signal before application of the mask



(c) A signal after application of the mask



(d) An applied mask

Figure 4.4: Application of mask to a pair of Gaussian peaks located at opposite corners with $\frac{\Lambda}{h} = 10$.

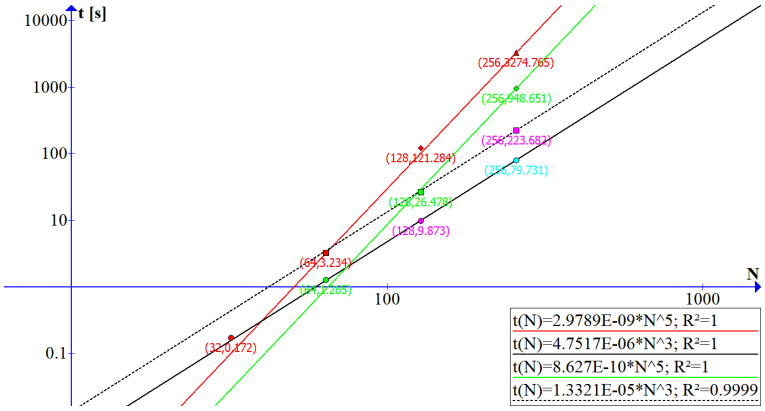


Figure 4.5: Computational times of mask application. Different colours indicate different smoothing intensities Λ (red – $8h$, green – $16h$, magenta – $32h$, cyan – $64h$). Individual symbols represents same ratio L/Λ (circle – 4, square – 8, diamond – 16, triangle – 32)

We considered only cubic domains and isotropic intensities. The results are illustrated on Fig. 4.5.

A same colour of symbols indicates same smoothing intensities. If we keep a constant smoothing intensity, a calculation of Fourier terms derived from faces becomes the step of the computation with highest time complexity. The reason is that a number of Fourier coefficients which has to be calculated increase with L/Λ and if Λ is constant, the time complexity of this step becomes $\mathcal{O}(N^5)$. A calculation of the Fourier coefficients is only of $\mathcal{O}(N^4)$. Therefore, we fitted the data with a function $t(N) = a \cdot N^5$ (red and green lines). A coefficient of determination is 1, thus it confirms that the designed function explains the computational times really well.

Individual symbols represent different ratios L/Λ. If we assume the ratio as constant, a time complexity of the computation of Fourier coefficients is only $\mathcal{O}(N^2)$. Therefore, the step with the highest time complexity is then an application of the mask which is of $\mathcal{O}(N^3)$, hence we fitted the data with a function $t(N) = a \cdot N^3$ (black lines) and we obtained a great fit.

The test revealed that the calculation and the consecutive application of the mask can be very computationally demanding for

large models at the lowest scales at which many terms of the Fourier series are needed to be calculated. Therefore, if we will work with large models, the calculation of the Fourier coefficients will have to be performed via a discrete fast Fourier sine transform (Frigo et al., 2005) and the application of the mask will have to be parallelised.

Conclusion

In this thesis, we addressed a topic of the adjoint tomography in local surface sedimentary structures in 3D. We built on the work of Kubina et al., 2018. We focused on kernel preprocessing.

First of all, we gave a brief insight to inverse problems. We explained the adjoint tomography method and the nature and necessity of kernel preprocessing.

Consequently, we made an overview of commonly used smoothing techniques in 3D. Besides a calculation of convolution in a space domain and filtering in a frequency domain, we briefly described two interesting methods – an application of the so-called Bessel filters and a topology-based smoothing. Especially, the former could be interesting if one needed anisotropic smoothing with main axes rotated with respect to Cartesian coordinate axes.

We developed and numerically tested:

- An efficient algorithm for smoothing of kernels in 3D. – We generalised the smoothing algorithm proposed by Kubina. In order to do that, we had to resolve a problem of interfaces between octants. We managed to do it by two distinct versions of the algorithm – one for the upper octants and another for the bottom ones. Furthermore, we managed to reduce RAM requirements of the smoothing algorithm by analytic calculation of the sums of weights and by expressing them in a close form.

