Learning Dictionaries for Inverse Problems on the Sphere

DISSERTATION
zur Erlangung des Grades eines Doktors
der Naturwissenschaften

vorgelegt von
Naomi Schneider, M. Sc.

eingereicht bei der Naturwissenschaftlich-Technischen Fakultät
der Universität Siegen
Siegen 2020

gedruckt auf Papier nach DIN ISO 9706

1. Gutachter: Prof. Dr. Volker Michel, Universität Siegen
2. Gutachter: Prof. Dr. Nico Sneeuw, Universität Stuttgart

Tag der mündlichen Prüfung: 10. August 2020

To

Mum – for planting all necessary seeds in me
Dad – for protecting their growth

Genesis 24:22

So I say thank you

First and foremost, I thank the Lord for guiding me through the minor and the major questions in life and this thesis with the hidden and the obvious hints.

Special thanks go to my supervisor Prof. Dr. Volker Michel for giving me the opportunity to write this thesis, for his commitment to mentoring the next generation and, in particular, for creating an atmosphere in the work group that gives you space to grow up as a scientist as well as a human being.

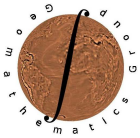
Furthermore, I thank Prof. Dr. Nico Sneeuw for his interest in my work.

The financial support of the "Deutsche Forschungsgemeinschaft (DFG)" is gratefully acknowledged.

Moreover, I thank all former and present members of the Geomathematics Group Siegen for many thought-provoking impulses. In particular, I thank Dr. Roger Telschow for providing me with an irregular data grid, Dr. Max Kontak for his interest in all kinds of mathematical questions and Dr. Sarah Leweke for handing me implementations of the spherical harmonics during my Master's Thesis which were re-used.
Major thanks go to Bianca Kretz for so many invaluable conversations and support – so many things done, so many years to come.

Last but not least, many heartfelt thanks go to those who are closest to me and without whom I would have surrendered long ago:
to my aunt for always being interested in me,
to my dad for his unquestionable support in every possible way,
for always listening and asking the right questions,
for so many hours on our feet,
to my feline Columbo for showing me everyday how little really matters for a good life.

Naomi Schneider
Geomathematics Group Siegen
University of Siegen



Gefördert durch
DFG Deutsche Forschungsgemeinschaft

Zusammenfassung

Die Berechnung des Gravitationspotentials auf der Erdoberfläche aus Satellitendaten ist ein schlecht gestelltes sphärisches inverses Problem. Die GRACE-Mission (NASA/DLR) liefert uns u. a. die monatlichen Abweichungen zu diesem Potential. Dadurch erhalten wir Einblicke in den Massentransport auf der Erde und können insbesondere den Klimawandel visualisieren.
Traditionell wird eine approximative Lösung eines inversen Problems in einer Basis, wie z. B. Kugelflächenfunktionen, dargestellt. Dagegen bestimmen die Inverse Problem Matching Pursuit (IPMP) Algorithmen iterativ eine Darstellung in Dictionary-Elementen. Ein Dictionary enthält üblicherweise globale und lokale Ansatzfunktionen wie Kugelflächenfunktionen, Slepian-Funktionen und radiale Basisfunktionen. Man sieht leicht, dass die A-priori-Wahl eines endlichen Dictionarys prägend für den Verlauf der Algorithmen ist.
Daher entwickeln wir in dieser Arbeit die Learning Inverse Problem Matching Pursuit (LIPMP) Algorithmen. Die Wahl eines Dictionary-Elementes in einem Iterationsschritt wird hier um mehrere nichtlineare Optimierungsprobleme erweitert. Auf diese Weise werden automatisch endlich viele optimierte Ansatzfunktionen aus unendlich Vielen gelernt.
Zunächst fassen wir in dieser Arbeit grundlegende Resultate zusammen, die notwendig für das weitere Verständnis sind. Danach entwickeln wir die LIPMP-Algorithmen. Dabei legen wir ein besonderes Augenmerk auf Satellitendaten des Gravitationspotentials. Wir betrachten ebenfalls einige theoretische Aspekte der Methoden. Außerdem präsentieren wir numerische Experimente, die die praktische Anwendbarkeit der Algorithmen für die Downward Continuation belegen.

Abstract

The downward continuation of the gravitational potential from satellite data is a spherical ill-posed inverse problem in the geosciences. The GRACE mission (NASA/DLR) supplied us, e. g., with monthly deviations from this potential. This enables us to consider the mass transport on the Earth and, thus, visualize the climate change.
Classically, an approximative solution of an inverse problem is expanded in a suitable basis like spherical harmonics. Alternatively, the Inverse Problem Matching Pursuit (IPMP) algorithms iteratively build a linear combination of dictionary elements. A dictionary usually consists of global and local trial functions, such as spherical harmonics, Slepian functions and radial basis functions. Consequently, an a-priori choice of a finite dictionary is of major importance for the IPMP algorithms.
Therefore, we develop the Learning Inverse Problem Matching Pursuit (LIPMP) algorithms in this thesis. Here, the choice of a dictionary element in each iteration is extended by several non-linear optimization problems. In this way, the LIPMP algorithms automatically learn a finite number of optimized trial functions from infinitely many possible ones.
First, we introduce the basic results necessary for the understanding of the novel learning methods. Then we develop the LIPMP algorithms. There, we have a closer look on the downward continuation of satellite data of the gravitational potential. We also consider some of the theoretical aspects of the methods. Further, we present numerical experiments which underline the applicability of the strategies for the downward continuation problem.

Contents

List of Figures

List of Tables

1. About dictionary learning in geomathematics

In this thesis, an enhancement of matching pursuit algorithms for spherical inverse problems is presented. The novel methods are based on non-linear optimization models for different types of trial functions and are applied to the downward continuation of satellite data.

Geophysics is concerned with the investigation of the system Earth. Improving our understanding of this field is of major importance because – as the well-known proverb goes – knowledge is power. It enables us to analyse past events, take precautions against potentially dangerous developments and react quickly to novel discoveries. In that respect, essential aspects of the system Earth that are under investigation are, for instance, the behaviour of the magnetic field of the Earth (see e. g. Akram and Michel, 2010; Friis-Christensen et al., 2006; Leweke et al., 2018b; Mandea and Dormy, 2003; Olsen and Mandea, 2007; Olson and Amit, 2006), of volcanoes (see e. g. Blackett, 2014; Casagli et al., 2010; Senyukov, 2013; Sparks and Aspinall, 2004; Vikulin et al., 2012) as well as of the climate change (see e. g. IPCC, 2014; Fischer and Michel, 2013b; Lin et al., 2018; NASA, 2020; Sneeuw and Saemian, 2019). For such studies, geomathematics aims to enhance geophysical methods as well as reference values that are significant.

The gravitational potential is an example for a quantity of the Earth in need to be monitored. The current reference model is the Earth Gravitational Model 2008 (EGM2008; see e. g. National Geospatial-Intelligence Agency, Office of Geomatics (SN), EGM Development Team, 2008; Pavlis et al., 2012). The potential is given in high-precision as an expansion in fully normalized spherical harmonics, i. e. orthogonal polynomials on the sphere, up to degree 2190 and order 2159 and is based on differently originated data like terrestrial as well as airborne measurements. Besides the EGM2008, spaceborne models are available from satellite missions like the Gravity Recovery and Climate Experiment (GRACE) and its successor GRACE-Follow On (GRACE-FO; for both see e. g. Devaraju and Sneeuw, 2017; Flechtner et al., 2014a,b; NASA Jet Propulsion Laboratory, 2020; Schmidt et al., 2008; Tapley et al., 2004; The University of Texas at Austin, Centre for Space Research, 2020). In contrast to the EGM2008, these models are time-dependent. Thus, the mass transport of the Earth can be investigated with them. This yields visualizations of seasonal short-term phenomena like the wet-season in the Amazon basin as well as long-term developments like the climate change. However, we are usually interested in the gravitational potential at the Earth's

surface. Thus, spaceborne data need to be downward continued to be meaningful. Hence, gravity field modelling is a critical challenge in physical geodesy (see e. g. Baur, 2014; Kusche, 2015).
It is a challenge because, mathematically formulated, the downward continuation of satellite data is an exponentially ill-posed inverse problem. Assume that the Earth's surface is the unit sphere Ω. Traditionally, the gravitational potential f on Ω can be expanded in a suitable basis like the mentioned spherical harmonics $Y_{n,j}$, $n \in \mathbb{N}_0$, $j = -n, ..., n$. Then we have a pointwise representation

$$V(\sigma\eta) = (\mathcal{T}f)(\sigma\eta) = \sum_{n=0}^{\infty} \sum_{j=-n}^{n} \langle f, Y_{n,j} \rangle_{\mathrm{L}^2(\Omega)} \sigma^{-n-1} Y_{n,j}(\eta) \tag{1.1}$$

of the gravitational potential on a relative satellite orbit height $\sigma > 1$ for any point $\eta \in \Omega$ (see e. g. Baur, 2014; Freeden and Michel, 2004a; Moritz, 2010; Telschow, 2014). In practice, we usually have perturbed measurements of V on discrete points $\sigma\eta$. Then the task is to determine (at least) an approximation of f, i. e. we have to downward continue the potential to the Earth's surface or, mathematically speaking, we need to solve the inverse problem $V = \mathcal{T}f$. For more details on inverse problems in general, see the classical literature such as Engl et al. (1996); Louis (1989); Rieder (2003). The challenge herein is due to the exponentially decreasing singular values of $\mathcal{T}$ (confer (1.1)) as it holds $\sigma > 1$ (see e. g. Michel, 2005; Telschow, 2014). Consequently, the respective inverse operator has exponentially increasing singular values. This contradicts a continuous dependence of the solution on the data. In other words, with perturbed data, the third characteristic of a well-posed problem according to Hadamard is violated. Due to this behaviour of the singular values, the problem of the downward continuation of satellite data is said to be exponentially ill-posed. For the sake of completeness, we mention that any solution f has to be in the range of $\mathcal{T}$. In this case, it is always unique.
Therefore, the approximation of the solution of an ill-posed inverse problem like the downward continuation of satellite data demands for sophisticated mathematical approaches. Traditionally, an approximation is sought as an expansion in a predefined basis. However, the a-priori choice for a particular basis may not be an easy one. This is in particular true if only irregularly distributed or local data are available.
Due to the wide range of possible bases, we discuss some common but very different trial functions at this point. A first intuitive choice of basis functions are the already mentioned global spherical harmonics. For more details on these functions, see, for instance, Freeden et al. (1998); Freeden and Gutting (2013); Freeden and Michel (2004a); Freeden and Schreiner (2009); Michel (2013); Müller (1966). These orthogonal polynomials are commonly used for approximation. However, it is well known that global functions lead to oscillations in data gaps. Further, altering the data locally may have a global impact on the approximation. Local functions like radial basis functions (RBFs), for instance the Abel–Poisson kernel

and wavelet (see e. g. Freeden et al., 1998; Freeden and Michel, 2004a; Freeden and Schreiner, 1998, 2009; Freeden and Windheuser, 1996; Michel, 2013; Windheuser, 1995), are better able to cope with data gaps or for local approximations. In general, these functions attain one extremum at a certain centre $\chi \in \Omega$ on the unit sphere Ω and are nearly zero on the outside of a neighbourhood of χ. Thus, they look like a 'hat' placed at χ on the sphere. The size of the neighbourhood or of the hat or – in short – the scale can be varied. This is helpful because, when used as a basis for approximation, the scale may need to be related to the maximal diameter of present data gaps. Otherwise the approximation could be overfitted in the gap. However, using a function of a largely-sized hat may lead to an approximation which is smoother than necessary on regions with a high density of data points. Similar problems occur when Slepian functions (see e. g. Albertella et al., 1999; Grünbaum et al., 1982; Leweke et al., 2018a; Michel, 2013; Seibert, 2018; Simons and Dahlen, 2006) are used. These are band-limited optimally localized trial functions. However, if the localization region is large, they are more similar to global functions. Otherwise, if the localization region is chosen to be small, they are more localized. Then, however, we have to deal again with overfitting in data gaps. Hence, we see that the different types of trial functions all carry particular advantages as well as disadvantages for approximation tasks.
In order to overcome the respective disadvantages, different approaches were developed in order to improve the approximation process (see e. g. Antoni, 2012; Barthelmes, 1986, 1989; Barthelmes et al., 1991; Claessens et al., 2001; DiMatteo et al., 2001; Eicker et al., 2014; Jekeli, 2005; Lehmann, 1993; Lin, 2016; Lin et al., 2014; Schall, 2019; Schall et al., 2011, 2014). In these methods, for instance, free point masses or splines with free-knots are used, the data is manually separated into global and local aspects or satellite tracks are considered as short arcs. The procedures proved to be suitable in numerical experiments. However, most of them work with only one type of trial functions which are or are similar to point masses (in our language radial basis functions). Then the location of their extremum as well as their scale is usually (stochastically or regionally) optimized. Hence, they do present automations for the selection of a certain type of trial functions in approximation tasks.
Another and rather different approach is given by the (Regularized) Functional Matching Pursuit ((R)FMP) and the (Regularized) Orthogonal Functional Matching Pursuit ((R)OFMP) algorithm (see e. g. Berkel et al., 2011; Fischer, 2011; Fischer and Michel, 2012, 2013a,b; Gutting et al., 2017; Kontak, 2018; Kontak and Michel, 2018, 2019; Michel, 2015a; Michel and Orzlowski, 2017; Michel and Schneider, 2020; Michel and Telschow, 2014, 2016; Telschow, 2014). Note that there exists also a weak variant, the (Regularized) Weak Functional Matching Pursuit ((R)WFMP) algorithm. However, this version will not be of interest here because the weakness strategy contradicts the improvements by learning. If we refer to either the RFMP or the ROFMP algorithm of these methods, we use the abbreviation Inverse Problem Matching Pursuit (IPMP) algorithms in this thesis.
The IPMP algorithms originate from the Matching Pursuit (MP) algorithm devel-

oped by Mallat and Zhang (1993) and the Orthogonal Matching Pursuit (OMP) algorithm introduced by Vincent and Bengio (2002) and Pati et al. (1993) (the RWFMP algorithm also utilizes ideas from Temlyakov (2000)). The (O)MP algorithms were then further developed to the IPMP algorithms in order to handle inverse problems. The strategy of the IPMP algorithms is as follows: let $\mathcal{D}$ be a dictionary, i. e. a set of trial functions like spherical harmonics, Slepian functions and RBFs. In particular, $\mathcal{D}$ may contain global and local functions simultaneously. An IPMP algorithm iteratively chooses dictionary elements and corresponding real coefficients such that the Tikhonov functional of the current approximation is minimized. Hence, they produce a linear combination of a 'best basis' that approximates the solution of an inverse problem. In the (R)OFMP algorithm, the chosen dictionary elements additionally fulfil a certain orthogonality relation. In comparison to the previously mentioned approaches, the IPMP algorithms are able to automatically construct global parts of the signal (by choosing global trial functions) and local ones (by choosing rather local trial functions). Furthermore, their particular advantages are an increased accuracy in comparison to traditional methods like for instance spline approximation and an improved interpretability due to the mixture of different types of trial functions. Moreover, the algorithms are stable because choosing a dictionary element does not involve inverting a matrix or solving a large system of linear equations. Additionally, working with trial functions yields a non-discretized result. At last, they are also capable of a joint-inversion of multiple-source data. The IPMP algorithms have already been successfully applied to several tasks like the downward continuation, the (linear and non-linear) inverse gravimetric problem as well as the inversion of magnetoencephalography (MEG) or electroencephalography (EEG) measurements (see e. g. Fischer, 2011; Fischer and Michel, 2012, 2013b; Kontak, 2018; Kontak and Michel, 2018; Michel and Telschow, 2014; Leweke, 2018; Leweke et al., 2018b; Telschow, 2014). In some of these publications, the algorithms have also been compared to traditional methods such as spline approximation and proved to be numerically competitive. All in all, an IPMP algorithm represents a simple and stable method that aims to combine the advantages of different types of trial functions.
In the previous summary of the IPMP algorithms, we did not discuss the dictionary $\mathcal{D}$ in further details. However, for the best basis, only dictionary elements can be used. Thus, it depends on the dictionary how well a computed approximation can be. The IPMP algorithms as previously published can only work with a finite dictionary. How can we be sure to choose a-priori a finite dictionary from infinitely many trial functions such that we obtain a satisfying approximation? In other words, how can we prevent the approximation to be (too) biased by the dictionary? For previous publications, usually, a finite dictionary was chosen in a trial-and-error or rule-of-thumb fashion. Hence, the need to investigate how the IPMP algorithms can be made independently of an a-priori manual choice of dictionary occurs. Moreover, the manually chosen dictionaries usually needed to be large such that a competitive approximation is obtained. This approximation is, however, represented by a relatively small subset of the dictionary in most cases.

In particular for an increasing number of data points, a large dictionary leads to a very high storage demand and a long CPU-runtime. Hence, in order to be competitive with respect to memory capacity and runtime, the IPMP algorithms need to be further developed for a use in large-scale experiments.
The (O)MP algorithms are also based on a dictionary and, thus, faced the same questions. This launched the development of dictionary learning strategies. Summarized, these strategies aim to determine an approximation and a sparse dictionary parallelly. The usual procedure is to alternate between improving the dictionary and computing the respective next (better) approximation. Prominent examples of these methods are the K-SVD and the MOD algorithm. For more details, see, for instance, Bruckstein et al. (2009); Engan et al. (1999a,b); Rubinstein et al. (2010). However, these approaches cannot be taken over to the IPMP algorithms straight-forwardly because of strategic aims of the methods and mathematical differences in the underlying tasks. The (O)MP algorithms aim to solve discretized approximation problems. Then the dictionary can be interpreted as a matrix and learning a dictionary means modifying the matrix entries. When it comes to the IPMP algorithms, we do not want to abandon their continuous output for a discretized one. Moreover, constructing new dictionary elements which are tailored for a specific task is also not favoured in our case because the comparability with traditional models might be lost. Furthermore, if RBFs are included in the dictionary, it was shown in previous publications that the IPMP algorithms allow for a multiscale expansion and, thus, a multiresolution analysis (see e. g. Fischer, 2011; Fischer and Michel, 2012, 2013a,b; Michel and Telschow, 2014; Telschow, 2014). That means, using well-known trial functions, the IPMP algorithms are able to reveal hidden detail structures. Most important, however, is the difference in the mathematical task which needs to be solved. The (O)MP algorithms consider pure interpolation / approximation problems. The IPMP algorithms are able to solve (the more general) ill-posed inverse problem. That means, the "dictionary matrix" is now a matrix of discretized values of dictionary elements applied in an operator, e. g. the upward operator $\mathcal{T}$ from (1.1). Thus, using traditional dictionary learning approaches like the K-SVD or MOD algorithm only yields optimized values of, e. g., upward continued dictionary elements. Determining the dictionary elements themselves leaves us, again, in need of solving ill-posed inverse problems. At last, if we work with a non-discretized approach and established trial functions, we are also able to use well-known singular value decompositions of a given inverse problem.
The dictionary learning given by Prünte (2008) follows a more theoretical approach. Similar to common approaches in machine learning, it considers the optimization of a quality measure in order to learn a dictionary. Even though the number of different trial functions considered there is smaller than we would like to use, using an error function seems promising because the IPMP algorithms themselves already consider a possible measure with the Tikhonov functional.
Hence, the problem at hand is to develop a novel strategy to automatize the determination of a dictionary for the IPMP algorithms. For this, we develop the

Learning Inverse Problem Matching Pursuit (LIPMP) algorithms in this thesis. In particular, we introduce the Learning Regularized Functional Matching Pursuit (LRFMP) and the Learning Regularized Orthogonal Matching Pursuit (LROFMP) algorithm. These methods enhance the original IPMP algorithms by the following approach. In an iteration of an IPMP algorithm, the next dictionary element is chosen such that it minimizes the Tikhonov functional of the current approximation. For practical purposes, this minimization can be exchanged with the maximization of a different objective function. Using a finite dictionary, the objective function can be evaluated at discrete points and their values can be compared. The dictionary element that yields the maximal value is chosen in the current iteration. In order to overcome the use of a finite dictionary, we now model nonlinear constrained optimization problems for maximizing the objective function. In this way, we are able to consider infinitely many possible RBFs and Slepian functions. Learning spherical harmonics is essentially learning a maximally suitable degree. The learnt maximal degree follows straight-forwardly from the characteristic behaviour of the IPMP algorithms which is inherited by the LIPMP algorithms as the remaining routine of the IPMP algorithm is taken over to the LIPMP algorithms. Summarized, the learning technique computes a finite number of new candidates in each iteration by solving continuous optimization problems where possible. This set then works again as a finite dictionary.
Thus, the advantage of the LIPMP algorithms is that they are able to work with an infinite dictionary and, in this way, resolve the need to manually choose a finite dictionary a-priori. Furthermore, the best basis chosen by the LIPMP algorithm represents a learnt dictionary which can be used in an IPMP algorithm. By building the LIPMP algorithms closely to their non-learning predecessors, we will see that they inherit the existing convergence results. Our numerical results show the applicability of this approach with respect to the downward continuation of satellite data.
The thesis is divided into five parts which cover the mathematical basics, the development of the learning algorithms, numerical experiments, a summary and a technical appendix.
The first part consists of three chapters that summarize the mathematical problem specification, the trial functions under consideration and the IPMP algorithms. Chapter 2 introduces polynomials on a sphere which are necessary to model the gravitational potential afterwards. Next, the mathematical description of an exterior Dirichlet problem follows as this models the related direct problem of the upward continuation and, thus, also the inverse downward continuation. At last, we generalize the task at hand one more step and summarize the main aspects of inverse problems: ill-posedness, the singular value decomposition of linear and compact operators, regularization strategies and the choice of a regularization parameter. In Chapter 3, we introduce further trial functions which will be used as dictionary elements here. In particular, we discuss scalar Slepian functions as well as radial basis low and band pass filters. For the latter ones, we first summarize aspects of spherical Sobolev spaces. We close the chapter with an overview

on the values of certain inner products as well as upward continued values of the presented trial functions. As a last chapter in this part, we state the algorithmic approaches of the matching pursuits in Chapter 4. In particular, we summarize the operator-free variants as well as the IPMP algorithms. Additionally, we have a closer look at the notation of the dictionary and summarize open questions with respect to the IPMP algorithms.
The second part consists of four chapters that present a motivation for dictionary learning, its theoretical formulation, the development of the LIPMP algorithms (with a detailed look at its formulation in the case of the downward continuation of satellite data) and its theoretical analysis. We start the part with a general view on learning itself and, in particular, machine and dictionary learning in Chapter 5. Then we summarize in Chapter 6 previous approaches of dictionary learning, outline the differences to our situation and close with a novel characterization of desirable dictionaries. This enables us to develop the LIPMP algorithms in Chapter 7. We present their main structure, formulate the newly risen optimization problems in general as well as for the downward continuation and summarize additional features. We close the chapter by giving pseudo-codes for the LIPMP algorithms. In Chapter 8, we end this part by considering the theoretical aspects of the novel algorithms. This includes discussing convergence results as well as evaluating the learning process.
In the third part, we present numerical experiments with the LIPMP algorithms. Hence, in Chapter 9, we first state a general test setting and introduce the used data from the EGM2008 and the GRACE mission. In Chapter 10, we consider the applicability of the learnt dictionary and compare it to the results obtained using a manually chosen dictionary. We consider this comparison for the downward continuation of both data origins presented in Chapter 9. We also include first and previously published results using less trial functions and a smaller version of the LRFMP algorithm. At last, we present a, from the LRFMP algorithm, learnt GRACE (year-)dictionary and apply it to unseen test data. In Chapter 11, we then consider the LIPMP algorithms as standalone approximation methods. In particular, we show results for the approximation of surface data as well as the downward continuation from a regular and irregular grid. We close the presentation of our numerical results with a test on synthetic data to show that the LROFMP algorithm is able to distinguish between global trends and local anomalies.
The fourth part contains a conclusion and an outlook of this work. At last, the technical appendix deals with aspects for practical purposes (Appendix A) and a detailed documentation (Appendix B) of the implementation of an (L)IPMP algorithm. In particular, Appendix A summarizes point grids, Legendre polynomials, associated Legendre functions, fully normalized spherical harmonics as well as an overview of optimization algorithms used in our experiments.

Part I.

Preparatory Work

2. From geodesy to inverse problems

We start with a mathematical view on the determination of the gravitational potential. For this, we first summarize some basic notation and results. Then we introduce spherical harmonics which are used, for instance, in one representation of the potential. Next, we shortly derive from Newton's law of gravitation different representations of the gravitational potential of the Earth. This allows us to formulate the potential also on a satellite orbit. The use of satellite data, however, turns the task of determining the potential on the surface of the Earth into a spherical inverse problem. Such problems will be discussed at the end of this chapter.

2.1. Preliminaries

First, we define a common notational language and recall some fundamental theorems in agreement with Fischer (2010); Heuser (2001, 2004, 2006); Königsberger (2004a,b); Michel (2013); Yosida (1995).
The common sets of numbers are denoted as follows:

$\mathbb{Z}$	integers,
$\mathbb{N}$	positive integers,
$\mathbb{N}_0$	non-negative integers,
$\mathbb{R}$	real numbers,
$\mathbb{R}^+$	positive real numbers,
$\mathbb{R}_0^+$	non-negative real numbers,
$\mathbb{C}$	complex numbers,
$\mathbb{R}^d$	d-dimensional vector space of real numbers,
$\mathbb{R}^{r\times s}$	matrices with r rows and s columns and real coefficients.

Further, we use the common quantifiers $\forall$, i. e. "for all", and $\exists$, i. e. "there exists". If we consider the j-th component of the i-th vector $x^{(i)} \in \mathbb{R}^d$, we write

$$x_j^{(i)}.$$

Further, we use $x \cdot y$ with $x, y \in \mathbb{R}^3$ for the Euclidean inner product of two vectors. For two indices x and y, the symbol $\delta_{x,y}$ stands for the Kronecker delta:

$$\delta_{x,y} := \begin{cases} 1, & x = y, \\ 0, & x \neq y. \end{cases}$$

In this thesis, the application under investigation will be a spherical inverse problem like the downward continuation of the gravitational potential of the Earth. For this, we introduce an abbreviation for the sphere which is commonly used within geomathematics (see e. g. Freeden et al., 1998; Freeden and Gutting, 2013; Freeden and Schreiner, 2009; Michel, 2013). The sphere of radius $a \in \mathbb{R}^+$ and centre 0 is denoted by

$$\Omega_a := \left\{ x \in \mathbb{R}^3 \,\middle|\, |x| = a \right\}.$$

In particular, we set $\Omega := \Omega_1$. The ball of radius $a \in \mathbb{R}^+$ and centre 0 is given by

$$\mathbb{B}_a := \left\{ x \in \mathbb{R}^3 \,\middle|\, |x| \leq a \right\}.$$

Again, we set $\mathbb{B} := \mathbb{B}_1$. Further, we denote the inner of a set D as $\mathring{D}$ and the boundary as ∂D.

Some function spaces The unit sphere Ω will be of interest as the domain of scalar real-valued functions $f\colon \Omega \to \mathbb{R}$, also called (scalar) spherical functions.
In general, we consider the following function spaces. The set of k-times continuously differentiable real functions on $\emptyset \neq D \subseteq \mathbb{R}^d$ compact is denoted by $\mathrm{C}^{(k)}(D) := \mathrm{C}^{(k)}(D, \mathbb{R})$ for $k \in \mathbb{N}_0$.The space can be equipped with the maximum norm

$$\|f\|_\infty := \max_{x \in D} |f(x)|.$$

Then, as D is presumably compact, $(\mathrm{C}^{(0)}(D), \|\cdot\|_\infty)$ is a Banach space (see e. g. Heuser, 2006, Example 9.7). Further, for $p \in [1, \infty[$, we set

$$\mathcal{L}^p(D) := \mathcal{L}^p(D, \mathbb{R}) := \left\{ f\colon D \to \mathbb{R} \,\middle|\, D \subseteq \mathbb{R}^d,\ \int_D |f(x)|^p \,\mathrm{d}x < \infty \right\}.$$

This is not a normed space yet as the Lebesgue integral vanishes on sets with zero measure which yields some ambiguity. Thus, to obtain a normed space, we consider the space

$$\mathcal{N}^p(D) := \mathcal{N}^p(D, \mathbb{R}) := \left\{ f\colon D \to \mathbb{R} \,\middle|\, D \subseteq \mathbb{R}^d,\ \int_D |f(x)|^p \,\mathrm{d}x = 0 \right\}.$$

Then the space

$$\mathrm{L}^p(D) := \mathcal{L}^p(D) / \mathcal{N}^p(D)$$

is a Banach space with the norm

$$\|f\|_{\mathrm{L}^p(D)} := \sqrt[p]{\int_D |f(x)|^p \,\mathrm{d}x}.$$

Moreover, for $p = 2$, the space $\mathrm{L}^2(D)$ is a Hilbert space with the inner product

$$\langle f, g \rangle_{\mathrm{L}^2(D)} := \int_D f(x) g(x) \, \mathrm{d}x$$

(see e. g. Heuser, 2006, Examples 9.11 and 18.5). A function $f \in \mathrm{L}^2(D)$ is also called square-integrable because the respective norm is given by

$$\|f\|_{\mathrm{L}^2(D)} := \sqrt{\int_D |f(x)|^2 \, \mathrm{d}x} < \infty.$$

Further aspects of a spherical geometry In general, a point x in $\mathbb{R}^3$ is characterized by spherical coordinates $r \in \mathbb{R}_0^+$, $\varphi \in [0, 2\pi[$ and $\theta \in [0, \pi]$. Note that the latter one can be exchanged by $t = \cos(\theta) \in [-1, 1]$. Then the point x is given by

$$x(r, \varphi, t) := \begin{pmatrix} r\sqrt{1-t^2}\cos(\varphi) \\ r\sqrt{1-t^2}\sin(\varphi) \\ rt \end{pmatrix} \quad \text{or} \quad x(r, \varphi, \theta) := \begin{pmatrix} r\sin(\theta)\cos(\varphi) \\ r\sin(\theta)\sin(\varphi) \\ r\cos(\theta) \end{pmatrix}, \tag{2.1}$$

respectively. Therefore, we immediately obtain a local orthonormal basis of $\mathbb{R}^3$ with

$$\varepsilon^r(\varphi, t) := \begin{pmatrix} \sin(\theta)\cos(\varphi) \\ \sin(\theta)\sin(\varphi) \\ \cos(\theta) \end{pmatrix}, \tag{2.2}$$

$$\varepsilon^\varphi(\varphi, \theta) := \begin{pmatrix} -\sin(\varphi) \\ \cos(\varphi) \\ 0 \end{pmatrix} \tag{2.3}$$

and

$$\varepsilon^t(\varphi, \theta) := \begin{pmatrix} -\cos(\theta)\cos(\varphi) \\ -\cos(\theta)\sin(\varphi) \\ \sin(\theta) \end{pmatrix}. \tag{2.4}$$

If we compare (2.1) and (2.2), we observe that ε^r is the outer normal unit vector of the unit sphere Ω. Thus, ε^φ and ε^t must be tangential vectors. In fact, ε^φ gives a longitudinal direction and ε^t a latitudinal one. An illustration can be found, for instance, in Michel (2013, p. 86).
This local orthonormal basis enables a spherical view on the gradient. We have

$$\nabla = \varepsilon^r \frac{\partial}{\partial r} + \frac{1}{r}\nabla^* = \varepsilon^r \frac{\partial}{\partial r} + \frac{1}{r}\left(\varepsilon^\varphi \frac{1}{\sqrt{1-t^2}}\frac{\partial}{\partial \varphi} + \varepsilon^t \sqrt{1-t^2}\frac{\partial}{\partial t}\right). \tag{2.5}$$

Further, remember that a function $f\colon \mathbb{R}^3 \to \mathbb{R}$ is called harmonic if it holds

$$\Delta f = (\nabla \cdot \nabla) f = 0.$$

If we emphasize the derivation argument of the Laplace operator Δ, we write

$$\Delta_x f = (\nabla_x \cdot \nabla_x) f = \sum_{i=0}^{3} \frac{\partial^2 f}{\partial x_i^2}.$$

Some functional analysis The local orthonormal basis of $\mathbb{R}^3$ has 3 elements. In spaces like $L^2(\Omega)$, a basis must be infinite. For these cases, we recall some fundamental relations from functional analysis (see e. g. Heuser, 2006) or (Michel, 2013, p. 22).

Theorem 2.1.1. *For a Hilbert space $(H, \langle\cdot,\cdot\rangle_H)$ and a countable orthonormal system $\{g_n\}_{n\in\mathbb{N}_0}$ in H, it is equivalent that*

(i) the system $\{g_n\}_{n\in\mathbb{N}_0}$ is complete,

(ii) every element $f \in H$ has a representation as a Fourier series in $\{g_n\}_{n\in\mathbb{N}_0}$, i. e.

$$\lim_{N\to\infty} \left\| f - \sum_{n=0}^{N} \langle f, g_n\rangle_H g_n \right\|_H = 0$$

for the induced norm $\|\cdot\|_H$,

(iii) the Parseval identity holds, i. e.

$$\langle f, h\rangle_H = \sum_{n=0}^{\infty} \langle f, g_n\rangle_H \langle h, g_n\rangle_H,$$

(iv) the system $\{g_n\}_{n\in\mathbb{N}_0}$ is a basis, i. e.

$$\overline{\operatorname{span}\{g_n \mid n \in \mathbb{N}_0\}}^{\|\cdot\|_H} = H.$$

Note that, for property (iv), we also say that $\{g_n\}_{n\in\mathbb{N}_0}$ is closed in H in the sense of the approximation theory.

Thus, we have a representation of each $f \in L^2(\Omega)$ as a Fourier series by

$$f = \sum_{n=0}^{\infty} \langle f, g_n\rangle_{L^2(\Omega)} g_n$$

for a complete and countable orthonormal system $\{g_n\}_{n\in\mathbb{N}_0}$ of $L^2(\Omega)$. Note that this equality holds in the sense of $(L^2(\Omega), \langle\cdot,\cdot\rangle_{L^2(\Omega)})$. The term $\langle f, g_n\rangle_{L^2(\Omega)}$ is called a Fourier coefficient.
Next, an operator is a mapping

$$T\colon X \to Y$$

between two normed spaces $(X, \|\cdot\|_X)$ and $(Y, \|\cdot\|_Y)$. If $Y = \mathbb{R}$, then the mapping is called a functional. Dependent on the norms of X and Y, the continuity of the operator is defined as usual. An operator is called linear if

$$T(\alpha x_1 + \beta x_2) = \alpha T x_1 + \beta T x_2$$

for $\alpha,\ \beta \in \mathbb{R}$ and $x_1,\ x_2 \in X$. The operator is bounded if

$$\sup_{x\in X,\ \|x\|_X=1} \|Tx\|_Y < \infty.$$

A linear operator is continuous if and only if it is bounded. A simple example of a linear and bounded operator is a matrix $A \in \mathbb{R}^{m\times n}$, $m,\ n \in \mathbb{N}$, with $A\colon \mathbb{R}^n \to \mathbb{R}^m$ and the common matrix-vector-multiplication $Ax = y,\ x \in \mathbb{R}^n,\ y \in \mathbb{R}^m$.
At last, we summarize some properties of matrices $A(c) \in \mathbb{R}^{n\times n}$ that are dependent on a scalar $c \in \mathbb{R}$. In particular, we are interested in their determinant. Recall that, for an eigenvector $\rho(c)$ of $A(c)$, it holds

$$0 = \det\left(A(c) - \rho(c)I\right), \tag{2.6}$$

where I is the identity matrix of the same size as $A(c)$. The derivative of a determinant $\det B(\nu)$ with respect to $\nu \in \mathbb{R}$ and for a parametrized matrix $B \in \mathbb{R}^{n\times n}$ is obtained via

$$\frac{\mathrm{d}}{\mathrm{d}\nu}\det B(\nu) = \sum_{j=1}^{n}\sum_{k=1}^{n}(-1)^{j+k}\left(\frac{\mathrm{d}}{\mathrm{d}\nu}\left(B(\nu)\right)_{j,k}\right)\det\left(B(\nu)\right)^{j,k}, \tag{2.7}$$

where $\left(B(\nu)\right)_{j,k}$ denotes the (j,k)-th component of $B(\nu)$ and $\left(B(\nu)\right)^{j,k}$ stands for the $(n-1)\times(n-1)$-matrix that is obtained when removing the j-th row and k-th column from $B(\nu)$. Equation (2.7) can be derived using linear algebra. In the special case that $B(\nu)$ is tridiagonal and symmetric, the determinant itself can also be determined via a three term recursion:

$$\begin{aligned} s_0 &= 1, \\ s_1 &= \left(B(\nu)\right)_{1,1}, \\ s_i &= \left(B(\nu)\right)_{i,i}s_{i-1} - \left(B(\nu)\right)^2_{i-1,i-1}s_{i-2},\ i = 2,\ldots,n, \\ \det B(\nu) &= s_n, \end{aligned} \tag{2.8}$$

confer, e. g., Muir (1882, Chapter 3, Continuants).

2.2. Some aspects about polynomials on the sphere

Next, we introduce an example of a complete and countable orthonormal system in $\mathrm{L}^2(\Omega)$. We will use this system in the formulation of the gravitational potential as well as for several aspects of our learning algorithm.
An intuitive approach to construct a complete and countable orthonormal system is based on polynomials. The corresponding polynomials on the sphere are the so-called spherical harmonics. They are defined to be homogeneous and harmonic (for the latter see Section 2.1) in the following way. This section is based on Freeden et al. (1998); Freeden and Gerhards (2013); Freeden and Gutting (2013); Freeden and Michel (2004a); Freeden and Schreiner (2009); Michel (2013); Müller (1966).

Definition 2.2.1. *Let $n \in \mathbb{N}_0$ be an integer. A polynomial $p\colon \mathbb{R}^3 \to \mathbb{R}$ of degree n is homogeneous if*

$$p(x) = \sum_{|\nu|=n} C_\nu x^\nu,$$
$$C_\nu \in \mathbb{R}, \quad \nu \in \mathbb{N}_0^3, \quad |\nu| := \nu_1 + \nu_2 + \nu_3, \quad x^\nu := x_1^{\nu_1} x_2^{\nu_2} x_3^{\nu_3}.$$

The set of all homogeneous and harmonic polynomials $p\colon \mathbb{R}^3 \to \mathbb{R}$ of degree n is called $\mathrm{Harm}_n\left(\mathbb{R}^3\right)$. *The set* $\mathrm{Harm}_n(\Omega)$ *is defined as the set of all homogeneous and harmonic polynomials of degree n on $\mathbb{R}^3$ which are restricted to the sphere Ω, i. e.*

$$\mathrm{Harm}_n(\Omega) := \left\{ p_{|\Omega} \,\middle|\, p \in \mathrm{Harm}_n\left(\mathbb{R}^3\right),\ p_{|\Omega}\colon \Omega \to \mathbb{R},\ p_{|\Omega}(\eta) = p(\eta),\ \eta \in \Omega \right\}.$$

The elements of $\mathrm{Harm}_n(\Omega)$ *are called (scalar) spherical harmonics of degree n and are denoted by Y_n.*

In the common literature on spherical harmonics mentioned above, we also find the following result (see e. g. Freeden and Gutting, 2013, p. 139).

Theorem 2.2.2. *For an integer $n \in \mathbb{N}_0$, the dimension of the spaces* $\mathrm{Harm}_n\left(\mathbb{R}^3\right)$ *and* $\mathrm{Harm}_n(\Omega)$, *respectively, is*

$$\dim \mathrm{Harm}_n\left(\mathbb{R}^3\right) = \dim \mathrm{Harm}_n(\Omega) = 2n + 1. \tag{2.9}$$

Thus, we can define an orthonormal system in $\mathrm{Harm}_n(\Omega)$.