- An algorithm for computation of a mask and its application to the kernel. – We adopted Kubina's design of the mask and generalised it to 3D. It required to resolve a problem of nulling edges of the computational domain.

We performed numerous numerical tests:

- A spike test confirmed a directional dependence of the used smoothing function. It also indicated how smoothing affects a spectrum of the smoothed data.

- Smoothing of a signal overlaid by noise verified an ability of the algorithm to separate the signal from the noise. We performed smoothing with different smoothing intensities. The stronger smoothing was applied, the better noise had been suppressed. However, a strong smoothing affected also the signal significantly. Numerical tests revealed that a smoothing over five grid points is enough to suppress the noise.

- A test of time complexity of the smoothing algorithm confirmed that the time complexity is $\mathcal{O}\left(N^3\right)$ which is even faster than Fast Fourier Transform ($\mathcal{O}\left(N^3 \log N\right)$). In addition, it revealed that the version of the algorithm with an analytic calculation of sums of weights for normalisation is slightly slower comparing to the version which calculates the sums of weights by explicit summation during a calculation of convolution sums. However, the faster algorithm has higher RAM requirements.

- Tests of a mask revealed a presence of the Gibbs phenomenon in the mask due to a discontinuity in boundary conditions along upper edges.

- A test of time complexity of the mask calculation and its application revealed that it is quite computationally demanding. Nevertheless, it can be done relatively quickly, except for large models at the lowest scales. In that case, the application of the mask has to be parallelised.

To sum up, we created routines for smoothing of the kernels and for application of the mask. We tested them on artificial data. Real data are not available at the time of submission of the thesis, as the computational program `FDAtom3D` for computation of the adjoint tomography of local surface sedimentary structures in 3D is not able to calculate kernels yet. It is a question of a next couple of weeks. Nevertheless, we performed a quite extensive numerical testing and we can conclude that the routines are prepared to be implemented into the program.

References

ASTER, Rick; BORCHERS, Brian; THURBER, Clifford, 2003. *Parameter Estimation and Inverse Problems*.

BAMBERGER, Alain; CHAVENT, Guy; LAILLY, Patrick, 1979. About the Stability of the Inverse Problem in 1-D Wave Equations – Application to the Interpretation of Seismic Profiles. *Applied Mathematics and Optimization*. Vol. 5, pp. 1–47.

BOZDAĞ, Ebru; PETER, Daniel; LEFEBVRE, Matthieu; KOMATITSCH, Dimitri; TROMP, Jeroen; HILL, Judith; PODHORZSKI, Norbert; PUGMIRE, David, 2016. Global adjoint tomography: first-generation model. *Geophysical Journal International*. Vol. 207, pp. 1739–1766. Available from DOI: 10.1093/gji/ggw356.

BRAVERMAN, Elena; ISRAELI, Moshe; AVERBUCH, Amir; VOZOVOI, Lev, 1998. A Fast 3D Poisson Solver of Arbitrary Order Accuracy. *Journal of Computational Physics*. Vol. 144, pp. 109–136. Available from DOI: 10.1006/jcph.1998.6001.

BREMER, Peer-Time; EDELSBRUNNER, Herbert; HAMANN, Bernd; PASCUCCI, Valerio, 2004. A Topological Hierarchy for Functions on Triangulated Surfaces. *Transactions on Visualization and Computer Graphics*. Vol. 10, no. 4, pp. 385–396. Available from DOI: 10.1109/TVCG.2004.3.

CAPDEVILLE, Yann; GUNG, Yuancheng; ROMANOWICZ, Barbara, 2005. Towards global earth tomography using the spectral element method: a technique based on source stacking. *Geophysical Journal International*. Vol. 162, pp. 541–554. Available from DOI: 10.1111/j.1365-246X.2005.02689.x.