Definition 2.2.3. *For $n \in \mathbb{N}_0$, the system*

$$\left\{ Y_{n,j} \in \mathrm{Harm}_n(\Omega) \right\}_{j=-n,\dots,n}$$

denotes an orthonormal system in $\mathrm{Harm}_n(\Omega)$, *i. e. for*

$$Y_{n,k_1},\ Y_{n,k_2} \in \left\{ Y_{n,j} \in \mathrm{Harm}_n(\Omega) \right\}_{j=-n,\dots,n},$$

it holds

$$\langle Y_{n,k_1}, Y_{n,k_2} \rangle_{\mathrm{L}^2(\Omega)} = \delta_{k_1,k_2}. \tag{2.10}$$

We call n the degree and j the order of the spherical harmonic $Y_{n,j}$.

Due to (2.9), the orthonormal system

$$\left\{ Y_{n,j} \in \mathrm{Harm}_n(\Omega) \right\}_{j=-n,\dots,n}$$

is a basis of $\mathrm{Harm}_n(\Omega)$. Note that only spherical harmonic basis functions have two indices.
For spherical harmonics of a fixed degree, the following prominent result, see e. g. Freeden and Schreiner (2009, p. 82) or Michel (2013, pp. 103-107), will be used at several points in our learning approach.

Theorem 2.2.4. ***(Addition theorem for spherical harmonics)*** *For an orthonormal system* $\{Y_{n,j}\}_{j=-n,\dots,n}$, $n \in \mathbb{N}_0$, *it holds*

$$\sum_{j=-n}^{n} Y_{n,j}(\eta) Y_{n,j}(\chi) = \frac{2n+1}{4\pi} P_n(\eta \cdot \chi), \qquad \forall\, \eta, \chi \in \Omega,$$

where P_n *denotes the n-th Legendre polynomial.*

For the theory of orthogonal polynomials such as the Legendre polynomials, we refer the reader, for instance, to Abramowitz and Stegun (1972); Freeden and Gutting (2013); Magnus et al. (1966); Michel (2013); Szegö (1975).
Let us now consider spherical harmonics of distinct degrees. We first note that an orthogonality relation can be shown also in this case (see e. g. Freeden et al., 1998, p. 36).

Lemma 2.2.5. *For* n *and* $m \in \mathbb{N}_0$, *each two spherical harmonics* $Y_n \in \mathrm{Harm}_n(\Omega)$ *and* $Y_m \in \mathrm{Harm}_m(\Omega)$ *are orthogonal with respect to the* $\mathrm{L}^2(\Omega)$*-inner product, i. e.*

$$\langle Y_n, Y_m \rangle_{\mathrm{L}^2(\Omega)} = 0, \qquad n \neq m. \tag{2.11}$$

Thus, together with (2.10), it holds

$$\langle Y_{n,j}, Y_{m,k} \rangle_{\mathrm{L}^2(\Omega)} = \delta_{n,m} \delta_{j,k} \tag{2.12}$$

for $n,\ m,\ j,\ k \in \mathbb{N}_0$ with $-n \leq j \leq n, -m \leq k \leq m$. Furthermore, spherical harmonics of distinct degrees are linearly independent. Therefore, we consider their union.

Definition 2.2.6. *For* $N \in \mathbb{N}_0$, *we define the space*

$$\mathrm{Harm}_{0,\dots,N}(\Omega) := \bigoplus_{n=0}^{N} \mathrm{Harm}_n(\Omega).$$

Further, we set

$$\mathrm{Harm}_{0,\dots,\infty}(\Omega) := \bigcup_{n=0}^{\infty} \mathrm{Harm}_{0,\dots,N}(\Omega).$$

Due to (2.9), Definition 2.2.3 and (2.11), we have the orthonormal basis

$$\bigcup_{n=0}^{N} \left\{ Y_{n,j} \in \mathrm{Harm}_n(\Omega) \right\}_{j=-n,\dots,n}$$

of $\mathrm{Harm}_{0,\dots,N}(\Omega)$ for $N \in \mathbb{N}_0$. Note that for every $N \in \mathbb{N}_0$, it holds

$$\dim \mathrm{Harm}_{0,\dots,N}(\Omega) = (N+1)^2$$

due to (2.9). Thus, for $\mathrm{Harm}_{0,\ldots,\infty}(\Omega)$, we have

$$\bigcup_{n=0}^{\infty} \left\{ Y_{n,j} \in \mathrm{Harm}_n(\Omega) \right\}_{j=-n,\ldots,n}$$

as a basis of $\mathrm{Harm}_{0,\ldots,\infty}(\Omega)$. It can be shown that this system is also a basis of $\mathrm{L}^2(\Omega)$.

Theorem 2.2.7. *In the sense of the approximation theory, any orthonormal system of spherical harmonics*

$$\left\{ Y_{n,j} \in \mathrm{Harm}_n(\Omega) \right\}_{n\in\mathbb{N}_0;\, j=-n,\ldots,n}$$

is closed in $(\mathrm{L}^2(\Omega), \|\cdot\|_{\mathrm{L}^2(\Omega)})$.

For the lengthy proof, see, for instance, Michel (2013, pp. 112-122).

An example: Fully normalized spherical harmonics For practical purposes, we consider the fully normalized spherical harmonics. These functions are obtained by a separation ansatz for an orthonormal basis of $\mathrm{L}^2(\Omega)$ in spherical coordinates. Their construction in this way is for instance explained in Michel (2013, Section 5.2). At this point, we state the result similarly to Freeden and Gutting (2013, p.142).

Example 2.2.8. Let $n \in \mathbb{N}_0$ be a degree and $j \in \mathbb{N}_0$ with $-n \leq j \leq n$ an order. Further, we consider spherical coordinates $\varphi \in [0, 2\pi[$ and $t \in [-1, 1]$. Moreover, we use associated Legendre functions $P_{n,j}\colon [-1,1] \to \mathbb{R}$ given by

$$P_{n,j}(t) := \left(1 - t^2\right)^{j/2} \frac{\mathrm{d}^j}{\mathrm{d}t^j} P_n(t) \tag{2.13}$$

for $j \geq 0$ and where $P_n\colon [-1,1] \to \mathbb{R}$ is the n-th Legendre polynomial. Then we define the fully normalized spherical harmonics $Y_{n,j}\colon \Omega \to \mathbb{R}$ by

$$Y_{n,j}(\eta(\varphi,t)) := \sqrt{\frac{2n+1}{2}\frac{(n-|j|)!}{(n+|j|)!}} P_{n,|j|}(t) \frac{1}{\sqrt{2\pi}} \begin{cases} \sqrt{2}\cos(j\varphi), & j < 0, \\ 1, & j = 0, \\ \sqrt{2}\sin(j\varphi), & j > 0. \end{cases} \tag{2.14}$$

For aspects of the implementation of either the Legendre polynomials, the associated Legendre functions or the fully normalized spherical harmonics, we refer for instance to Fengler (2005, Technical Appendix).
Examples of fully normalized spherical harmonics are given in Figure 2.1. Blue colour stands for minimal values and red colour for maximal ones. The fully normalized spherical harmonics from degree 1 to degree 5 with non-negative order are depicted. Thus, from left to right, in the first row, the fully normalized

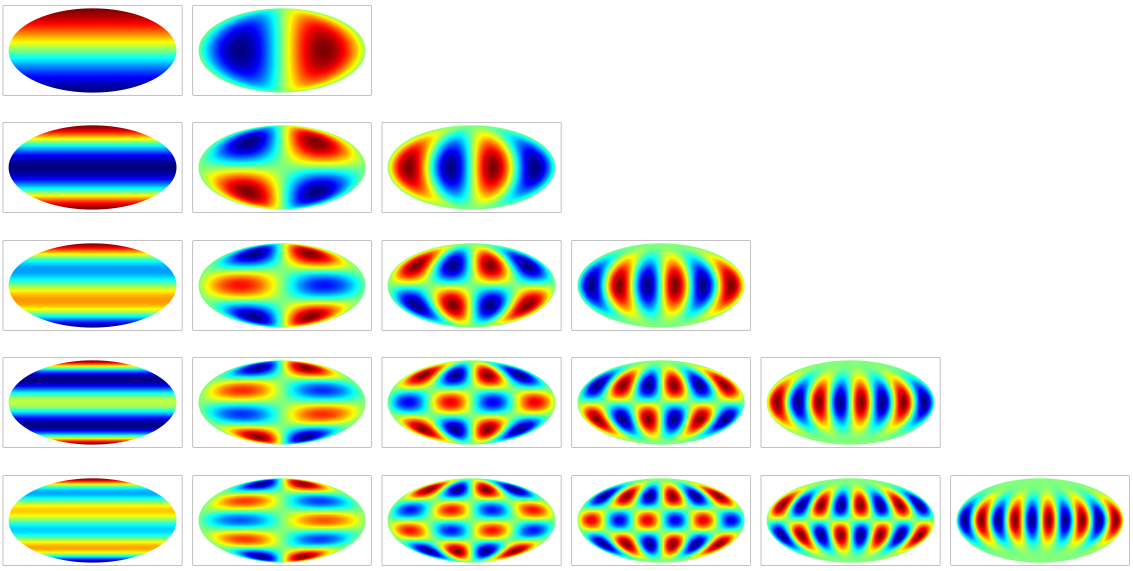


Figure 2.1.: Fully normalized spherical harmonics of degree 1 (first row), 2 (second row), 3 (third row), 4 (fourth row), 5 (fifth row). In each row all non-negative orders are presented in increasing order.

spherical harmonics $Y_{1,0}$ and $Y_{1,1}$ are given. The second row contains $Y_{2,0}$, $Y_{2,1}$ and $Y_{2,2}$. The third row shows $Y_{3,0}$, $Y_{3,1}$, $Y_{3,2}$ and $Y_{3,3}$. The fourth row depicts $Y_{4,0}$, $Y_{4,1}$, $Y_{4,2}$, $Y_{4,3}$ and $Y_{4,4}$. And the last row shows the functions $Y_{5,0}$, $Y_{5,1}$, $Y_{5,2}$, $Y_{5,3}$, $Y_{5,4}$ and $Y_{5,5}$. First of all, we notice that the fully normalized spherical harmonics indeed show a polynomial structure on the whole sphere and, thus, are global functions. Note that a fully normalized spherical harmonic is by construction perfectly localized in frequency because it represents exactly one degree and order. However, it is nowhere localized in space. Further, we see that, for a vanishing order ($j = 0$), the spherical harmonics have extrema only in the latitudinal and none in the longitudinal range (see the figures in the first column). In this case, they are called zonal harmonics. For equal degree and order ($|j| = n$), extrema are only observed in the longitudinal range (see the last figures in each row). These functions are called sectorial harmonics. At last, in all other cases ($0 \neq |j| \neq n$), there exist functions which attain several extrema in the latitudinal as well as the longitudinal range (see for instance the figure in the second row and the second column). These functions are called tesseral harmonics (see e. g. Freeden and Michel, 2004b).

For fully normalized spherical harmonics, we obtain the following properties.

Remark 2.2.9. Note that there exist also complex spherical harmonics (see e. g. Dahlen and Tromp, 1998; Freeden and Gutting, 2013). In general, let f be a real square-integrable function on the sphere. Its expansion in real spherical harmonics and in complex spherical harmonics is related by the Fourier coefficients in

the following way (see e. g. Dahlen and Tromp, 1998, pp. 858-859). For $n,\ j \in \mathbb{N}_0$ with $-n \leq j \leq n$, let ${}_cf^\wedge(n,j)$ denote a Fourier coefficient of f with respect to the complex basis of spherical harmonics and ${}_rf^\wedge(n,j)$ a Fourier coefficient of f with respect to the real basis of spherical harmonics. Then it holds

$$ {}_cf^\wedge(n,j) = \begin{cases} (-1)^j \frac{1}{\sqrt{2}} \left({}_rf^\wedge(n,j) + \mathrm{i}\ {}_rf^\wedge(n,-j) \right), & -n \leq j < 0, \\ {}_rf^\wedge(n,0), & j = 0, \\ \frac{1}{\sqrt{2}} \left({}_rf^\wedge(n,-j) - \mathrm{i}\ {}_rf^\wedge(n,j) \right), & 0 < j \leq n \end{cases} \tag{2.15} $$

and

$$ {}_rf^\wedge(n,j) = \begin{cases} \sqrt{2}\ \mathrm{Re}\left({}_cf^\wedge(n,|j|) \right), & -n \leq j < 0, \\ {}_cf^\wedge(n,0), & j = 0, \\ -\sqrt{2}\ \mathrm{Im}\left({}_cf^\wedge(n,j) \right), & 0 < j \leq n, \end{cases} \tag{2.16} $$

where i denotes the imaginary unit. In this thesis, we will mainly consider real spherical harmonics. If complex spherical harmonics are necessary at a certain point, we will emphasize that. Thus, we refer to the real fully normalized spherical harmonics and drop the "real" in the sequel.

Further, the derivative with respect to the longitude φ of a fully normalized spherical harmonic can be expressed as a spherical harmonic as well.

Lemma 2.2.10. *Let $n,\ j \in \mathbb{N}_0$ with $-n \leq j \leq n$ be fixed. Then it holds*

$$ \frac{\partial}{\partial \varphi} Y_{n,j}(\eta(\varphi,t)) = j Y_{n,-j}(\eta(\varphi,t)) \tag{2.17} $$

for the fully normalized spherical harmonic $Y_{n,j}$ of degree n and order j.

Proof. We start at the left-hand side and insert the definition of the fully normalized spherical harmonics:

$$ \begin{aligned} &\frac{\partial}{\partial \varphi} Y_{n,j}(\eta(\varphi,t)) \\ &\quad = \frac{\partial}{\partial \varphi} \left(\sqrt{\frac{2n+1}{2} \frac{(n-|j|)!}{(n+|j|)!}} P_{n,|j|}(t) \frac{1}{\sqrt{2\pi}} \begin{cases} \sqrt{2}\cos(j\varphi), & j < 0, \\ 1, & j = 0, \\ \sqrt{2}\sin(j\varphi), & j > 0 \end{cases} \right) \\ &\quad = \sqrt{\frac{2n+1}{2} \frac{(n-|j|)!}{(n+|j|)!}} P_{n,|j|}(t) \frac{1}{\sqrt{2\pi}} \begin{cases} \sqrt{2} \frac{\partial}{\partial \varphi} \left(\cos(j\varphi) \right), & j < 0, \\ \frac{\partial}{\partial \varphi} (1), & j = 0, \\ \sqrt{2} \frac{\partial}{\partial \varphi} \left(\sin(j\varphi) \right), & j > 0 \end{cases} \end{aligned} $$

$$= \sqrt{\frac{2n+1}{2}\frac{(n-|j|)!}{(n+|j|)!}} P_{n,|j|}(t) \frac{1}{\sqrt{2\pi}} \begin{cases} -\sqrt{2} j \sin(j\varphi), & j < 0, \\ 0, & j = 0, \\ \sqrt{2} j \cos(j\varphi), & j > 0 \end{cases}$$

$$= j\sqrt{\frac{2n+1}{2}\frac{(n-|j|)!}{(n+|j|)!}} P_{n,|j|}(t) \frac{1}{\sqrt{2\pi}} \begin{cases} \sqrt{2} \sin(-j\varphi), & j < 0, \\ 1, & j = 0, \\ \sqrt{2} \cos(-j\varphi), & j > 0 \end{cases}$$

$$= j Y_{n,-j}(\eta(\varphi, t)).$$

□

Outer harmonics In the geosciences, we are not only interested in functions on a sphere but also in functions outside a sphere. The spherical harmonics can be extended for this purpose in the following way (see e. g. Freeden and Schneider, 1998).

Definition 2.2.11. *Let $\{Y_{n,j}\}_{n\in\mathbb{N}_0;\, j=-n,\dots,n}$ be a basis system of spherical harmonics. Further, let $\mathbb{B}_a^{\text{ext}} := \mathbb{R}^3 \backslash \overline{\mathbb{B}_a}$ be the outer space of the ball $\mathbb{B}_a$ with radius $a \in \mathbb{R}^+$. We define the outer harmonics as follows:*

$$H_{-n-1,j}(a; x) := \frac{1}{a} \left(\frac{a}{|x|} \right)^{n+1} Y_{n,j} \left(\frac{x}{|x|} \right)$$

for $x \in \overline{\mathbb{B}_a^{\text{ext}}}$.

Theorem 2.2.12. *For $a \in \mathbb{R}^+$, $n \in \mathbb{N}_0$ and $j \in \{-n, \dots, n\}$, an outer harmonic $H_{-n-1,j}(a; \cdot)$ fulfils the following properties*

(i) $H_{-n-1,j}(a; \cdot) \in \mathrm{C}^{(\infty)}(\mathbb{B}_a^{\text{ext}})$,

(ii) $\Delta_x H_{-n-1,j}(a; x) = 0$ *for* $x \in \mathbb{B}_a^{\text{ext}}$,

(iii) $H_{-n-1,j}(a; \cdot)_{|\Omega_a} = a^{-1} Y_{n,j}$ *for* $\Omega_a = \partial \mathbb{B}_a$.

For another outer harmonic $H_{-m-1,k}(a; \cdot)$, $m \in \mathbb{N}_0$ and $k \in \{-m, \dots, m\}$, we have the relation $\langle H_{-n-1,j}(a; \cdot), H_{-m-1,k}(a; \cdot) \rangle_{\mathrm{L}^2(\Omega_a)} = \delta_{n,m} \delta_{j,k}$.

Thus, we have that $\{H_{-n-1,j}(a; \cdot)_{|\Omega_a}\}_{n\in\mathbb{N}_0;\, j=-n,\dots,n}$ for $a \in \mathbb{R}^+$ is a complete orthonormal system in $\mathrm{L}^2(\Omega_a)$.

2.3. A geodetic reference model: the gravitational potential

As a geophysical application, we consider the gravitational potential in more detail in this thesis. In this section, we will derive a formula for the potential from Newton's law of gravitation and show that it is a harmonic function. This section is based on Barthelmes (2014); Demtröder (2006); Halliday et al. (2007); Heuser (2004); Königsberger (2004b).

Newton's law of gravitation Newton (1687) formulated the law of gravitation. Let m_1 and m_2 be two point-shaped masses and r be the distance between them. Then the magnitude of the attracting force between m_1 and m_2 is given by

$$F = G\frac{m_1 m_2}{r^2}, \tag{2.18}$$

where G is the gravitational constant. Furthermore, the attracting force of m_2 that effects m_1 is given by the vectorial version of (2.18) as

$$\widehat{F} = Gm_1 m_2 \frac{x-y}{|x-y|^3},$$

where y denotes the centre of m_1 and x denotes the centre of m_2. Now, we would like to consider m_2 to be the Earth. Obviously, the Earth is not one mass particle but rather a body full of infinitely many mass particles in which every single one attracts m_1. Thus, the attracting force of the Earth is the sum over the attracting forces of each mass particle. Mathematically speaking, for the mass density ρ of the Earth, the attracting force is given by

$$\widehat{F}(y) = -Gm_1 \int_{\text{Earth}} \frac{\rho(x)(y-x)}{|x-y|^3}\,\mathrm{d}x = Gm_1 \int_{\text{Earth}} \frac{\rho(x)(x-y)}{|x-y|^3}\,\mathrm{d}x, \tag{2.19}$$

for any centre y of the mass m_1 outside the closed body of the Earth. The surface of the Earth is naturally a regular region, i. e. a bounded region whose boundary is an orientable piecewise smooth Lipschitzian manifold of dimension 2 (see Freeden and Gutting, 2013, Definition 6.1.1). Note that (2.19) is a volume integral as we substitute the mass m_2 by density times volume. From this attracting force, we obtain the acceleration $\hat{a}(y)$ of m_1 from the Earth due to a division by the mass m_1:

$$\hat{a}(y) = G \int_{\text{Earth}} \frac{\rho(x)(x-y)}{|x-y|^3}\,\mathrm{d}x. \tag{2.20}$$

This is independent of a second mass and, thus, describes the gravitational field of the Earth. The gravitational potential V is given in a straight-forward fashion by

$$\hat{a}(y) = \nabla V(y),$$

i. e. the acceleration is the gradient of the potential. Thus, we obtain the following well-known result which can be found, for instance, in Barthelmes (2014).

Theorem 2.3.1. *For the gravitational constant G, the gravitational potential V of the Earth is given by*

$$V(y) = G \int_{\text{Earth}} \frac{\rho(x)}{|x-y|}\,\mathrm{d}x + C, \tag{2.21}$$

where $C \in \mathbb{R}$ is constant and $y \in \mathbb{R}^3 \backslash \text{Earth}$.