CHEN, Min; NIU, Fenglin; LIU, Qinya; TROMP, Jeroen, 2015. Mantle-driven uplift of Hangai Dome: New seismic constraints from adjoint tomography. *Geophysical Research Letters*. Vol. 42, pp. 6967–6974. Available from DOI: 10.1002/2015GL065018.

CHEN, Min; NIU, Fenglin; LIU, Qinya; TROMP, Jeroen; ZHENG, Xiufen, 2015. Multi-parameter adjoint tomography of the crust and upper mantle beneath East Asia: 1. Model construction and comparison. *Journal of Geophysical Research: Solid Earth*. Vol. 120, pp. 1762–1786. Available from DOI: 10.1002/2014JB011638.

CHEN, Min; NIU, Fenglin; TROMP, Jeroen; LENARDIC, Adrian; LEE, Cin-Ty A.; CAO, Wenrong; RIBEIRO, Julia, 2017. Lithospheric foundering and underthrusting imaged beneath Tibet. *Nature Communications*. Vol. 8, pp. 1–10. Available from DOI: 10.1038/ncomms15659.

CHEN, Po, 2013. Full-3D, Full-Wave Seismic Tomography for Crustal Structure in Southern California, USA. *Acta Geologica Sinica*. Vol. 87, pp. 25–27.

CHEN, Po; JORDAN, Thomas Hillman; ZHAO, Li, 2007. Full three-dimensional tomography: a comparison between the scattering-integral and adjoint-wavefield methods. *Geophysical Journal International*. Vol. 170, pp. 175–181. Available from DOI: 10.1111/j.1365-246X.2007.03429.x.

CHEN, Po; ZHAO, Li; JORDAN, Thomas Hillman, 2007. Full 3D Tomography for the Crustal Structure of the Los Angeles Region. *Bulletin of the Seismological Society of America*. Vol. 97, no. 4, pp. 1094–1120. Available from DOI: 10.1785/0120060222.

EDELSBRUNNER, Herbert; HARER, John; ZOMORODIAN, Afra, 2003. Hierarchical Morse-Smale Complexes for Piecewise Linear 2-Manifolds. *Discrete and Computational Geometry*. Vol. 30, no. 1, pp. 87–107. Available from DOI: 10.1007/s00454-003-2926-5.

EDELSBRUNNER, Herbert; LETSCHER, David; ZOMORODIAN, Afra, 2002. Topological Persistance and Simpification. *Discrete and Computational Geometry*. Vol. 28, pp. 511–533. Available from DOI: 10.1007/s00454-002-2885-2.

FICHTNER, Andreas, 2011. *Full Seismic Waveform Modelling and Inversion*. 1st ed. Springer-Verlag Berlin Heidelberg. ISBN 978-3-642-15806-3. Available from DOI: 10.1007/978-3-642-15807-0.

FICHTNER, Andreas; BUNGE, Hans-Peter; IGEL, Heiner, 2006a. The adjoint method in seismology I. Theory. *Physics of the Earth and Planetary Interiors*. Vol. 157, pp. 86–104. Available from DOI: 10.1016/j.pepi.2006.03.016.

FICHTNER, Andreas; BUNGE, Hans-Peter; IGEL, Heiner, 2006b. The adjoint method in seismology II. Applications: traveltimes and sensitivity functionals. *Physics of the Earth and Planetary Interiors*. Vol. 157, pp. 105–123. Available from DOI: 10.1016/j.pepi.2006.03.018.

FICHTNER, Andreas; KENNETT, Brian Leslie Norman; IGEL, Heiner; BUNGE, Hans-Peter, 2009. Full seismic waveform tomography for upper-mantle structure in the Australasian region using adjoint methods. *Geophysical Journal International*. Vol. 179, pp. 1703–1725. Available from DOI: 10.1111/j.1365-246X.2009.04368.x.

FICHTNER, Andreas; KENNETT, Brian Leslie Norman; IGEL, Heiner; BUNGE, Hans-Peter, 2010. Full waveform tomography for radially anisotropic structure: New insights into present and past states of the Australasian upper mantle. *Earth and Planetary Science Letters*. Vol. 290, pp. 270–280. Available from DOI: 10.1016/j.epsl.2009.12.003.

FICHTNER, Andreas; TRAMPERT, Jeannot; CUPILLARD, Paul; SAYGIN, Erdinc; TAYMAZ, Tuncay; CAPDEVILLE, Yan; VILLASEÑOR, Antonio, 2013. Multiscale full waveform inversion. *Geophysical Journal International*. Vol. 194, pp. 534–556. Available from DOI: 10.1093/gji/ggt118.

FORMAN, Robin, 1998. Morse Theory for Cell Complexes. *Advances in Mathematics*. Vol. 134, pp. 90–145. Available from DOI: 10.1006/aima.1997.1650.

FRIGO, Matteo; JOHNSON, Steven Glenn, 2005. The Design and Implementation of FFTW3. *Proceedings of the IEEE*. Vol. 93, no. 2, pp. 216–231. Available from DOI: 10.1109/JPROC.2004.840301.

GÜNTHER, David; JACOBSON, Alec; REININGHAUS, Jan; SEIDEL, Hans-Peter; SORKINE-HORNUNG, Olga; WEINKAUF, Tino, 2014. Fast and Memory-Efficient Topological Denoising of 2D and 3D Scalar Fields. *IEEE Transactions on Visualization and Computer Graphics*. Vol. 20, no. 12, pp. 2585–2594. Available from DOI: 10.1109/TVCG.2014.2346432.

GYULASSY, Attila; BREMER, Peer-Timo; HERMANN, Bernd; PASCUCCI, Valerio, 2008. A Practical Approach to Morse-Smale Complex Computation: Scalability and

Generality. *IEEE Transactions on Visualization and Computer Graphics*. Vol. 14, no. 6, pp. 1619–1626. Available from DOI: 10.1109/TVCG.2008.110.

GYULASSY, Attila; NATARAJAN, Vijay; PASCUCCI, Valerio; BREMER, Peer-Timo; HAMANN, Bernd, 2005. Topology-based Simplification for Feature Extraction from 3D Scalar Fields. In: *Topology-based Simplification for Feature Extraction from 3D Scalar Fields. 16th IEEE Visualization Conference*, pp. 535–542. Available from DOI: 10.1109/VISUAL.2005.1532839.

KUBINA, Filip, 2013. *Adjungovaná tomografia a jej aplikácia na Mygdónsky bazén*. Master's thesis. Univerzita Komenského v Bratislave.

KUBINA, Filip, 2017. *Adjoint tomography of 2D local surface sedimentary structures*. PhD thesis. Comenius University in Bratislava.

KUBINA, Filip; MICHLÍK, Filip; MOCZO, Peter; KRISTEK, Jozef; STRIPAJOVÁ, Svetlana, 2018. Adjoint Tomography for Predicting Earthquake Ground Motion: Methodology and a Blind Test. *Bulletin of the Seismological Society of America*. Vol. 108, no. 3A, pp. 1257–1271. Available from DOI: 10.1785/0120170265.

LEE, En-Jui; CHEN, Po, 2016. Improved Basin Structures in Southern California Obtained Through Full-3D Seismic Waveform Tomography (F3DT). *Seismological Research Letters*. Vol. 87, no. 4, pp. 1–8. Available from DOI: 10.1785/0220160013.

LEE, En-Jui; CHEN, Po; JORDAN, Thomas Hillman; MAECHLING, Phillip B.; DENOLLE, Marine A. M.; BEROZA, Gregory C., 2014. Full-3-D tomography for crustal structure in Southern California based on the scattering-integral and the adjoint-wavefield methods. *Journal of Geophysical Research: Solid Earth*. Vol. 119, pp. 6421–6451. Available from DOI: 10.1002/2014JB011346.