Proof. The differentiation and integration can be interchanged in this case due to, e. g., Heuser (2004, Theorem 201.13). Then we have for $i = 1, 2, 3$,

$$\begin{aligned}\frac{\partial}{\partial y_i} V(y) &= \frac{\partial}{\partial y_i}\left(G \int_{\text{Earth}} \frac{\rho(x)}{|x-y|}\,\mathrm{d}x + C\right) = G \int_{\text{Earth}} \rho(x) \frac{\partial}{\partial y_i} |x-y|^{-1}\,\mathrm{d}x \\ &= G \int_{\text{Earth}} \rho(x) \left(-|x-y|^{-2}\right) \frac{\partial}{\partial y_i} |x-y|\,\mathrm{d}x \\ &= G \int_{\text{Earth}} \rho(x) \left(-\frac{1}{2}|x-y|^{-3}\right) \cdot (-2) \cdot (x_i - y_i)\,\mathrm{d}x \\ &= G \int_{\text{Earth}} \frac{\rho(x)(x_i - y_i)}{|x-y|^3}\,\mathrm{d}x.\end{aligned}$$

This equals the i-th component of (2.20). □

With a suitably chosen value of C, the gravitational potential is always positive and zero at infinity. Then its value equals the necessary energy to move the mass m_1 from its current point in space to infinity.
For a full model of the acceleration of a mass particle from the Earth, one has to take into account the centrifugal acceleration of this particle due to the rotation of the Earth as well. The centrifugal acceleration needs to be added and, in the same manner as we did above, this sum defines the gravity potential of the Earth. However, in this thesis, we will concentrate only on the gravitational potential of the Earth, i. e. we consider only the potential without any centrifugal force.
A potential like the gravitational potential or the gravity potential can be visualized in two ways. Either we consider its value on a given surface. Or we give one of its equipotential surfaces, that is, the surfaces where the potential attains only one value. A well-known equipotential surface of the gravity potential determines the geoid. This is the surface of the gravity potential that equals the surface of the undisturbed sea. It is used as a reference height for the real surface of the Earth in order to compute the heights of mountains and depths of the sea.

Harmonicity of the gravitational potential For further investigations of the gravitational potential, its harmonicity is of major importance. Thus, we state it at this point (see e. g. Heuser, 2004, Chapter 220).

Theorem 2.3.2. *The gravitational potential V of the Earth is harmonic outside the Earth, i. e.*

$$\Delta V(y) = 0,$$

$y \in \mathbb{R}^3 \backslash \overline{\text{Earth}}$.

Proof. We already mentioned in the proof of Theorem 2.3.1 that due to, e. g., Heuser (2004, Theorem 201.13) the differentiation and integration can be interchanged with respect to the gravitational potential V. This can also be done for a

second partial derivative. We obtain

$$\Delta_y V(y) = G \int_{\text{Earth}} \rho(x) \Delta_y \frac{1}{|x-y|} \, \mathrm{d}x$$

with the linearity of the integral. Thus, we consider the Laplace operator with respect to y of the function $|x-y|^{-1}$, x, $y \in \mathbb{R}^3$, $x \neq y$, in more detail:

$$\frac{\partial}{\partial y_i} |x-y|^{-1} = \left(-\frac{1}{|x-y|^2} \frac{\partial}{\partial y_i} |x-y| \right) = \left(-\frac{(-2) \cdot (x_i - y_i)}{2|x-y|^3} \right) = \frac{x_i - y_i}{|x-y|^3}.$$

Next, we obtain

$$\frac{\partial}{\partial y_i} \frac{x_i - y_i}{|x-y|^3} = \frac{-|x-y|^3 + 3|x-y|(x_i-y_i)^2}{|x-y|^6} = -\frac{1}{|x-y|^3} + 3\frac{(x_i-y_i)^2}{|x-y|^5}$$

and, thus, for the Laplace operator of $|x-y|^{-1}$

$$\begin{aligned} \Delta_y |x-y|^{-1} &= \sum_{i=1}^{3} \left(-\frac{1}{|x-y|^3} + 3\frac{(x_i-y_i)^2}{|x-y|^5} \right) \\ &= -\frac{3}{|x-y|^3} + \frac{3}{|x-y|^5} \sum_{i=1}^{3} (x_i-y_i)^2 = -\frac{3}{|x-y|^3} + \frac{3}{|x-y|^3} = 0. \end{aligned}$$

Hence, we have

$$\Delta V(y) = G \int_{\text{Earth}} 0 \, \mathrm{d}x = 0. \qquad \square$$

2.4. The exterior Dirichlet problem for satellite orbits

Next, we take a first step to embedding the geodetic task of determining the gravitational potential in a mathematical context. For this, we first discuss the exterior Dirichlet problem. Then we consider a more realistic setting of using satellite data instead of terrestrial ones. This section is based on Freeden and Michel (2004a); Freeden and Schneider (1998); Kellogg (1967); Telschow (2014).

Approximation on a surface: the exterior Dirichlet problem In Theorem 2.3.1, we derived a first representation of the gravitational potential outside the Earth as a volume integral. Due to the harmonicity and the regularity towards infinity (i. e. the potential decreases with increasing distance to the Earth and is zero in infinity), we can reformulate (2.21) to a surface integral as follows.
First, we make an assumption for the modelling of the problem in this thesis.

Remark 2.4.1. In this thesis, we identify the surface of the Earth with a sphere Ω_a of radius a.

Naturally, the gravitational potential is assumed to be continuous. Further, we assume that we know the values of the potential on the surface of the Earth. Then the determination of the potential outside of the Earth is a classical exterior Dirichlet problem.

Definition 2.4.2. *Let* $\mathbb{B}_a^{\text{ext}} := \mathbb{R}^3 \backslash \overline{\mathbb{B}_a}$ *denote the exterior of a closed ball* $\mathbb{B}_a$ *and let* $u \colon \overline{\mathbb{B}_a^{\text{ext}}} \to \mathbb{R}$ *be a function that fulfils*

(i) $u \in \mathrm{C}^{(0)}(\overline{\mathbb{B}_a^{\text{ext}}}) \cap \mathrm{C}^{(2)}(\mathbb{B}_a^{\text{ext}})$, *(continuity)*

(ii) $\Delta u = 0$ *in* $\mathbb{B}_a^{\text{ext}}$, *(harmonicity)*

(iii) $\limsup_{|y|\to\infty} |u(y)y| < \infty$ *as well as* $\limsup_{|y|\to\infty} |\nabla u(y)||y|^2 < \infty$. *(regularity at infinity)*

Further, let the boundary values of u at the boundary $\Omega_a = \partial\mathbb{B}_a$ *be given by a function* $f := u_{|\Omega_a}$. *Then the determination of u is called the exterior Dirichlet problem (EDP).*

The definition of the exterior Dirichlet problem can be found in many textbooks. We used a formulation similar to Telschow (2014, p. 33).

Theorem 2.4.3. *The EDP from Definition 2.4.2 has a unique solution. This solution* $u \colon \overline{\mathbb{B}_a^{\text{ext}}} \to \mathbb{R}$ *is given by*

$$\begin{aligned} u(y) &= \frac{1}{4\pi a} \int_{\Omega_a} \frac{f(\eta)(|y|^2 - a^2)}{|y-\eta|^3} \, \mathrm{d}\omega(\eta), && y \in \mathbb{B}_a^{\text{ext}}, \\ u(y) &= f(y), && y \in \Omega_a = \partial\mathbb{B}_a^{\text{ext}}. \end{aligned} \tag{2.22}$$

For a proof, see, for instance, Freeden and Michel (2004a, pp. 52-53). Thus, we obtained a representation of the gravitational potential V outside the Earth if we are given values of the potential on the boundary sphere Ω_a (here f). Note that, in comparison to (2.21), this representation of the solution is independent of the mass density ρ.

About the function space of the EDP solution We investigate the function space of EDP solutions.

Definition 2.4.4. *For* $\mathbb{B}_a^{\text{ext}} := \mathbb{R}^3 \backslash \overline{\mathbb{B}_a}$, *we define*

$$\mathrm{Pot}\left(\mathbb{B}_a^{\text{ext}}\right) := \left\{ u \;\middle|\; u \in \mathrm{C}^{(2)}\left(\mathbb{B}_a^{\text{ext}}\right),\ \Delta u(y) = 0 \ \forall\, y \in \mathbb{B}_a^{\text{ext}},\ \limsup_{y\to\infty} |u(y)y| < \infty,\ \limsup_{y\to\infty} |\nabla u(y)||y|^2 < \infty \right\}.$$

For $k \in \mathbb{N}_0$, *we set*

$$\mathrm{Pot}^{(k)}\left(\overline{\mathbb{B}_a^{\text{ext}}}\right) := \mathrm{Pot}\left(\mathbb{B}_a^{\text{ext}}\right) \cap \mathrm{C}^{(k)}\left(\overline{\mathbb{B}_a^{\text{ext}}}\right).$$

Then the EDP can be reformulated (see e. g. Freeden and Schneider, 1998): for a function $f \in \mathrm{C}^{(0)}(\Omega_a)$, determine

$$u \in \mathrm{Pot}^{(0)}\left(\overline{\mathbb{B}_a^{\mathrm{ext}}}\right)$$

such that

$$u_{\searrow\Omega_a}(x) := \lim_{\tau \searrow 0} u(x + \tau\nu(x)) = f(x)$$

for $x \in \Omega_a$ and $\nu(x)$ is the outer normal unit vector. Then the set of boundary values is given as follows.

Definition 2.4.5. *For a sphere Ω_a with radius $a \in \mathbb{R}^+$, we define the set of boundary values as*

$$\mathrm{B}_{\searrow\Omega_a} := \left\{ u_{\searrow\Omega_a} \,\middle|\, u \in \mathrm{Pot}^{(0)}\left(\overline{\mathbb{B}_a^{\mathrm{ext}}}\right) \right\}.$$

Due to the existence and uniqueness of the EDP solution, it holds

$$\mathrm{B}_{\searrow\Omega_a} = \mathrm{C}^{(0)}(\Omega_a)$$

and, thus,

$$\mathrm{L}^2(\Omega_a) = \overline{\mathrm{B}_{\searrow\Omega_a}}^{\|\cdot\|_{\mathrm{L}^2(\Omega_a)}} = \overline{\operatorname*{span}_{n\in\mathbb{N}_0;\, j=-n,\dots,n} \left\{ H_{-n-1,j}(a;\cdot)_{|\Omega_a} \right\}}^{\|\cdot\|_{\mathrm{L}^2(\Omega_a)}}$$

Hence, any function $f \in \mathrm{L}^2(\Omega_a)$ can be approximated by a solution of the EDP and any solution can be expanded in outer harmonics by

$$u(x) = \sum_{n=0}^{\infty} \sum_{j=-n}^{n} \left\langle f, H_{-n-1,j}(a;\cdot)_{|\Omega_a} \right\rangle_{\mathrm{L}^2(\Omega_a)} \left(\frac{a}{|x|}\right)^{n+1} H_{-n-1,j}(a;\cdot)_{|\Omega_a}\left(\frac{x}{|x|}\right) \tag{2.23}$$

for $x \in \mathbb{B}_a^{\mathrm{ext}}$. A further reason is given as follows: as the outer harmonics are indeed harmonic, the series on the right-hand side of (2.23) is harmonic as well for $|x| > a$. Due to the completeness of the outer harmonics in $\mathrm{L}^2(\Omega_a)$, we know that the series converges for $|x| = a$ to the Fourier series of f, see Theorem 2.1.1. Furthermore, the regularity at infinity is also fulfilled because we have $(a^n)/(|x|^{n+1}) \leq 1$ for all $x \in \overline{\mathbb{B}_a^{\mathrm{ext}}}$. Thus, the series representation is also a solution of the EDP. Due to the uniqueness of the solution of the EDP, (2.23) coincides with (2.22).
Note that the series expansion (as well as the integral representation) holds pointwise for $x \in \mathbb{B}_a^{\mathrm{ext}}$. However, if $x \in \Omega_a = \partial\mathbb{B}_a^{\mathrm{ext}}$, the series expansion holds only in the sense of $\mathrm{L}^2(\Omega_a)$ due to Theorem 2.1.1.

Using satellite data: the downward continuation problem If we have measurements of the gravitational potential on the surface of the Earth, we can compute $V(y)$ via the unique solution of the EDP for points y outside the Earth. An interpolation of these measurements yields the potential on the surface.
However, in practice, we are given satellite data: we have values of the potential above the Earth's surface. Such values are obtained by, e. g., GRACE (Gravity Recovery and Climate Experiment) and its successor GRACE-FO (Follow On) (see e. g. Flechtner et al., 2014a; NASA Jet Propulsion Laboratory, 2020; Schmidt et al., 2008; Tapley et al., 2004). Then the altitude of the satellite or the satellite orbit defines the radius b of a sphere Ω_b with $b > a$. We make another assumption for this thesis.

Remark 2.4.6. In this thesis, we assume that satellites have a constant altitude, i. e. the radius b of the satellite orbit Ω_b is constant. The theory as well as the algorithms are also suitable for varying altitudes as well. However, we neglect this to keep the notation better readable.

We can consider the EDP of the gravitational potential with respect to a satellite orbit in the same manner as we did above. Then we obtain a representation of the gravitational potential for every point outside a satellite orbit which depends only on the values of the potential on this orbit. However, the more interesting challenge is to compute the gravitational potential on the Earth's surface from values of the potential on a satellite orbit. That means, we are not so much interested in the values of the potential at points with greater distance to the Earth than the satellite but much more in the values of the potential at points with less (to no) distance to the Earth in comparison to a satellite. To obtain the gravitational potential at the surface of the Earth we need to continue its value from a satellite orbit downwards onto a sphere with smaller radius, i. e. to the inner of the sphere Ω_b. To do this, we first change the perspective.
Assume, we have the gravitational potential on the surface of the Earth Ω_a and we want to compute it on a satellite orbit Ω_b with $b > a$. This continuation of the potential upwards into space is defined by the solution of the EDP. For the gravitational potential V, we have the representation

$$V(y) = \frac{1}{4\pi a} \int_{\Omega_a} \frac{V(\eta)(|y|^2 - a^2)}{|y-\eta|^3} \, \mathrm{d}\omega(\eta)$$

for $y \in \mathbb{B}_a^{\mathrm{ext}} := \mathbb{R}^3 \backslash \overline{\mathbb{B}_a}$ and given boundary values $V(\eta)$, $\eta \in \Omega_a$. This can be written in short by means of an operator

$${}^a\mathcal{T}^b \colon \mathrm{B}_{\searrow \Omega_a} \to \mathrm{Pot}^{(0)}\left(\overline{\mathbb{B}_a^{\mathrm{ext}}}\right)$$

for $a < b$ as

$$V(\cdot)_{|\Omega_b} := {}^a\mathcal{T}^b V(\cdot)_{|\Omega_a},$$
$${}^a\mathcal{T}^b V(\cdot)_{|\Omega_a} = \frac{1}{4\pi a} \int_{\Omega_a} \frac{V(\eta)(|\cdot|^2 - a^2)}{|\cdot - \eta|^3} \, \mathrm{d}\omega(\eta)$$

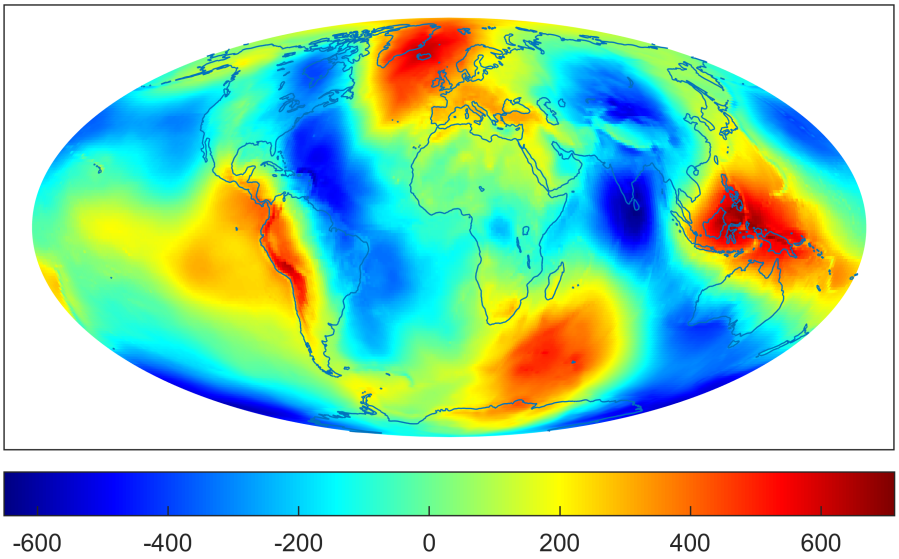


Figure 2.2.: The gravitational potential as given by the EGM2008, i. e. an expansion in fully normalized spherical harmonics from degree 3 up to degree 2190 and order 2159. We included the coastlines of the continents for a better visualization. All values in m^2/s^2.

or, with the use of relative satellite orbits,

$$V(\cdot)_{|\Omega_{b/a}} := \mathcal{T}V(\cdot)_{|\Omega} :=^1\mathcal{T}^{b/a}V(\cdot)_{|\Omega}, \tag{2.24}$$

$$^1\mathcal{T}^{b/a}V(\cdot)_{|\Omega} = \frac{1}{4\pi}\int_\Omega \frac{V(\eta)(|\cdot|^2-1)}{|\cdot-\eta|^3}\,\mathrm{d}\omega(\eta). \tag{2.25}$$

This is the integral representation of the upward continuation operator. We used a notation similar to, e. g., Telschow (2014, p. 62). With (2.23), we obtain a series representation of the upward continuation operator which is useful in practice. For further literature, see also, for instance, Michel and Fokas (2008) and the references therein.

Theorem 2.4.7. *In terms of spherical harmonics* $\{Y_{n,j}\}_{n\in\mathbb{N}_0;\, j=-n,\dots,n}$, *the upward continuation operator is given by*

$$(\mathcal{T}V)(\sigma\eta) = \sum_{n=0}^{\infty}\sum_{j=-n}^{n}\langle V, Y_{n,j}\rangle_{\mathrm{L}^2(\Omega)}\sigma^{-n-1}Y_{n,j}(\eta), \qquad \sigma>1,\ \eta\in\Omega. \tag{2.26}$$

Note that σ refers to the satellite orbit b/a. For practical purposes, we use fully normalized spherical harmonics in the representation (2.26) of the upward continuation operator. This system is also used by the EGM Development Team at the National Geospatial-Intelligence Agency that developed the current reference model of the gravitational potential of the Earth, the Earth Gravitational Model

2008 (EGM2008). An evaluation of this model is given in Figure 2.2. There, the model from degree 3 up to degree 2190 and order 2159 is evaluated. Note that this resembles only the more local structures of the gravitational potential as the degrees below 3 are left out. The EGM2008 is the current official model of the gravitational potential. However, it is expected to be superseded by a successor model which shall be published in 2020 (see e. g. Barnes et al., 2015).
For the representation (2.26), we immediately obtain the following result.

Lemma 2.4.8. *The upward continued value of a spherical harmonic basis function $Y_{m,k}$, for a fixed m, $j \in \mathbb{N}_0$ with $-m \leq j \leq m$, is given by*

$$(\mathcal{T} Y_{m,k})(\sigma\eta) = \sigma^{-m-1} Y_{m,k}(\eta) \tag{2.27}$$

for all $\sigma > 1$ and $\eta \in \Omega$.

Proof. We start on the left-hand side of the equation:

$$\begin{aligned}(\mathcal{T} Y_{m,k})(\sigma\eta) &= \sum_{n=0}^{\infty} \sum_{j=-n}^{n} \langle Y_{m,k}, Y_{n,j} \rangle_{\mathrm{L}^2(\Omega)} \sigma^{-n-1} Y_{n,j}(\eta) \\ &= \sum_{n=0}^{\infty} \sum_{j=-n}^{n} \delta_{n,m}\delta_{j,k} \sigma^{-n-1} Y_{n,j}(\eta) = \sigma^{-m-1} Y_{m,k}(\eta). \qquad \square\end{aligned}$$

The upward continuation operator gives us the gravitational potential on a satellite orbit while it uses only values of the potential at the Earth's surface. In other words, we obtain the effect, i. e. the potential on the orbit, with only knowledge of the cause, i. e. the potential at the surface. However, our actual interest of continuing the potential downwards onto the Earth's surface is described by the inverse task. In other words, we want to know the cause while we only have knowledge of its effect. This is a description of an inverse problem.

2.5. An overview of inverse problems

In this section, we summarize some aspects of the theory of inverse problems. We start with a general model and give criteria for the well- and ill-posedness of such a problem, i. e. for whether it can be solved easily or not. Then we introduce regularization schemes which are used to remedy the ill-posedness. We state the Tikhonov-Philipps regularization as an example of such a strategy. This section is based on Engl et al. (1996); Hofmann (1999); Kirsch (1996); Kontak (2018); Rieder (2003).

A general model Modelling a physical law is often done by the formulation of a relation between the cause and its effect. For instance, the gravitational potential at the Earth surface is caused by the mass density in the Earth *(inverse gravimetry)*. The same potential measured in space depends on its value at the surface of

the Earth *(downward continuation)*. Also in medical imaging similar problems occur. For instance, the activity of the neural currents in the brain cause an electric as well as a magnetic field outside the head *(electro- and magnetoencephalography problem)*. Further, the amount of radiation that can be measured behind an object under investigation is determined by the inner of the object itself *((computer) tomography)*. Thus, the use of cause-effect-relations in diverse applications gives rise to an abstract, mathematical formulation.

Definition 2.5.1. *Let X and Y denote Hilbert spaces. The space X stands for the causes and Y for the effects. The relation of a cause and its effect is modelled by an operator $T\colon X \to Y$ such that*

$$Tx = y$$

for $x \in X$ and $y \in Y$. The determination of y if x is known is called the direct problem. Conversely, if y is known and x is sought, the problem is called an inverse problem. If X and Y are function spaces of spherical functions, the inverse problem is also called a spherical inverse problem.

Example 2.5.2. An example of an inverse problem is the downward continuation of the gravitational potential. The Hilbert spaces are $X = \mathrm{L}^2(\Omega)$ and $Y = \mathrm{L}^{(2)}(\Omega_{b/a})$. However, note that T is only well-defined on $\mathrm{B}_{\searrow\Omega}$ and $T(\mathrm{B}_{\searrow\Omega}) = \mathrm{Pot}^{(0)}(\overline{\mathbb{B}^{\mathrm{ext}}})$ with $\mathbb{B}^{\mathrm{ext}} := \mathbb{R}^3 \backslash \overline{\mathbb{B}}$. The cause $x \in \mathrm{B}_{\searrow\Omega}$ is given by the values of the potential at the surface of the Earth and the effect $y \in \mathrm{Pot}^{(0)}(\overline{\mathbb{B}^{\mathrm{ext}}})$ is the potential outside the Earth, for instance on a satellite orbit $\Omega_{b/a}$. Then we have $T = \mathcal{T}$ from (2.25) .

It is intuitive that not all inverse problem are easy to solve. In fact, the direct problem is mostly much easier to handle. Next, we consider a characterization for inverse problems.

Definition 2.5.3. ***(Well- and Ill-Posedness in the sense of Hadamard)*** *Let X, Y be Hilbert spaces and $T\colon X \to Y$ an operator. The inverse problem $Tx = y$ for $x \in X$ and $y \in Y$ is called well-posed if the following three properties are fulfilled:*
the solution $x \in X$
(I) exists,
(II) is unique, and
(III) depends continuously on y, i. e. the inverse operator T^{-1} is continuous.
If at least one property is violated, the problem is called ill-posed.

In the sequel, we consider inverse problems with linear and bounded operators T.

A look on ill-posed inverse problems We start with a detailed look on criteria (I) and (II) of Definition 2.5.3. Let $T\colon X \to Y$ be a linear and bounded operator between Hilbert spaces X and Y. From property (I), we obtain that T must be well-defined on the whole space X and be surjective, i. e. $T(X) = Y$. However, if the range $T(X)$ is a real subset of Y and $y \in Y \backslash T(X)$, then we can consider an element $x \in X$ whose image $Tx \in Y$ has a minimal distance to y. In the sequel, this element denotes a solution of the inverse problem as follows. For a better readability, we assume that T is well-defined on the whole space X from now on. At first, we consider the so-called normal-equation, see, for instance, Engl et al. (1996, p. 35) or Rieder (2003, pp. 21-22).

Theorem 2.5.4. *Let X, Y be Hilbert spaces, $T\colon X \to Y$ be a linear and bounded operator and T^* is the adjoint operator, i. e. $\langle Tx, y\rangle_Y = \langle x, T^*y\rangle_X$ for all $x \in X$ and $y \in Y$. The normal equation*

$$T^*Tx = T^*y,$$

has at least one solution if $y \in T(X) \oplus T(X)^\perp$. The set of solutions of the normal equation is closed and convex. Thus, there exists a unique solution with minimal norm. Further, for a fixed $y \in Y$ and a linear and bounded operator T, each solution minimizes the residual $\|Tx - y\|_Y$.

The existence and uniqueness of such a solution yields the following.

Definition 2.5.5. *Let X, Y be Hilbert spaces and $T\colon X \to Y$ be a linear and bounded operator. The generalized inverse $T^\dagger\colon T(X) \oplus T(X)^\perp \to X$ or Moore-Penrose inverse is defined by*

$$T^\dagger y = x^\dagger$$

with

$$\left\|x^\dagger\right\|_X < \|\widetilde{x}\|_X$$

for all $\widetilde{x} \in X, \widetilde{x} \neq x^\dagger$, which fulfil $T^\dagger T\widetilde{x} = T^\dagger y$. The solution $x^\dagger$ is called the minimum norm solution and is the unique solution of the normal equation that fulfils

$$x^\dagger = \operatorname*{arg\,min}_{x \in X} \|Tx - y\|_Y.$$

With the generalized inverse and the minimum-norm solution of an ill-posed inverse problem, we found a way to circumvent problems caused by property (I) and (II) in Definition 2.5.3. Thus, with respect to ill-posed inverse problems, we are only concerned with property (III) in the sequel. For our solution, the generalized inverse, this property can be reformulated (see e. g. Rieder, 2003, p. 23-24).

Theorem 2.5.6. *The generalized inverse $T^\dagger$ as defined in Definition 2.5.5 is continuous if and only if the range of T is closed, i. e. $T(X) = \overline{T(X)}$.*

Definition 2.5.7. ***(Well- and Ill-Posedness in the sense of Nashed)*** *Let X, Y be Hilbert spaces and $T\colon X \to Y$ be a linear and bounded operator. The inverse problem $Tx = y$ for $x \in X$ and $y \in Y$ is called well-posed in the sense of Nashed if the range of T is closed in Y, i. e. $T(X) = \overline{T(X)}$. Otherwise, the inverse problem is called ill-posed in the sense of Nashed.*

Note that it obviously holds $T(X) \subseteq \overline{T(X)}$. For determining whether an inverse problem is ill-posed, we have to show that $\overline{T(X)} \subseteq T(X)$.
For practical applications, the continuity of the inverse is of major importance. The effect $y \in Y$ is usually obtained from physical experiments. For instance, we use measurements from a satellite mission. However, naturally, these measurements are defective. Thus, we only have perturbed data $y^\delta \approx y$ instead of the exact y where δ is called the noise level. This may cause problems when solving an inverse problem. If the problem is ill-posed, a – however obtained – solution x^δ of the perturbed inverse problem $Tx^\delta = y^\delta$ can be far from the solution x of the exact problem $Tx = y$ due to the missing continuity. Thus, an approach to solve an ill-posed inverse problem needs to take care of this instability.
One possible approach is to regularize the problem before solving it. Before we define regularization strategies in general and give an overview of the Tikhonov-Philipps regularization in particular, we first outline that the downward continuation is ill-posed. For this, we consider compact operators next.

Definition 2.5.8. *Let X and Y be normed spaces. An operator $T\colon X \to Y$ is compact if the image $(Tx_n)_{n\in\mathbb{N}_0}$ of every bounded sequence $(x_n)_{n\in\mathbb{N}_0} \subset X$ has a convergent subsequence.*

Theorem 2.5.9. *Every linear and compact operator is also bounded.*

Example 2.5.10. Let $G_i \subset \mathbb{R}^d$, $i = 1, 2$, be non-empty and compact sets and let be $k \in \mathrm{L}^2(G_1 \times G_2)$. We consider the Fredholm integral operator of the first kind $T\colon \mathrm{L}^2(G_2) \to \mathrm{L}^2(G_1)$

$$Tf(\cdot) := \int_{G_2} k(\cdot, y) f(y)\, \mathrm{d}y.$$

It can be shown that this operator is compact (see e. g. Heuser, 2006, Chapter 87) if G_1 and G_2 are intervals. Analogously, it holds for compact subsets G_1 and G_2 of higher dimension. An example is the upward continuation operator $\mathcal{T}$ as given in (2.24) if we set $k \in \mathrm{L}^{(2)}(\Omega \times \Omega_{b/a})$ with

$$k(\chi, \sigma\eta) := \frac{1}{4\pi} \frac{\sigma^2 - 1}{|\chi - \sigma\eta|^3}. \tag{2.28}$$

Now, for two Hilbert spaces X and Y, let $T\colon X \to Y$ be a linear and compact operator. Further, we assume that its inverse T^{-1} exists. If T^{-1} is continuous, then the identity $I = T^{-1} \circ T$ is compact as well. However, it is known (see e. g. Rieder, 2003, pp. 27-28) that this only holds if X has a finite dimension. Thus, for compact operators between infinite dimensional spaces like the Fredholm integral operator of the first kind, the inverse T^{-1} cannot be continuous. That means the corresponding inverse problem is ill-posed.

Remark 2.5.11. We note the following aspects:

(a) In the sequel, we consider inverse problems between infinite dimensional spaces.

(b) The previous considerations show that the downward continuation of satellite data is an ill-posed inverse problem.

The singular value decomposition of a linear and compact operator In the last section, we have seen that the gravitational potential allows a series representation in outer harmonics. This approach can be generalized for compact operators.

Definition 2.5.12. *Let X be a normed space and $T\colon X \to X$ a linear and bounded operator. If $\mu I - T$, $\mu \in \mathbb{C}$, has a continuous inverse, then μ is called a regular value of T. The set of non-regular values is called the spectrum of T. A non-regular value μ of T is called an eigenvalue, if $(\mu I - T)v \neq 0$ for at least one $0 \neq v \in X$. Then the element $v \in X$ is called an eigenvector or eigenfunction of T with respect to μ.*

We use this result for the product T^*T of the compact linear operator T and its adjoint operator T^* (for the latter see Theorem 2.5.4). It can be shown that the non-regular values of T^*T are real and positive (see e. g. Rieder, 2003, p. 31) if not zero. Then we can define the singular system of T.

Definition 2.5.13. *Let X, Y be Hilbert spaces and $T\colon X \to Y$ be a compact linear operator. Except for the zero value, the non-regular values of the self-adjoint product T^*T of the operator T and its adjoint operator T^* are positive and, thus, can be ordered as $\mu_0 \geq \mu_1 \geq \cdots > 0$. For $n \in \mathbb{N}_0$, the respective orthonormal system of eigenvectors is denoted by $\{v_n\} \subset X$. At last, we set*

$$\sigma_n := \sqrt{\mu_n}, \qquad u_n := \sigma_n^{-1} T v_n.$$

The system

$$\{(\sigma_n; v_n, u_n) \mid n \in \mathbb{N}_0\} \subset]0, \infty[\times X \times Y$$

is called a singular system of T. The values σ_n are called singular values and the elements v_n and u_n are called singular functions or singular vectors. Further, there exists an expansion of T as

$$Tx = \sum_{n=0}^{\infty} \langle x, v_n \rangle_X \sigma_n u_n \tag{2.29}$$

which is called the singular value decomposition (SVD) of T.

Compare (2.23) and in particular (2.26) with (2.29): for the upward continuation operator, we have

$$Tx = \sum_{n=0}^{\infty} \sum_{j=-n}^{n} \langle x, v_{n,j} \rangle_X \sigma_{n,j} u_{n,j}$$

and, hence, $\sigma_{n,j} = \sigma^{-n-1}$ for all $j = -n, ..., n$ and with $\sigma > 1$. With the singular value decomposition, we obtain another condition for a well-posed problem (see e. g. Rieder (2003, pp. 31-32) or Kirsch (1996, pp. 241-242)).

Theorem 2.5.14. ***(Picard condition)*** *Let X, Y be Hilbert spaces and $T\colon X \to Y$ be a compact linear operator with a singular system $\{(\sigma_n, v_n, u_n) \mid n \in \mathbb{N}_0\}$. Any element $y \in \overline{T(X)}$ is also an element of $T(X)$ if and only if*

$$\sum_{n=0}^{\infty} \frac{|\langle y, u_n \rangle_Y|^2}{\sigma_n^2} < \infty. \tag{2.30}$$

That means, an inverse problem $Tx = y$ is well-posed in the sense of Nashed if and only if (2.30) holds for all $y \in \overline{T(X)}$.

Remark 2.5.15. To have (2.30) fulfilled for all $y \in \overline{T(\mathrm{L}^2(\Omega))}$, the coefficients $\langle y, u_n \rangle_Y$ need to have a faster decrease rate than the singular values. With respect to the upward continuation, we have seen that the singular values $\sigma^{-n-1} = (a/b)^{n+1}$, for $\sigma = b/a > 1$, are exponentially decreasing. Thus, the condition (2.30) cannot be fulfilled in general for $y \in \mathrm{L}^{(2)}(\Omega_{b/a})$. Thus, we see again that the downward continuation is an ill-posed inverse problem.

Approximating the ill-posed problem: regularization strategies Solving an ill-posed inverse problem $Tx = y$ has its difficulties as we have seen. It is even worse if we need to solve the perturbed problem $Tx^\delta = y^\delta$ for a noise level $\delta > 0$. We already mentioned that regularization strategies are a possible approach for handling this. In the sequel, we assume that $y^\delta \in Y$.
The idea how to obtain the solution of $Tx^\delta = y^\delta$ is as follows. We do not aim to solve the problem as best as possible but as best as necessary (see Hofmann, 1999, p. 127). Mathematically, we substitute the operator T with a certain approximate one $T_{\lambda(\delta, y^\delta)}$ such that $T_{\lambda(\delta, y^\delta)}$ corresponds to a well-posed inverse problem $T_{\lambda(\delta, y^\delta)} x^\delta = y^\delta$. This problem yields an approximate solution x^δ of x – at least, both vectors are solutions in the sense of the minimum-norm solution (i. e. for a solution related to unperturbed data x, we are given the respective unperturbed minimum-norm solution $x^\dagger$ and, for a solution related to perturbed data x^δ, we are given the respective perturbed minimum-norm solution $(x^\dagger)^\delta$). When solving the approximate problem instead of the original one, we trade the exactness of the solution for a guarantee to be near the exact solution.
The approximative problem $T_{\lambda(\delta, y^\delta)} x^\delta = y^\delta$ and, thus, the solution x^δ is obtained via a regularization strategy.

Definition 2.5.16. *Let X, Y be Hilbert spaces and $T\colon X \to Y$ be a linear and bounded operator. A noise level of the related inverse problem is given by $\delta > 0$. Further, $T^\dagger$ denotes the generalized inverse and $x^\dagger$ the minimum-norm solution. At last, let $(T^\dagger_{\lambda(\delta,y^\delta)})_{\lambda(\delta,y^\delta)\in\mathbb{R}^+}$ with $T^\dagger_{\lambda(\delta,y^\delta)}\colon Y \to X$ be a family of continuous (not necessarily linear) operators. If, for all $y \in T(X)$, it holds*

$$\limsup_{\delta\searrow 0}\left\{\left\|T^\dagger y - T^\dagger_{\lambda(\delta,y^\delta)}y^\delta\right\|_X \;\middle|\; y^\delta \in Y,\ \left\|y-y^\delta\right\|_Y \le \delta\right\} = 0$$

and

$$\limsup_{\delta\searrow 0}\left\{\lambda\left(\delta,y^\delta\right) \;\middle|\; y^\delta \in Y,\ \left\|y-y^\delta\right\|_Y \le \delta\right\} = 0, \tag{2.31}$$

then $(T^\dagger_{\lambda(\delta,y^\delta)})_{\lambda(\delta,y^\delta)\in\mathbb{R}^+}$ is called a regularization. The mapping $\lambda\colon \mathbb{R}^+\times Y \to \mathbb{R}^+$ is called a parameter choice rule, a particular value $\lambda\left(\delta,y^\delta\right)$ is called a regularization parameter and a particular operator $T^\dagger_{\lambda(\delta,y^\delta)}$ is called regularization strategy.

Thus, a regularization is a family of approximative operators for which the solutions of the related inverse problems converge to the minimum norm solution for a decreasing noise level. Such a strategy can be investigated with respect to its convergence rates and optimality. A way to construct a regularization strategy is given by certain filters. However, in this thesis, we are only interested in their definition and the properties of a particular strategy. Thus, for further details on general properties and construction methods, we refer the reader to the common literature on the regularization of inverse problems, for instance, Engl et al. (1996); Hofmann (1999); Kirsch (1996); Rieder (2003).
The regularization of interest in this thesis is the so-called Tikhonov-Philipps regularization. We introduce the method next. The example is based on Kontak (2018, Subsect. 6.1.3 and 6.1.4) and Rieder (2003, pp. 70-71 and Chapter 4) where more details can be found as well.

Example 2.5.17. Let X, Y be Hilbert spaces and $T\colon X \to Y$ be a linear and bounded operator. A noise level of the related inverse problem is given by $\delta > 0$. Further, let $\lambda\colon \mathbb{R}^+\times Y \to \mathbb{R}^+$ be a parameter choice rule, i. e. fulfil (2.31).
In the Tikhonov-Philipps regularization, we set

$$T^\dagger_{\lambda\left(\delta,y^\delta\right)} := \left(T^*T + \lambda\left(\delta,y^\delta\right)I\right)^{-1}T^*,$$

where I is the identity operator on X. Then we have

$$\begin{aligned} T^*y^\delta &= \left(T^*T + \lambda\left(\delta,y^\delta\right)I\right)\left(T^*T + \lambda\left(\delta,y^\delta\right)I\right)^{-1}T^*y^\delta \\ &= \left(T^*T + \lambda\left(\delta,y^\delta\right)I\right)T^\dagger_{\lambda\left(\delta,y^\delta\right)}y^\delta, \end{aligned}$$

which is called the regularized normal equation (compare with Theorem 2.5.4). Its unique solution is denoted by

$$x^{\dagger}_{\lambda\left(\delta,y^{\delta}\right)} := T^{\dagger}_{\lambda\left(\delta,y^{\delta}\right)} y^{\delta}.$$

This solution is equivalently obtained by

$$\begin{aligned} x^{\dagger}_{\lambda\left(\delta,y^{\delta}\right)} &:= \arg\min_{x\in X} \mathcal{J}\left(x; T, \lambda\left(\delta, y^{\delta}\right), y^{\delta}\right) \\ &:= \arg\min_{x\in X} \left(\left\| y^{\delta} - Tx \right\|^{2}_{Y} + \lambda\left(\delta, y^{\delta}\right) \|x\|^{2}_{X} \right). \end{aligned}$$

This functional is called the Tikhonov-Philipps functional. The first summand is called data fidelity term or data error and the second summand is called penalty term or approximation error. The convergence rate of the Tikhonov-Philipps regularization is estimated by

$$\left\| x_{\lambda\left(\delta,y^{\delta}\right)} - x^{\dagger}_{\lambda\left(\delta,y^{\delta}\right)} \right\|_{X} \leq \left(\frac{1}{2\sqrt{m}} + m \right) c^{1/3} \delta^{2/3}$$

with a constant $m \in \mathbb{R}^{+}$ and $\|z\|_{X} \leq c \in \mathbb{R}^{+}$ with $x^{\dagger}_{\lambda(\delta,y^{\delta})} = T^{*}Tz$. Note that an iterated Tikhonov-Philipps regularization exists in the literature. In this regularization the penalty term is modified to

$$\mathcal{J}\left(x; T, \lambda\left(\delta, y^{\delta}\right), y^{\delta}, i\right) := \left\| y^{\delta} - Tx \right\|^{2}_{Y} + \lambda\left(\delta, y^{\delta}\right) \left\| x - \left(x^{\dagger}_{\lambda\left(\delta,y^{\delta}\right)} \right)^{i} \right\|^{2}_{X}$$

where

$$\left(x^{\dagger}_{\lambda\left(\delta,y^{\delta}\right)} \right)^{i} := \arg\min_{x\in X} \mathcal{J}\left(x; T, \lambda\left(\delta, y^{\delta}\right), y^{\delta}, i-1\right).$$

Hence, in the iterated Tikhonov-Philipps regularization, not the solution itself is considered to stabilize the problem but the distance between two successive iterates. The iterated variant is used, for instance, in order to improve the speed of the convergence.