LIU, Qinya; TROMP, Jeroen, 2008. Finite-frequency sensitivity kernels for global seismic wave propagation based upon adjoint methods. *Geophysical Journal International*. Vol. 174, pp. 265–286. Available from DOI: 10.1111/j.1365-246X.2008.03798.x.

LIU, Yaning; NIU, Fenglin; CHEN, Min; YANG, Wencai, 2017. 3-D crustal and uppermost mantle structure beneath Ne China revealed by ambient noise adjoint tomography. *Earth and Planetary Science Letters*. Vol. 461, pp. 20–29. Available from DOI: 10.1016/j.epsl.2016.12.029.

MICHLÍK, Filip, 2013. *Adjungovaná tomografia seizmického pohybu v lokálnych povrchových štruktúrach*. Bachelor Thesis. Univerzita Komenského v Bratislave.

MICHLÍK, Filip, 2015. *Adjungovaná tomografia lokálnej štruktúry vo vysokofrekvenčnom prípade*. Master's thesis. Univerzita Komenského v Bratislave.

MOCZO, Peter; KRISTEK, Jozef; GÁLIS, Martin, 2014. *The Finite-Difference Modelling of Earthquake Motions: Waves and Ruptures*. Cambridge University Press. ISBN 978-1-107-02881-4.

OPRSAL, Ivan; ZAHRADNIK, Jiri, 2002. Three-dimensional finite difference method and hybrid modeling of earthquake ground motion. *Journal of Geophysical Research*. Vol. 107, no. B8, pp. ESE 2-1-ESE 2-16. Available from DOI: 10.1029/2000JB000082.

RICKERS, Florian; FICHTNER, Andreas; TRAMPERT, Jeannot, 2013. The Iceland-Jan Mayen plume system and its impact on mantle dynamics in the North Atlantic region: Evidence from full-waveform inversion. *Earth and Planetary Science Letters*. Vol. 367, pp. 39–51. Available from DOI: 10.1016/j.epsl.2013.02.022.

SIEMINSKI, Anne; LIU, Qinya; TRAMPERT, Jeannot; TROMP, Jeroen, 2007a. Finite-frequency sensitivity of body waves to anisotropy based upon adjoint methods. *Geophysical Journal International*. Vol. 171, pp. 368–389. Available from DOI: 10.1111/j.1365-246X.2007.03528.x.

SIEMINSKI, Anne; LIU, Qinya; TRAMPERT, Jeannot; TROMP, Jeroen, 2007b. Finite-frequency sensitivity of surface waves to anisotropy based upon adjoint methods. *Geophysical Journal International*. Vol. 168, pp. 1153–1174. Available from DOI: 10.1111/j.1365-246X.2006.03261.x.

STICH, Daniel; DANECEK, Peter; MORELLI, Andrea; TROMP, Jeroen, 2009. Imaging lateral heterogeneity in the northern Apennines from time reversal of reflected surface waves. *Geophysical Journal International*. Vol. 177, pp. 543–554. Available from DOI: 10.1111/j.1365-246X.2008.04044.x.

TAPE, Carl; LIU, Qinya; MAGGI, Alessia; TROMP, Jeroen, 2009. Adjoint Tomography of the Southern California Crust. *Science*. Vol. 325, pp. 988–992. Available from DOI: 10.1126/science.1175298.

TAPE, Carl; LIU, Qinya; MAGGI, Alessia; TROMP, Jeroen, 2010. Seismic tomography of the southern California crust based on spectral-element and adjoint methods. *Geophysical Journal International*. Vol. 180, pp. 433–462. Available from DOI: 10.1111/j.1365-246X.2009.04429.x.

TARANTOLA, Albert, 1984. Inversion of seismic reflection data in the acoustic approximation. *Geophysics*. Vol. 49, no. 8, pp. 1259–1266. Available from DOI: 10.1190/1.1441754.