About the choice of a regularization parameter The choice of the regularization parameter is of importance because it influences which summand in the Tikhonov-Philipps functional has more weight. Either, if $\lambda(\delta, y^{\delta}) = 0$, we only consider the data fidelity of a solution and neglect any regularization. Or, if $\lambda(\delta, y^{\delta})$ is chosen such that $\|y^{\delta} - Tx\|^{2}_{Y} \ll \lambda(\delta, y^{\delta})\|x\|^{2}_{X}$, the solution will not be very true to the given y^{δ}.

There exists a wide range of parameter choice rules or methods which are approaches to determining the best suitable value $\lambda(\delta, y^\delta)$ for a given perturbed inverse problem (see e. g. Bauer et al., 2015; Bauer and Lukas, 2011; Gutting et al., 2017, and the references therein). Further, we can use a separation ansatz for the parameter choice rule, e. g., by

$$\lambda\left(\delta, y^\delta\right) := \lambda_0(\delta) \left\| y^\delta \right\|_Y$$

for a $\lambda_0 \colon \mathbb{R}^+ \to \mathbb{R}^+$. In this way, the value $\lambda_0(\delta)$ can be viewed as a percentage of the norm of the data y^δ.
In future works, the following approaches could be considered for the determination of λ_0. In principle, the separation ansatz can be extended by the use of a total variation (see e. g. Chambolle and Lions, 1997; Defrise et al., 2011; Li et al., 2013; Rudin et al., 1992). The total variation can be able to distinguish the signal and the noise because it penalizes high frequencies in the signal. It can be either used instead of the norm $\|y^\delta\|_Y$. Or, with respect to the Tikhonov regularization, we can also use it as a second penalty term. Then the regularization is similar to an elastic-net approach (see e. g. Daubechies et al., 2004; De Mol et al., 2009; Zou and Hastie, 2005).
A description of how we choose the regularization parameter in our numerical experiments will be given later in dependence of our algorithms for ill-posed inverse problems.

3. Particular real-valued trial functions on the sphere

The aim of this thesis is to develop a dictionary learning strategy for spherical inverse problems. In other words, we want to automatically select trial functions for the approximation of real-valued signals on the sphere. Thus, the main concern of this thesis is about choosing specific functions which are suitable for this need. The gravitational potential as our example of a given signal has certain mathematical idiosyncrasies like, e. g., having a global support. Therefore it is sensible to preselect types of trial functions from the range of possible ones.
We have already introduced spherical harmonics in Section 2.2 which are a traditional choice of global trial functions. In this chapter, we introduce further types of trial functions which can be used in the learning approach for spherical inverse problems that are similar to the downward continuation of satellite data of the gravitational potential of the Earth. We start with scalar Slepian functions which can be viewed as a bridge between global and local functions. After that, we discuss radial basis functions and related wavelets as examples for local functions. For this discussion, we first take a look at spherical Sobolev spaces. The section closes with an overview of the values of the upward continued trial functions as well as certain inner products of them.

3.1. A few aspects of scalar Slepian functions

We use spherical harmonics to construct another type of trial functions: (scalar) Slepian functions. Due to this construction, they inherit the polynomial structure of spherical harmonics. However, they are built as optimally localized functions. In this way, they build a bridge between purely global and local functions. The Slepian functions discussed here are based on, e. g., Albertella et al. (1999); Grünbaum et al. (1982); Leweke et al. (2018a); Michel (2013); Seibert (2018); Simons et al. (2006).
Due to well-known uncertainty principles, a function cannot be simultaneously perfectly localized in space and frequency. We have seen that, for example, spherical harmonics are localized in frequency but not in space. On the contrary, we will see later that, for instance, the Abel–Poisson kernel is spatially localized, but has infinitely many non-vanishing Legendre coefficients. To combine these concepts, we consider band-limited functions, i. e. functions with a truncated Fourier expansion.

Definition 3.1.1. *Let $N \in \mathbb{N}_0$ be fixed. A function $g^N \colon \Omega \to \mathbb{R}$, $g \in \mathrm{L}^2(\Omega)$, is band-limited if*

$$g^N = \sum_{n=0}^{N} \sum_{j=-n}^{n} g^\wedge(n,j) Y_{n,j} \tag{3.1}$$

for spherical harmonics $Y_{n,j}$ and Fourier coefficients $(g^N)^\wedge(n,j) \in \mathbb{R}$.

Thus, band-limited functions are – to some extent – localized in frequency and it is not possible for them to be perfectly localized in space as well. Therefore, we consider the approach to have a band-limited function which is only optimally localized in space. For this, we define a parameter to measure how well a band-limited function is localized in space. This parameter is called the energy ratio.

Definition 3.1.2. *Let $g^N \in \mathrm{L}^2(\Omega)$ be a band-limited function on the sphere. The energy ratio $\rho \in [0,1]$ with respect to g^N and a localization region $R \subseteq \Omega$ is given by*

$$\rho := \frac{\|g^N\|^2_{\mathrm{L}^2(R)}}{\|g^N\|^2_{\mathrm{L}^2(\Omega)}}.$$

This quotient is the most intuitive approach to measure how well localized a function $g^N \in \mathrm{L}^2(\Omega)$ is in a particular region R. The energy ratio gives us the share of the energy of g^N in a subset $R \subseteq \Omega$ in the energy of g^N on the whole sphere. Obviously, the value ρ is a real number between 0 and 1. This means, if ρ is nearly one, the energy of g^N in the localization region nearly equals the energy of g^N on the whole of its domain. Then, if the localization region is much smaller than the sphere, the function must nearly vanish outside of it. If ρ is nearly 0, the energy of g^N in the localization region is negligible in comparison to the energy on the sphere. Thus, if ρ is large, g^N is localized in R. If ρ is small, g^N is localized in the complement $\Omega \backslash R$.
For a characterization of optimally localized functions, we consider the energy ratio in more detail. Using (3.1), the ratio can be reformulated to an algebraic eigenvalue problem. For N, n, n', j, $j' \in \mathbb{N}_0$ with $-n \leq j \leq n$ and $-n' \leq j' \leq n'$, we have

$$\begin{aligned} \rho &= \frac{\sum_{n=0}^{N} \sum_{j=-n}^{n} \sum_{n'=0}^{N} \sum_{j=-n'}^{n'} \left(g^N\right)^\wedge (n,j) \left(g^N\right)^\wedge (n',j') \langle Y_{n,j}, Y_{n',j'} \rangle_{\mathrm{L}^2(R)}}{\sum_{n=0}^{N} \sum_{j=-n}^{n} \left(g^N\right)^\wedge (n,j) \left(g^N\right)^\wedge (n,j)} \\ &= \frac{(\hat{g}^N)^{\mathrm{T}} L(R) \hat{g}^N}{(\hat{g}^N)^{\mathrm{T}} \hat{g}^N} \end{aligned} \tag{3.2}$$

for the vector notation

$$\hat{g}^N = \left(\left(g^N\right)^\wedge (0,0), \ldots, \left(g^N\right)^\wedge (n,j), \ldots, \left(g^N\right)^\wedge (N,N) \right)$$

of real coefficients $(g^N)^\wedge(n,j)$ and the localization matrix $L(R) \in \mathbb{R}^{(N+1)^2\times(N+1)^2}$ defined by

$$L(R) := \left(\langle Y_{n,j}, Y_{n',j'}\rangle_{\mathrm{L}^2(R)}\right)_{n,n'=0,\dots,N;\ j,j'=-n,\dots,n}. \tag{3.3}$$

Note that $L(R)$ is a Gramian matrix. Thus, it is symmetric, real and positive definite. Due to the Principal Axis Theorem, its eigenvalues are real and positive and the respective eigenvectors are real and pairwise orthonormal. By construction, the eigenvectors of $L(R)$ are built from the Fourier coefficients $(g^N)^\wedge(n,j)$ of the band-limited function g^N under consideration. Thus, we define scalar Slepian functions via the solutions of the eigenvalue problem (3.2).

Definition 3.1.3. *Let $Y_{n,j}$, n, $j \in \mathbb{N}_0$, $n \le N \in \mathbb{N}_0$, $-n \le j \le n$, denote spherical harmonics. A (scalar) Slepian function on the sphere localized in a region $R \subseteq \Omega$ is a band-limited function $g^{(k,N)}(R,\cdot)\colon \Omega \to \mathbb{R}$, $k = 1,\dots,(N+1)^2$, with*

$$g^{(k,N)}(R,\cdot) = \sum_{n=0}^{N}\sum_{j=-n}^{n} g_{n,j}^{(k,N)}(R)Y_{n,j}, \tag{3.4}$$

where $g_{n,j}^{(k,N)}(R) := ((g^{(k,N)})^\wedge(n,j))(R)$ is the (n,j)-th entry of the k-th eigenvector of the Gramian matrix $L(R)$ as given in (3.3)*. The related eigenvalue is denoted $\rho^{(k,N)}(R)$ and equals the energy ratio of the Slepian function $g^{(k,N)}(R,\cdot)$.*

Note that the functions $g^{(k,N)}$, $k = 1,\dots,(N+1)^2$, are usually ordered such that $g^{(1,N)}$ corresponds to the highest eigenvalue and $g^{((N+1)^2,N)}$ corresponds to the lowest one. We constructed Slepian functions as band-limited linear combinations of spherical harmonics. Thus, a Slepian function is obviously an element of $\mathrm{L}^2(\Omega)$. However, a set of Slepian functions cannot be closed in infinite-dimensional spaces like, e. g., $\mathrm{L}^2(\Omega)$ due to the finite band-limit. Nonetheless, they have an interesting property for a dictionary.

Theorem 3.1.4. *For k and $N \in \mathbb{N}_0$, a system of Slepian functions*

$$\left\{ g^{(k,N)}(R,\cdot) \;\middle|\; k = 1,\dots,(N+1)^2 \right\}$$

with respect to a localization region $R \subseteq \Omega$ is an orthonormal system in the space $(\mathrm{L}^2(\Omega), \langle\cdot,\cdot\rangle_{\mathrm{L}^2(\Omega)})$ as well as an orthogonal system in $(\mathrm{L}^2(R), \langle\cdot,\cdot\rangle_{\mathrm{L}^2(R)})$.

Proof. Let $g^{(k,N)}$ and $g^{(m,N)}$ be Slepian functions with band-limit N for k, $m \in \mathbb{N}_0$ and k, $m \le (N+1)^2$. We first consider the space $(\mathrm{L}^2(\Omega), \langle\cdot,\cdot\rangle_{\mathrm{L}^2(\Omega)})$. It holds

$$\left\langle g^{(k,N)}, g^{(m,N)} \right\rangle_{\mathrm{L}^2(\Omega)} = \sum_{n=0}^{N}\sum_{j=-n}^{n} \hat{g}_{n,j}^{(k,N)}\hat{g}_{n,j}^{(m,N)} = \delta_{k,m}$$

as the eigenvectors of the localization matrix $L(R)$ form an orthonormal basis in $\mathbb{R}^{(N+1)^2}$. From the eigenvalue problem (3.2), we obtain the generalization

$$\left\langle g^{(k,N)}, g^{(m,N)} \right\rangle_{\mathrm{L}^2(R)} = \rho \left\langle g^{(k,N)}, g^{(m,N)} \right\rangle_{\mathrm{L}^2(\Omega)} = \rho \delta_{k,m}$$

and, thus, the orthogonality in $(\mathrm{L}^2(R), \langle \cdot, \cdot \rangle_{\mathrm{L}^2(R)})$. □

At least, for a band-limit $N \in \mathbb{N}_0$, the set of Slepian functions of band-limit N is a basis of $\mathrm{Harm}_{0,\dots,N}(\Omega)$ because they are a linear combination of all spherical harmonic basis functions up to degree N.
From the approach taken, it is clear how to compute Slepian functions for practical purposes. We have to compute the Gramian matrix $L(R), R \subseteq \Omega$, and solve the respective eigenvalue problem. However, a difficulty evolves from this naive ansatz. Numerical experiments (see e. g. Albertella et al., 1999; Khalid et al., 2016; Leweke et al., 2018a; Plattner and Simons, 2014; Seibert, 2018; Simons, 2010; Simons and Dahlen, 2006; Simons et al., 2006) show that the obtained eigenvalues $\rho^{(k,N)}(R),\ k = 1, \dots, (N+1)^2$, are situated either in a neighbourhood of 1 or near 0. On the one hand, this is good because it underlines the idea of constructing optimally localized functions. Slepian functions related to an eigenvalue, or energy ratio, near 1 are usually called well-localized in the localization region as the energy ratio shows that they have nearly all their volume in the specified region R. Slepian functions related to a nearly vanishing energy ratio are called poorly-localized because they are nearly vanishing in the localization region. Thus, the separation of the eigenvalues shows that a clear distinction in well- and poorly localized functions is obtained by this approach. On the other hand, the separation of eigenvalues in being nearly 1 and being nearly 0 means that the localization matrix is nearly singular due to a high condition number. Hence, the numerical computation of Slepian functions can be problematic.
However, for particular regions such as a spherical cap, the instability of the localization matrix can be bypassed. Thus, we consider spherical caps as particular localization regions next.

Example 3.1.5. We consider Slepian functions which are localized in a spherical cap with an arbitrary centre on the sphere. Thus, for $\varepsilon^3 = (0,0,1)^{\mathrm{T}}$, we have the localization region

$$R\left(c, A\varepsilon^3\right) := \left\{ \eta \in \Omega \;\middle|\; \left| \eta - A\varepsilon^3 \right| \leq c \right\}$$

for $c \in [-1,1]$ and a rotation matrix $A \in \mathrm{SO}(3)$. Note that for $A = I$ (I the identity matrix in $\mathbb{R}^{3\times 3}$), we consider the spherical cap around the North Pole, i. e. with the centre of the localization region given by ε^3, and simply write $R(c, \varepsilon^3)$.
We can parametrize the Slepian functions with the cap size c and the centre of the localization region. The latter one is given by the rotation of ε^3 about A. We

obtain the following notation:

$$g^{(k,N)}\left(\left(c, A\varepsilon^3\right), \cdot\right) := g^{(k,N)}\left(R\left(c, A\varepsilon^3\right), \cdot\right) := \sum_{n=0}^{N}\sum_{j=-n}^{n} g_{n,j}^{(k,N)}\left(R\left(c, A\varepsilon^3\right)\right) Y_{n,j}(\cdot).$$

An example of a set of Slepian functions with respect to a fixed band-limit and localization region is given in Figure 3.1. There, the 36 Slepian functions of band-limit 5 which are localized in a spherical cap of size $c = \pi/4$ and centre of the spherical cap at $(0,1,0)^{\mathrm{T}}$ are presented. They are ordered with respect to decreasing eigenvalues of the commuting matrix (see Theorem 3.1.11). Thus, the figures start with well-localized functions and end with poorly-localized ones. Note that, as usual, one of the well- and poorly-localized functions have exactly one extremum. Obviously, the Slepian functions inherit the polynomial structure of the – in this case – fully normalized spherical harmonics (see Figure 2.1). Further, we notice that in the range of the Slepian functions the well-localized functions attain their polynomial structure only in the desired localization region (see the first row). Moreover, the poorly-localized functions show the same behaviour in the complement of the localization region (see the last row). At last, there are functions which are neither localized in the localization region nor in its compliment (see the third and fourth row). This separation is naturally seen with Slepian functions because they form an orthonormal system in $(\mathrm{L}^2(\Omega), \langle\cdot,\cdot\rangle_{\mathrm{L}^2(\Omega)})$.

About rotating Slepian functions From Example 3.1.5, we can generalize the idea of rotating a standard localization region to obtain arbitrary regions. In the sequel, we consider the localization region R of a Slepian function to be the result of the rotation of a standard region R' by means of a rotation matrix $A \in \mathrm{SO}(3)$ where $\varepsilon^3 = (0,0,1)^{\mathrm{T}}$ is the centre of R'. Then we consider Slepian functions to be localized in R' and rotated by means of $A \in \mathrm{SO}(3)$. Therefore, we investigate some properties of rotation matrices next (see e. g. Dahlen and Tromp, 1998; Freeden and Gutting, 2013; Gutting, 2007).

Definition 3.1.6. *A rotation matrix $A(\alpha,\beta,\gamma) \in \mathrm{SO}(3)$ is a $3\times 3-$matrix of the form*

$$A(\alpha,\beta,\gamma) = \begin{pmatrix} \cos(\alpha)\cos(\beta)\cos(\gamma) - \sin(\alpha)\sin(\gamma) & -\cos(\alpha)\cos(\beta)\sin(\gamma) - \sin(\alpha)\cos(\gamma) & \cos(\alpha)\sin(\beta) \\ \sin(\alpha)\cos(\beta)\cos(\gamma) + \cos(\alpha)\sin(\gamma) & -\sin(\alpha)\cos(\beta)\sin(\gamma) - \cos(\alpha)\cos(\gamma) & \sin(\alpha)\sin(\beta) \\ -\sin(\beta)\cos(\gamma) & \sin(\beta)\sin(\gamma) & \cos(\beta) \end{pmatrix} \tag{3.5}$$

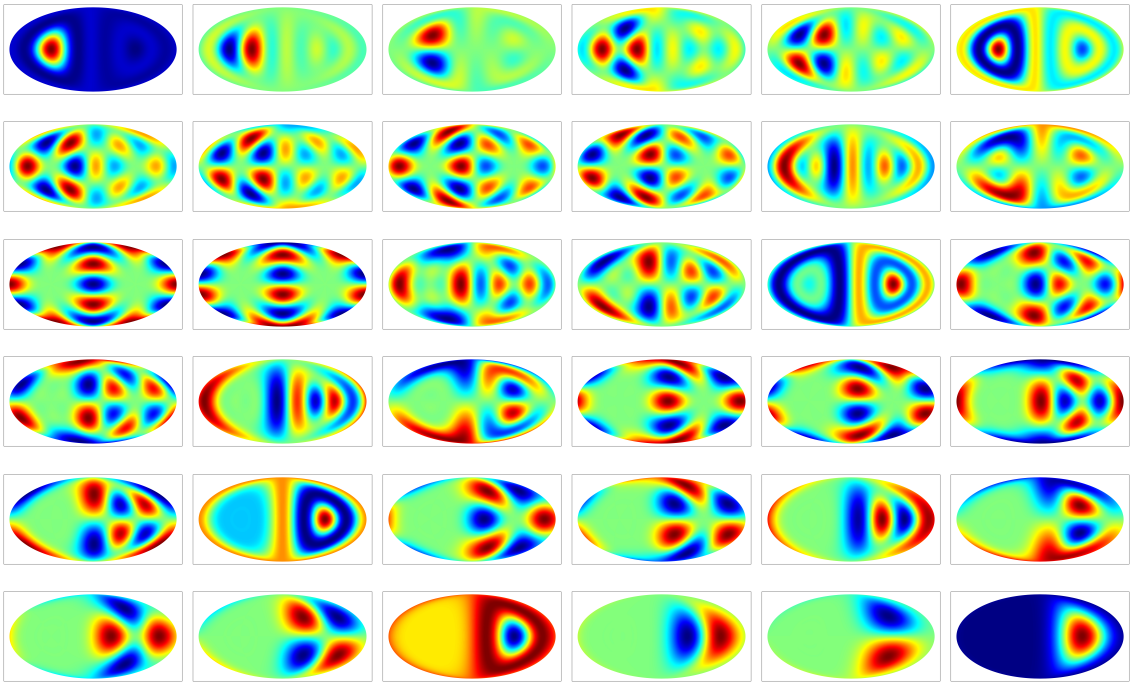


Figure 3.1.: Slepian functions of band-limit 5 localized in a spherical cap with $c = \pi/4$ and centre $(0,1,0)^{\mathrm{T}}$ ordered by decreasing eigenvalues.

with the Euler angles $\alpha \in [0, 2\pi)$, $\beta \in [0, \pi]$ *and* $\gamma \in [0, 2\pi)$. *With such a matrix, we can rotate a point* $\eta' \in \Omega$ *towards another point* $\eta \in \Omega$ *via the relation* $\eta = A(\alpha, \beta, \gamma)\eta'$. *Note that it holds* $\eta' = A^{\mathrm{T}}(\alpha, \beta, \gamma)\eta \in \Omega$ *as well. If for all* $\eta' \in R'$ *it holds that* $\eta = A(\alpha, \beta, \gamma)\eta' \in R$ *and vice versa, we write* $R = A(\alpha, \beta, \gamma)R'$ *for this relation.*

Obviously to rotate the Slepian functions, we could work with the values of the spherical harmonics at the rotated points, i. e.

$$g^{(k,N)}(R, \eta) = \sum_{n=0}^{N} \sum_{j=-n}^{n} g_{n,j}^{(k,N)}(R') Y_{n,j}\left(A\,(\alpha, \beta, \gamma)^{\mathrm{T}}\,\eta\right), \tag{3.6}$$

$$R = A(\alpha, \beta, \gamma)R',\ R', R \subseteq \Omega,\ \eta = A(\alpha, \beta, \gamma)\eta',\ \eta', \eta \in \Omega, A(\alpha, \beta, \gamma) \in \mathrm{SO}(3).$$

However, this is not useful in practical applications. It is more convenient to work with a rotated function $g^k(R, \cdot)$ than with rotated evaluation points. For this, the rotation of spherical harmonics needs to be considered. Then we obtain a rotated function $g^k(R, \cdot)$ which can be evaluated in the unaffected points η', i. e.

$$g^{(k,N)}(R, \eta') = \sum_{n=0}^{N} \sum_{j=-n}^{n} g_{n,j}^{(k,N)}(R) Y_{n,j}(\eta'), \tag{3.7}$$

$$R = A(\alpha, \beta, \gamma)R',\ R' \subseteq \Omega,\ A(\alpha, \beta, \gamma) \in \mathrm{SO}(3),\ \eta' \in \Omega.$$

In the sequel, we derive the relation between (3.6) and (3.7). At first, we consider Wigner rotation matrices (see e. g. Choi et al., 1999; Dahlen and Tromp, 1998; Freeden and Gutting, 2013; Gutting, 2007).