TARANTOLA, Albert, 1988. Theoretical Background for the Inversion of Seismic Waveforms, Including Elasticity and Attenuation. *Pure and Applied Geophysics*. Vol. 128, pp. 365–399. Available from DOI: 10.1007/BF01772605.

TARANTOLA, Albert, 2005. The Norm Associated with the 1D Exponential Covariance. In: *Inverse Problems Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, chap. 7 Problems, pp. 308–311. ISBN 0-89871-572-5.

TRINH, Phuong-Thu; BROSSIER, Romain; MÉTIVIER, Ludovic; VIRIEUX, Jean; WELLINGTON, Paul, 2017. Bessel smoothing filter for spectral-element mesh. *Geophysical Journal International*. Vol. 209, pp. 1489–1512. Available from DOI: 10.1093/gji/ggx103.

TROMP, Jeroen; TAPE, Carl; LIU, Qinya, 2005. Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels. *Geophysical Journal International*. Vol. 160, pp. 195–216. Available from DOI: 10.1111/j.1365-246X.2004.02453.x.

WEINKAUF, Tino; GINGOLD, Yotam; SORKINE, Olga, 2010. Topology-based Smoothing of 2D Scalar Fields with C1-Continuity. In: *Topology-based Smoothing of 2D Scalar Fields with C1-Continuity. Computer Graphics Forum*. Vol. 29, pp. 1221–1230. No. 3. Available from DOI: 10.1111/j.1467-8659.2009.01702.x.

WELLINGTON, Paul John, 2016. *Efficient 1D, 2D and 3D Geostatistical constraints and their application to Full Waveform Inversion*. PhD thesis. Université Grenoble Alpes.

ZHU, Hejun; BOZDAĞ, Ebru; DUFFY, Thomas S.; TROMP, Jeroen, 2013. Seismic attenuation beneath Europe and the North Atlantic: implications for water in the

mantle. *Earth and Planetary Science Letters*. Vol. 381, pp. 1–11. Available from DOI: 10.1016/j.epsl.2013.08.030.

ZHU, Hejun; BOZDAĞ, Ebru; PETER, Daniel; TROMP, Jeroen, 2012. Structure of the European upper mantle revealed by adjoint tomography. *Nature Geoscience*. Vol. 5, pp. 493–498. Available from DOI: 10.1038/NGEO1501.

ZHU, Hejun; BOZDAĞ, Ebru; TROMP, Jeroen, 2015. Seismic structure of the European upper mantle based on adjoint tomography. *Geophysical Journal International*. Vol. 201, pp. 18–52. Available from DOI: 10.1093/gji/ggu492.

Abstract

Jaroslav Valovčan: *Kernel preprocessing in 3D adjoint tomography of local surface sedimentary structures.* [Master's thesis]. Comenius University in Bratislava. Faculty of Mathematics, Physics and Informatics, Department of Astronomy, Physics of the Earth and Meteorology. Supervisor: prof. RNDr. Peter Moczo, DrSc. Bratislava 2019. 127 pgs. Degree of qualification: Master.

Seismic ground motion can be strongly affected by local surface structures. Therefore, if we are to model a propagation of seismic motion in the local surface sedimentary structures, we do not need only a precise program for a numerical modelling of the propagation of seismic motion, but we need also a sufficiently precise structural model. It can be obtained by seismic tomographic methods. In the recent years, full-waveform inverse methods have been widely used. One of the full-waveform methods is the adjoint tomography. The adjoint tomography improves a structural model using a so-called kernel which is a volume density of gradient of misfit between the observed and calculated seismograms. The gradient is evaluated with respect to model parameters. The adjoint tomography has been applied over wide range of scales – from regional to global. However, it has not been employed at a local scale in 3D yet. The reason for that is an ill-posedness of the problem. The inversion in the local surface structures is specific by a relatively small amount of available data, a high initial misfit and short length scale heterogeneities in the model. Therefore, a proper kernel preconditioning is necessary. We build on the pioneering work of Filip Kubina who was the first to employ the adjoint method to a tomography of local surface sedimentary structures in 2D. We focus on kernel preconditioning. We generalised algorithms for smoothing of a kernel and application of a mask to the kernel proposed by Kubina. We created routines for efficient smoothing of kernels and for a cal-

culation and an application of the mask in 3D. We performed an extensive numerical testing of the routines on artificial data, in order to determine their properties. The routines will be implemented in the program `FDAtom3D` by Filip Michlík which performs the adjoint tomography of local surface sedimentary structures.

Keywords: adjoint tomography, local surface sedimentary structures, kernel preprocessing, smoothing, mask

Abstrakt

Jaroslav Valovčan: *Kernel-Vorverarbeitung in der 3D adjungierten Tomographie lokaler Oberflächen-Sedimentstrukturen.* [Master-Arbeit]. Comenius-Universität in Bratislava. Fakultät für Mathematik, Physik und Informatik, Abteilung für Astronomie, Physik der Erde und Meteorologie. Betreuer: prof. RNDr. Peter Moczo, DrSc. Bratislava 2019. 127 Stn. Qualifikationsgrad: Master.

Seismische Bodenbewegungen können durch lokale Oberflächenstrukturen stark beeinflusst werden. Wenn wir deshalb eine Ausbreitung der seismischen Bewegung in den lokalen Oberflächen-Sedimentstrukturen modellieren sollen, brauchen wir nicht nur ein genaues Programm für eine numerische Modellierung der Ausbreitung der seismischen Bewegung, sondern auch ein ausreichend genaues Strukturmodell. Das kann durch seismische tomographische Methoden erhalten werden. In den letzten Jahren sind Vollwellenform inverse Methoden in großem Umfang verwendet worden. Eine der Vollwellenformmethoden ist die adjungierte Tomographie. Die adjungierte Tomographie verbessert ein Strukturmodell unter Verwendung eines sogenannten Kernels, bei dem es sich um eine Volumendichte des Gradienten der Misfit-Funktion zwischen dem beobachteten und dem berechneten Seismogramm handelt. Der Gradient wird in Bezug auf Modellparameter ausgewertet. Die adjungierte Tomographie ist über ein breites Spektrum von Skalen angewendet worden - von regional bis global. Es wurde jedoch noch nicht in 3D auf lokaler Skala eingesetzt. Der Grund dafür ist eine Schlechtgestelltheit des Problems. Die Inversion in den lokalen Oberflächenstrukturen ist spezifisch durch eine relativ kleine Menge verfügbarer Daten, eine hohe anfängliche Misfit-Funktion und Heterogenitäten im Modell mit kurzer Länge. Daher ist eine ordnungsgemäße

Kernel-Vorkonditionierung erforderlich. Wir bauen auf der Pionierarbeit von Filip Kubina auf, der als erster die adjungierte Methode für eine Tomographie lokaler Oberflächen-Sedimentstrukturen in 2D einsetzte. Wir fokussieren uns auf die Vorkonditionierung des Kernels. Wir verallgemeinerten Algorithmen zum Glätten eines Kernels und Anwenden einer Maske auf den Kernel, die von Kubina vorgeschlagen worden waren. Wir erstellten Routinen zum effizienten Glätten von Kerneln sowie zur Berechnung und Anwendung der Maske in 3D. Wir führten eine umfangreiche numerische Prüfung der Routinen an künstlichen Daten durch, um deren Eigenschaften zu bestimmen. Die Routinen werden im Programm `FDAtom3D` von Filip Michlík implementiert, das die adjungierte Tomographie lokaler Oberflächen-Sedimentstrukturen durchführt.

Schlagwörter: adjungierte Tomographie, lokale Oberfläche--Sedimentstrukturen, Kernel-Vorverarbeitung, Glätten, Maske