Definition 3.1.7. *For $n,\ k,\ j \in \mathbb{N}_0$, the n-th Wigner rotation matrix D^n is a complex matrix of size $(2n+1) \times (2n+1)$. The rows and columns are indexed from $-n$ to n. The matrix D^n depends on the Euler angles $\alpha,\ \beta,\ \gamma$. For $-n \leq k \leq n$, the matrix is iteratively given by*

$$D^0(\alpha,\beta,\gamma) = 1,$$

$$D^1(\alpha,\beta,\gamma) = \begin{pmatrix} \frac{1+\cos(\beta)}{2}\mathrm{e}^{\mathrm{i}(\alpha+\gamma)} & \frac{\sin(\beta)}{\sqrt{2}}\mathrm{e}^{\mathrm{i}\alpha} & \frac{1-\cos(\beta)}{2}\mathrm{e}^{\mathrm{i}(\alpha-\gamma)} \\ -\frac{\sin(\beta)}{\sqrt{2}}\mathrm{e}^{\mathrm{i}\gamma} & \cos(\beta) & \frac{\sin(\beta)}{\sqrt{2}}\mathrm{e}^{-\mathrm{i}\gamma} \\ \frac{1-\cos(\beta)}{2}\mathrm{e}^{\mathrm{i}(\gamma-\alpha)} & -\frac{\sin(\beta)}{\sqrt{2}}\mathrm{e}^{-\mathrm{i}\alpha} & \frac{1+\cos(\beta)}{2}\mathrm{e}^{-\mathrm{i}(\alpha+\gamma)} \end{pmatrix},$$

$$D^n_{k,j} = a^n_{k,j} D^1_{0,0} D^{n-1}_{k,j} + b^n_{k,j} D^1_{1,0} D^{n-1}_{k-1,j} + b^n_{-k,j} D^1_{-1,0} D^{n-1}_{k+1,j},$$
$$-n+1 \leq j \leq n-1$$

$$D^n_{k,j} = c^n_{k,-j} D^1_{0,-1} D^{n-1}_{k,j+1} + d^n_{k,-j} D^1_{1,-1} D^{n-1}_{k-1,j+1} + d^n_{-k,-j} D^1_{-1,-1} D^{n-1}_{k+1,j+1},$$
$$-n \leq j \leq n-2$$

$$D^n_{k,j} = c^n_{k,j} D^1_{0,1} D^{n-1}_{k,j-1} + d^n_{k,j} D^1_{1,1} D^{n-1}_{k-1,j-1} + d^n_{-k,j} D^1_{-1,1} D^{n-1}_{k+1,j-1},$$
$$-n+2 \leq j \leq n$$

with

$$a^n_{k,j} = \sqrt{\frac{(n+k)(n-k)}{(n+j)(n-j)}}, \qquad a^n_{k,j} = 0 \quad \text{for } k = \pm n,$$

$$b^n_{k,j} = \sqrt{\frac{(n+k)(n+k-1)}{2(n+j)(n-j)}}, \qquad b^n_{k,j} = 0 \quad \text{for } k = -n \text{ or } k = -n+1,$$

$$c^n_{k,j} = \sqrt{\frac{2(n+k)(n-k)}{(n+j)(n+j-1)}}, \qquad c^n_{k,j} = 0 \quad \text{for } k = \pm n,$$

$$d^n_{k,j} = \sqrt{\frac{(n+k)(n+k-1)}{(n+j)(n+j-1)}}, \qquad d^n_{k,j} = 0 \quad \text{for } k = -n \text{ or } k = -n+1.$$

With the Wigner rotation matrices, the following result is known for the rotation of complex spherical harmonics (see e. g. Freeden and Gutting, 2013, p. 192).

Theorem 3.1.8. *Let $n,\ j \in \mathbb{N}_0$ with $-n \leq j \leq n$ be fixed. The value of a complex spherical harmonic $Y_{n,j}$ at a point η' can be obtained as the linear combination of all complex spherical harmonics of the same degree at a point η in the following way:*

$$Y_{n,j}(\eta') = \sum_{k=-n}^{n} D^n_{k,j}(\alpha,\beta,\gamma) Y_{n,k}(\eta) \tag{3.8}$$

where $\eta = A(\alpha,\beta,\gamma)\eta'$. The complex value $D^n_{k,j}(\alpha,\beta,\gamma)$ is the (k,j)-th entry of the n-th Wigner rotation matrix with respect to the Euler angles $\alpha,\ \beta,\ \gamma$.

Therefore, we can write the Slepian functions localized in an arbitrary region R of the sphere in relation to the Slepian functions localized in a region R' of the same form which contains ε^3.

Theorem 3.1.9. *Let $Y_{n,j},\ n,\ j \in \mathbb{N}_0,\ n \leq N \in \mathbb{N}_0,\ -n \leq j \leq n$, denote complex spherical harmonics. For $\eta \in \Omega,\ A(\alpha,\beta,\gamma) \in \mathrm{SO}(3)$, a Slepian function can be written with the use of Wigner rotation matrices dependent on Euler angles $\alpha,\ \beta,\ \gamma$ as*

$$\begin{aligned} g^{(k,N)}(R,\eta) &:= \sum_{n=0}^{N}\sum_{i=-n}^{n} \widetilde{g}^{(k,N)}_{n,i}(R)Y_{n,i}(\eta) \\ &= \sum_{n=0}^{N}\sum_{i=-n}^{n}\sum_{j=-n}^{n} D^n_{i,j}(\alpha,\beta,\gamma)\widetilde{g}^{(k,N)}_{n,j}(R')Y_{n,i}(\eta) \end{aligned} \tag{3.9}$$

if $R = A(\alpha,\beta,\gamma)R',\ R,\ R' \subseteq \Omega$.

Proof. We start with a representation of g^k similar to (3.6) only in complex spherical harmonics and insert (3.8) into it.

$$\begin{aligned} g^{(k,N)}(R,\eta) &= \sum_{n=0}^{N}\sum_{j=-n}^{n} \widetilde{g}^{(k,N)}_{n,j}(R')Y_{n,j}\left(A\left(\alpha,\beta,\gamma\right)^{\mathrm{T}}\eta\right) \\ &= \sum_{n=0}^{N}\sum_{i=-n}^{n}\sum_{j=-n}^{n} \widetilde{g}^{(k,N)}_{n,j}(R')D^n_{i,j}(\alpha,\beta,\gamma)Y_{n,i}(\eta) \\ &:= \sum_{n=0}^{N}\sum_{i=-n}^{n} \widetilde{g}^{(k,N)}_{n,i}(R)Y_{n,i}(\eta). \end{aligned}$$

□

Note that the coefficients $\widetilde{g}^{(k,N)}_{n,j}$ denote the coefficients with respect to an expansion in complex spherical harmonics. These coefficients are related to the coefficients $g^{(k,N)}_{n,j}$ from Definition 3.1.3 as described in Remark 2.2.9.

Example 3.1.10. Considering in particular a spherical cap as the localization region R of a set of Slepian functions (see Example 3.1.5), we obtain the following notation from (3.9):

$$g^{(k,N)}\left(\left(c, A(\alpha,\beta,\gamma)\varepsilon^3\right),\cdot\right) = \sum_{n=0}^{N}\sum_{i=-n}^{n}\sum_{j=-n}^{n} D^n_{i,j}(\alpha,\beta,\gamma)\widetilde{g}^{(k,N)}_{n,j}\left(c,\varepsilon^3\right)Y_{n,i}(\cdot)$$

for $k \in \mathbb{N},\ k \leq (N+1)^2$ with a band-limit $N \in \mathbb{N}$ and for $R\left(c, A\left(\alpha,\beta,\gamma\right)\varepsilon^3\right) = A(\alpha,\beta,\gamma)R'\left(c,\varepsilon^3\right)$ with $A(\alpha,\beta,\gamma) \in \mathrm{SO}(3)$ and $c \in [-1,1]$.

We showed the relation of the case of an arbitrary localization region R with the case of a localization region R' of the same form but containing ε^3. Thus, for the next considerations we can work with solely the latter kind of regions.

A view on the computation and representation of Slepian functions In the beginning of this section, we mentioned stability problems emerging from the naive approach of computing Slepian functions. In detail, the numerical solution of the eigenvalue problem in the background of the Slepian functions can be problematic due to the high condition number of the localization matrix $L(R)$. However, if the Slepian functions are localized, e. g., in a spherical cap, we can circumvent this situation. We summarize how the numerical computations can be stabilized next. These results are taken from, e. g., Grünbaum et al. (1982); Seibert (2018); Simons (2010); Simons et al. (2006).
For the numerical computation of Slepian functions, we consider the following localization region:

$$R\left(c, \varepsilon^3\right) = \left\{ \eta \in \Omega \;\middle|\; \left| \eta - \varepsilon^3 \right| \leq c \right\} \tag{3.10}$$

which can be parametrized by

$$\overline{R}\left(c, \varepsilon^3\right) = \{ (\varphi, t) \mid \varphi \in [0, 2\pi[,\ t \in [c, 1] \}$$

for practical purposes. For this region, a commuting differential operator on the set of twice continuously differentiable functions on the sphere can be found. We look for the eigenfunctions of the commuting differential operator to bypass the ill-conditioned localization matrix $L(R(c, \varepsilon^3))$. These investigations show that the Fourier coefficients of an eigenfunction of the commuting operator are obtained as the coefficients of an eigenvector of a certain matrix $M(c)$ dependent on the size of the spherical cap $c \in [-1, 1]$. Furthermore, it can be shown that the matrix $M(c)$ commutes with the localization matrix $L(R(c, \varepsilon^3))$. The entries of the matrix $M(c)$ are $\mathrm{L}^2(\Omega)$-inner products of a spherical harmonic and the image of a spherical harmonic under the commuting differential operator. If we choose fully normalized spherical harmonics, the matrix $M(c)$ is given as follows, see, for instance, Grünbaum et al. (1982) or Seibert (2018, p. 212).

Theorem 3.1.11. *Let $N \in \mathbb{N}_0$ be the band-limit. Further, let $c \in [-1, 1]$ define the size of a spherical cap with centre ε^3. A $(N+1)^2 \times (N+1)^2$-dimensional matrix $M(c)$ with entries*

$$\begin{aligned}
M_{(n,j),(n,j)}(c) &= -n(n+1)c, \\
M_{(n,j),(n+1,j)}(c) &= (n(n+2) - N(N+2)) \sqrt{\frac{(n+1-j)(n+1+j)}{(2n+3)(2n+1)}}, \\
M_{(n+1,j),(n,j)}(c) &= M_{(n,j),(n+1,j)}, \\
M_{(m,k),(n,j)}(c) &= 0, \ \textit{else},
\end{aligned}$$

commutes with the localization matrix $L(R(c, \varepsilon^3))$ from (3.3) *if $R(c, \varepsilon^3)$ is given by* (3.10) *and fully normalized spherical harmonics are used in the representation* (3.4) *of the Slepian functions.*

Obviously, $M(c)$ is real and symmetric. If the rows and columns are ordered for fixed $j \in \mathbb{N}_0$ with $-n \leq j \leq n$ and running $n \in \mathbb{N}_0$, the matrix $M(c)$ is block-tridiagonal. Note that $L\left(R\left(c, \varepsilon^3\right)\right)$ also has a block structure if fully normalized spherical harmonics are used and the matrix is ordered analogously. However, $L\left(R\left(c, \varepsilon^3\right)\right)$ is not tridiagonal. As the matrices commute and due to their particular structures, it can be shown that they have identical eigenvectors (see e. g. Seibert, 2018, pp. 209-215). Therefore, for practical purposes, it suffices to compute the eigenvectors of the substitute matrix $M(c)$ to obtain Slepian functions which are localized in a spherical cap.
The computation of the eigenvectors of the matrix $M(c)$ is the next aspect we consider. Although there exist many algorithms for numerically solving algebraic eigenvalue problems, (see the usual literature on numerical mathematics, e. g., Hanke-Bourgeois, 2009; Schwarz and Köckler, 2011; Stoer and Bulirsch, 1973; Stummel and Hainer, 1971), we need a particular solution. The gradient-based learning algorithm needs an explicit representation of the eigenvectors of the matrix $M(c)$. For this, we first recall the Gauß algorithm for solving tridiagonal systems of linear equations (see e. g. Stummel and Hainer, 1971, pp. 120-121).

Theorem 3.1.12. *Let M be a real, symmetric, tridiagonal matrix of dimension $n \in \mathbb{N}_0$, i. e.*

$$\begin{pmatrix} a_{1,1} & a_{1,2} & & & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & & & a_{n,n-1} & a_{n,n} \end{pmatrix}.$$

An equivalent system of $Av = f$ is given by

$$v_j + b_{j,j+1} v_{j+1} = \widetilde{f}_j, \qquad v_n = \widetilde{f}_n$$

for $j \in \mathbb{N}_0$, $j = 1, ..., n-1$ and with

$$b_{1,2} = \frac{a_{1,2}}{a_{1,1}}, \qquad b_{j,j+1} = \frac{a_{j,j+1}}{a_{j,j} - a_{j,j-1} b_{j-1,j}},$$

$$\widetilde{f}_1 = \frac{f_1}{a_{1,1}}, \qquad \widetilde{f}_j = \frac{f_j - a_{j-1,j} \widetilde{f}_{j-1}}{a_{j,j} - a_{j,j-1} b_{j-1,j}}, \qquad \widetilde{f}_n = \frac{f_n - a_{n-1,n} \widetilde{f}_{n-1}}{a_{n,n} - a_{n,n-1} b_{n-1,n}}$$

for $j = 2, ..., n-1$. A solution of $Av = f$ is obtained by first computing the values $b_{j,j+1}$ and $\widetilde{f}_j$ for $j = 1, ..., n-1$ (increasing) and then computing v_j for $j = n-1, ..., 1$ (decreasing).

This means, for the solution of an eigenvalue problem as given in the computation of Slepian functions, we have the explicit representation of the eigenvectors as follows.

Theorem 3.1.13. *For $c \in [-1,1]$ and $N \in \mathbb{N}_0$, let $M^j(c)$ be the sub-block of $M(c)$ from Theorem 3.1.11 with respect to a fixed $j \in \mathbb{Z}$ with $|j| \leq N$ for the band-limit N of the Slepian functions under investigation. That means, the entries of $M^j(c)$ are given by $(M^j(c))_{(p,|j|),(q,|j|)}$ for $p,\ q = |j|, ..., N$. The matrix $M^j(c)$ is real, symmetric and tridiagonal. Further, $M^j(c)$ has the dimension $N - |j| + 1$. For an eigenvalue ρ of $M^j(c)$, we consider the system $(M^j(c) - \rho I)v = 0$ for the identity matrix $I \in \mathbb{R}^{(N-|j|+1)\times(N-|j|+1)}$ in order to determine a related eigenvector $v(c) \in \mathbb{R}^{N-|j|+1}$. The eigenvector v is given by*

$$v_i(c) + b_{i,i+1}(c)v_{i+1}(c) = 0, \qquad v_{N-|j|+1} \equiv \tau$$

for $i = 1, ..., N - |j|$, a degree of freedom $\tau \neq 0$ and with

$$b_{1,2}(c) = \frac{\left(M^j(c)\right)_{(|j|,|j|),(|j|+1,|j|)}}{\left(M^j(c)\right)_{(|j|,|j|),(|j|,|j|)} - \rho}$$

and

$$b_{i,i+1}(c) = \frac{\left(M^j(c)\right)_{(i,|j|),(i+1,|j|)}}{\left(M^j(c)\right)_{(i,|j|),(i,|j|)} - \rho - \left(M^j(c)\right)_{(i,|j|),(i-1,|j|)} b_{i-1,i}(c)}$$

for $i = 2, ..., N - |j|$. In practice, the eigenvector is obtained by first computing the values $b_{i,i+1}(c)$ for $j = 1, ..., N - |j|$ (increasing) and then computing v_i for $i = N - |j| + 1, ..., 1$ (decreasing).

Proof. The assumed characteristics of $M^j(c)$ are obtained straightforward from the definition of $M(c)$ as given in Theorem 3.1.11. The application of Theorem 3.1.12 to the matrix $M^j(c)$ gives the representation of its eigenvector. □

Note that it suffices to compute the eigenvectors of the matrices $M^j(c)$, for $j \in \mathbb{N}_0$ with $-N \leq j \leq N$, to obtain the eigenvectors of the full matrix $M(c)$ and, thus, of the matrix $L\left(R\left(c, \varepsilon^3\right)\right)$ because $M^j(c)$ represents one block in the rearranged matrix for each j.

3.2. An overview of spherical Sobolev spaces

Spherical harmonics form an orthonormal basis of $\mathrm{L}^2(\Omega)$. However, we want to consider another set of Hilbert spaces, so-called Sobolev spaces, as well because of practical and theoretical reasons. The local trial functions we want to consider in a dictionary are strongly related to these spaces. Thus, we introduce the Sobolev spaces first. These spaces generalize the concept of the function space $\mathrm{L}^2(\Omega)$. The section is based on Freeden et al. (1998); Michel (2013); Telschow (2014).

Definition 3.2.1. *For a real sequence $(A_n)_{n\in\mathbb{N}_0}$ with $A_n \neq 0$ for all $n \in \mathbb{N}_0$, we define a (spherical) Sobolev space $\mathcal{H}((A_n);\Omega)$ as the completion of the set*

$$\left\{ f \in \mathrm{C}^{(\infty)}(\Omega) \;\middle|\; \sum_{n=0}^{\infty} \sum_{j=-n}^{n} A_n^2 \langle f, Y_{n,j} \rangle_{\mathrm{L}^2(\Omega)}^2 < \infty \right\}.$$

Its inner product is correspondingly

$$\langle f, g \rangle_{\mathcal{H}((A_n);\Omega)} := \sum_{n=0}^{\infty} \sum_{j=-n}^{n} A_n^2 \langle f, Y_{n,j} \rangle_{\mathrm{L}^2(\Omega)} \langle g, Y_{n,j} \rangle_{\mathrm{L}^2(\Omega)}$$

for $f, g \in \mathcal{H}((A_n);\Omega)$.

Note that the space $\mathcal{H}((A_n);\Omega)$ is a generalization of the space $\mathrm{L}^2(\Omega)$ as we have $\mathcal{H}((1);\Omega) = \mathrm{L}^2(\Omega)$. Further note that these Sobolev spaces are not empty. Examples for elements of a Sobolev space $\mathcal{H}((A_n);\Omega)$ are for instance spherical harmonics.

Lemma 3.2.2. *Let $Y_{m,k}$, m, $k \in \mathbb{N}_0$ with $-m \leq k \leq m$, be a spherical harmonic. Then it holds that $Y_{m,k} \in \mathcal{H}((A_n);\Omega)$ for a real sequence $(A_n)_{n\in\mathbb{N}_0}$ with $A_n \neq 0$ for all $n \in \mathbb{N}_0$.*

Proof. We have to prove that

$$\sum_{n=0}^{\infty} \sum_{j=-n}^{n} A_n^2 \langle Y_{m,k}, Y_{n,j} \rangle_{\mathrm{L}^2(\Omega)}^2 < \infty.$$

With the use of (2.12), we obtain

$$\sum_{n=0}^{\infty} \sum_{j=-n}^{n} A_n^2 \langle Y_{m,k}, Y_{n,j} \rangle_{\mathrm{L}^2(\Omega)}^2 = \sum_{n=0}^{\infty} \sum_{j=-n}^{n} A_n^2 \left(\delta_{n,m}\delta_{j,k}\right)^2 = A_m^2 < \infty. \qquad \square$$

Remark 3.2.3. Note that also a Slepian function is an element of a Sobolev space $\mathcal{H}((A_n);\Omega)$ for a real sequence $(A_n)_{n\in\mathbb{N}_0}$ with $A_n \neq 0$. Let $g^{(p,M)}(R,\cdot)$ be a Slepian function of band-limit $M \in \mathbb{N}_0$. Then we have

$$\begin{aligned}
&\sum_{n=0}^{\infty} \sum_{j=-n}^{n} A_n^2 \left\langle g^{(p,M)}(R,\cdot), Y_{n,j} \right\rangle_{\mathrm{L}^2(\Omega)}^2 \\
&\qquad = \sum_{n=0}^{\infty} \sum_{j=-n}^{n} A_n^2 \left(\sum_{m=0}^{M} \sum_{k=-m}^{m} g_{m,k}^{(p,M)}(R) \langle Y_{m,k}, Y_{n,j} \rangle_{\mathrm{L}^2(\Omega)} \right)^2 \\
&\qquad = \sum_{n=0}^{M} \sum_{j=-n}^{n} A_n^2 \left(g_{n,j}^{(p,M)}(R) \right)^2 < \infty.
\end{aligned}$$

We have a more detailed look at the inner product of a Sobolev space. At first, we consider the case of the inner product of an arbitrary function and a spherical harmonic.

Lemma 3.2.4. *Let $f \in \mathcal{H}((A_n);\Omega)$ and $Y_{m,k}$, m, $k \in \mathbb{N}_0$ with $-m \leq k \leq m$, be a spherical harmonic. Then it holds*

$$\langle f, Y_{m,k}\rangle^2_{\mathcal{H}((A_n);\Omega)} = A_m^2 \langle f, Y_{m,k}\rangle^2_{\mathrm{L}^2(\Omega)}.$$

Proof. We start at the left-hand side. Using (2.12), we obtain

$$\begin{aligned}\langle f, Y_{m,k}\rangle^2_{\mathcal{H}((A_n);\Omega)} &= \sum_{n=0}^{\infty}\sum_{j=-n}^{n} A_n^2 \langle f, Y_{n,j}\rangle^2_{\mathrm{L}^2(\Omega)} \langle Y_{m,k}, Y_{n,j}\rangle^2_{\mathrm{L}^2(\Omega)} \\ &= \sum_{n=0}^{\infty}\sum_{j=-n}^{n} A_n^2 \langle f, Y_{n,j}\rangle^2_{\mathrm{L}^2(\Omega)} \delta_{n,m}\delta_{j,k} = A_m^2 \langle f, Y_{m,k}\rangle^2_{\mathrm{L}^2(\Omega)}. \qquad \square\end{aligned}$$

Now we consider the inner product of two arbitrary elements f, $g \in \mathcal{H}((A_n);\Omega)$ and obtain the following result.

Theorem 3.2.5. *For a real sequence $(A_n)_{n\in\mathbb{N}_0}$ with $A_n \neq 0$ for all $n \in \mathbb{N}_0$, the system*

$$\left\{A_n^{-1} Y_{n,j}\right\}_{n,\ j\in\mathbb{N}_0;\ j=-n,\ldots,n}$$

is an orthonormal basis of the Sobolev space $\mathcal{H}((A_n);\Omega)$.

Proof. The orthonormality is easily seen with the use of Lemma 3.2.4:

$$\left\langle A_n^{-1} Y_{n,j}, A_m^{-1} Y_{m,k}\right\rangle_{\mathcal{H}((A_n);\Omega)} = \langle Y_{n,j}, Y_{m,k}\rangle_{\mathrm{L}^2(\Omega)} = \delta_{n,m}\delta_{j,k}.$$

Due to Theorem 2.1.1, we can show the completeness by showing that the Parseval identity holds:

$$\begin{aligned}\langle f, g\rangle_{\mathcal{H}((A_n);\Omega)} &= \sum_{n=0}^{\infty}\sum_{j=-n}^{n} A_n^2 \langle f, Y_{n,j}\rangle^2_{\mathrm{L}^2(\Omega)} \langle g, Y_{n,j}\rangle^2_{\mathrm{L}^2(\Omega)} \\ &= \sum_{n=0}^{\infty}\sum_{j=-n}^{n} \frac{A_n^4}{A_n^2} \langle f, Y_{n,j}\rangle^2_{\mathrm{L}^2(\Omega)} \langle g, Y_{n,j}\rangle^2_{\mathrm{L}^2(\Omega)} \\ &= \sum_{n=0}^{\infty}\sum_{j=-n}^{n} A_n^{-2} \langle f, Y_{n,j}\rangle^2_{\mathcal{H}((A_n);\Omega)} \langle g, Y_{n,j}\rangle^2_{\mathcal{H}((A_n);\Omega)} \\ &= \sum_{n=0}^{\infty}\sum_{j=-n}^{n} \left\langle f, A_n^{-1} Y_{n,j}\right\rangle^2_{\mathcal{H}((A_n);\Omega)} \left\langle g, A_n^{-1} Y_{n,j}\right\rangle^2_{\mathcal{H}((A_n);\Omega)}.\end{aligned}$$

Thus, the Parseval identity holds and the system is complete. $\square$

Note that Slepian functions are not orthogonal with respect to the $\mathcal{H}((A_n);\Omega)$-inner product because of the additional values of sequence elements in each summand (compare also (3.28)).
A first non-trivial example for a Sobolev space is given by the choice of $A_n = (n+0.5)^s$, $s \in \mathbb{R}$ and $n \in \mathbb{N}_0$ (see e. g. Freeden et al., 1998, p. 83). In particular, we consider the Sobolev space $\mathcal{H}(((n+0.5)^2);\Omega)$ in more detail throughout this thesis (see e. g. Freeden and Michel, 2004a, p. 131).

Example 3.2.6. The spherical Sobolev space $\mathcal{H}_2(\Omega) := \mathcal{H}(((n+0.5)^2);\Omega)$ is the Sobolev space with

$$A_n = (n+0.5)^2, \qquad n \in \mathbb{N}_0,$$

i. e. the completion of the set of functions with

$$\left\{ f \in \mathrm{C}^{(\infty)}(\Omega) \;\middle|\; \sum_{n=0}^{\infty} \sum_{j=-n}^{n} (n+0.5)^4 \langle f, Y_{n,j} \rangle^2_{\mathrm{L}^2(\Omega)} < \infty \right\}.$$

Its inner product is correspondingly

$$\langle f, g \rangle_{\mathcal{H}_2(\Omega)} = \sum_{n=0}^{\infty} \sum_{j=-n}^{n} (n+0.5)^4 \langle f, Y_{n,j} \rangle^2_{\mathrm{L}^2(\Omega)} \langle g, Y_{n,j} \rangle^2_{\mathrm{L}^2(\Omega)}$$

for $f, g \in \mathcal{H}_2(\Omega)$.

We mentioned that the space $\mathrm{L}^2(\Omega)$ is identical with $\mathcal{H}((1);\Omega)$. Thus, the question arises how two Sobolev spaces are related. We formulate the obvious statement similar to Michel (2013, Theorem 6.7) and Telschow (2014, Theorem 2.8.5).

Theorem 3.2.7. *Let $(A_n)_{n\in\mathbb{N}_0}$ and $(B_n)_{n\in\mathbb{N}_0}$ be two real sequences with*
(i) $A_n \neq 0$ and $B_n \neq 0$ for all $n \in \mathbb{N}_0$ and
(ii) $|A_n| \leq |B_n|$ for all $n \in \mathbb{N}_0$ with $n \geq n_0 \in \mathbb{N}_0$.
Then it holds

$$\mathcal{H}((B_n);\Omega) \subseteq \mathcal{H}((A_n);\Omega).$$

In this context, we have a relation between $\mathrm{L}^2(\Omega)$ and $\mathcal{H}_2(\Omega)$.

Lemma 3.2.8. *Let $f \in \mathcal{H}_2(\Omega)$. Then it holds $f \in \mathrm{L}^2(\Omega)$.*

Proof. Set $A_n \equiv 1$ and $B_n = (n+0.5)^2$ in Lemma 3.2.8. □

Remark 3.2.9. Note that, in general, it does not hold that $f \in \mathcal{H}_2(\Omega)$ for all $f \in \mathrm{L}^2(\Omega)$. Consider the following counter example. Let $f \in \mathrm{L}^2(\Omega)$ with

$$\langle f, Y_{n,j} \rangle_{\mathrm{L}^2(\Omega)} := \frac{1}{(n+1)^2}.$$

Then it follows that

$$\|f\|^2_{\mathrm{L}^2(\Omega)} = \sum_{n=0}^{\infty}\sum_{j=-n}^{n} \langle f, Y_{n,j}\rangle^2_{\mathrm{L}^2(\Omega)} = \sum_{n=0}^{\infty}\sum_{j=-n}^{n} \frac{1}{(n+1)^4} = \sum_{n=0}^{\infty} \frac{2n+1}{(n+1)^4} < \infty$$

but

$$\begin{aligned}\|f\|^2_{\mathcal{H}_2(\Omega)} &= \sum_{n=0}^{\infty}\sum_{j=-n}^{n} (n+0.5)^4 \langle f, Y_{n,j}\rangle^2_{\mathrm{L}^2(\Omega)} = \sum_{n=0}^{\infty}\sum_{j=-n}^{n} \frac{(n+0.5)^4}{(n+1)^4} \\ &= \sum_{n=0}^{\infty} \frac{(2n+1)(n+0.5)^4}{(n+1)^4} = \infty.\end{aligned}$$

That means that if a function f is an element of $\mathcal{H}_2(\Omega)$, then its Fourier coefficients $\langle f, Y_{n,j}\rangle_{\mathrm{L}^2(\Omega)}$ must decrease in a way such that $\|f\|_{\mathcal{H}((A_n);\Omega)} < \infty$. However, for a $f \in \mathrm{L}^2(\Omega)$, we only have that it holds $\|f\|_{\mathrm{L}^2(\Omega)} < \infty$. Due to the additional factor $(n+0.5)^4$ in $\|\cdot\|_{\mathcal{H}_2(\Omega)}$, we cannot derive $\|f\|_{\mathcal{H}_2(\Omega)} < \infty$ from $\|f\|_{\mathrm{L}^2(\Omega)} < \infty$ in general. Thus, one can say, the elements of $\mathcal{H}_2(\Omega)$ must be "smoother" than an arbitrary element of $\mathrm{L}^2(\Omega)$.

We also obtain a relation between the norms of two nested Sobolev spaces.

Theorem 3.2.10. *Let $(A_n)_{n\in\mathbb{N}_0}$ and $(B_n)_{n\in\mathbb{N}_0}$ be two real sequences with*
(i) $A_n \neq 0$ and $B_n \neq 0$ for all $n \in \mathbb{N}_0$ and
(ii) $|A_n| \leq |B_n|$ for all $n \in \mathbb{N}_0$ with $n \geq n_0 \in \mathbb{N}_0$.
Then there exists a $c \in \mathbb{R}^+$ such that for any $f \in \mathcal{H}((B_n);\Omega)$ it holds

$$\|f\|_{\mathcal{H}((A_n);\Omega)} \leq \max\{c,1\}\|f\|_{\mathcal{H}((B_n);\Omega)}.$$

Proof. First of all, we separate $\mathcal{H}((A_n);\Omega)$ into

$$\begin{aligned}\mathcal{H}((A_n);\Omega) &= \mathrm{Harm}_{0,\dots,n_0}(\Omega) \oplus (\mathrm{Harm}_{0,\dots,n_0}(\Omega))^{\perp} \\ &= \mathrm{Harm}_{0,\dots,n_0}(\Omega) \oplus \overline{\mathrm{Harm}_{n_0+1,\dots,\infty}(\Omega)}^{\|\cdot\|_{\mathcal{H}((A_n);\Omega)}}.\end{aligned}$$

Thus, a function $f \in \mathcal{H}((A_n);\Omega)$ can be written as

$$f = f_{0,\dots,n_0} + (f_{0,\dots,n_0})^{\perp}.$$

Then we obtain

$$\begin{aligned}\|f\|^2_{\mathcal{H}((A_n);\Omega)} &= \|f_{0,\dots,n_0}\|^2_{\mathcal{H}((A_n);\Omega)} + \left\|(f_{0,\dots,n_0})^{\perp}\right\|^2_{\mathcal{H}((A_n);\Omega)} \\ &\leq c\|f_{0,\dots,n_0}\|^2_{\mathcal{H}((B_n);\Omega)} + \left\|(f_{0,\dots,n_0})^{\perp}\right\|^2_{\mathcal{H}((B_n);\Omega)} \\ &\leq \max\{c,1\}\|f\|^2_{\mathcal{H}((B_n);\Omega)}\end{aligned}$$

due to property (ii) and the fact that all norms are equivalent in the finite dimensional case. Note that because $f_{0,\dots,n_0}$ is a finite linear combination of spherical harmonics, the separation of f is also identical in both Sobolev spaces $\mathcal{H}((A_n);\Omega)$ and $\mathcal{H}((B_n);\Omega)$. □

The Sobolev spaces have a tight relation to the trial functions which we introduce in the next section. These functions are examples for reproducing kernels.

Definition 3.2.11. *For a Hilbert space $(H, \langle\cdot,\cdot\rangle_H)$ of functions on a domain $D \subseteq \mathbb{R}^n$, $n \in \mathbb{N}_0$, a function $R^H\colon D \times D \to \mathbb{R}$ is called reproducing kernel of H if the following properties are fulfilled:*

(i) $R^H(z,\cdot) \in H$ for all $z \in D$ and

(ii) $\langle R^H(z,\cdot), f\rangle_H = f(z)$ for all $f \in H$ and all $z \in D$.

Then the Hilbert space H is called a reproducing kernel Hilbert space.

Reproducing kernels can be used for instance in the construction of spherical splines. In this thesis, however, we concentrate on different aspects of reproducing kernels, namely that they are basis systems of certain Sobolev spaces. In particular, we consider Sobolev spaces whose corresponding sequences are called summable.

Definition 3.2.12. *A real sequence $(A_n)_{n\in\mathbb{N}_0}$ with $A_n \neq 0$ for all $n \in \mathbb{N}_0$ and with*

$$\sum_{n=0}^{\infty} \frac{2n+1}{4\pi} A_n^{-2} < \infty$$

is called summable.

Obviously, the sequence $A_n = (n+0.5)^2,\ n \in \mathbb{N}_0$, from the Sobolev space $\mathcal{H}_2(\Omega)$ (see Example 3.2.6) is summable.
Now we can relate a reproducing kernel to a Sobolev space as it is usually done in the literature (see e. g. Freeden et al., 1998, p. 90-91).

Theorem 3.2.13. *For a real, summable sequence $(A_n)_{n\in\mathbb{N}_0}$ with $A_n \neq 0$ for all $n \in \mathbb{N}_0$, the corresponding Sobolev space $\mathcal{H}((A_n);\Omega)$ is a reproducing kernel Hilbert space with the reproducing kernel*

$$k_{\mathcal{H}((A_n);\Omega)}(\chi,\eta) := \sum_{n=0}^{\infty} \frac{2n+1}{4\pi} A_n^{-2} P_n(\chi\cdot\eta)$$

for $\chi,\ \eta \in \Omega$.

For summable sequences, the Sobolev lemma gives us another embedding of Sobolev spaces. We recall a simplified version.

Lemma 3.2.14. ***(Sobolev lemma)*** *Let $(A_n)_{n\in\mathbb{N}_0}$ be a real summable sequence with $A_n \neq 0$ for all $n \in \mathbb{N}_0$. Then it holds*

$$\mathcal{H}((A_n);\Omega) \subseteq \mathrm{C}^{(0)}(\Omega).$$

Further, the $\mathrm{L}^2(\Omega)$-Fourier series of each $f \in \mathcal{H}((A_n);\Omega)$ is uniformly convergent.

For a proof, see, for instance, Michel (2013, pp. 154-155) and replace $(B_n)_{n\in\mathbb{N}_0}$ with the constant sequence of 1.
The reproducing kernels also form a basis in different spaces under certain assumptions.

Theorem 3.2.15. *Let $\mathcal{H}((A_n);\Omega)$ be a Sobolev space with a real, summable sequence $(A_n)_{n\in\mathbb{N}_0}$. Further, we consider a countable and dense set $X \subseteq \Omega$. The system of reproducing kernels*

$$\left\{ k_{\mathcal{H}((A_n);\Omega)}(x,\cdot) \;\middle|\; x \in X \right\}$$

is closed in the spaces

$$\left(\mathcal{H}((A_n);\Omega), \|\cdot\|_{\mathcal{H}((A_n);\Omega)}\right), \left(\mathrm{C}^{(0)}(\Omega), \|\cdot\|_{\mathrm{C}(\Omega)}\right) \text{ and } \left(\mathrm{L}^2(\Omega), \|\cdot\|_{\mathrm{L}^2(\Omega)}\right)$$

in the sense of the approximation theory.

For a proof, see, for instance, Michel (2013, p. 175-177). Furthermore, these systems of reproducing kernels relate nested Sobolev spaces in the following way.

Theorem 3.2.16. *Let $\mathcal{H}((A_n);\Omega)$ and $\mathcal{H}((B_n);\Omega)$ be Sobolev spaces with real, summable sequences $(A_n)_{n\in\mathbb{N}_0}$ and $(B_n)_{n\in\mathbb{N}_0}$. Further, for the sequences $(A_n)_{n\in\mathbb{N}_0}$ and $(B_n)_{n\in\mathbb{N}_0}$, it holds*

(i) $A_n \neq 0$ and $B_n \neq 0$ for all $n \in \mathbb{N}_0$ and
(ii) $|A_n| \leq |B_n|$ for all $n \in \mathbb{N}_0$ with $n \geq n_0 \in \mathbb{N}_0$.

At last, we consider a countable and dense set $X \subseteq \Omega$. The system of reproducing kernels

$$\left\{ k_{\mathcal{H}((B_n);\Omega)}(x,\cdot) \;\middle|\; x \in X \right\}$$

is closed in the space

$$\left(\mathcal{H}((A_n);\Omega), \|\cdot\|_{\mathcal{H}((A_n);\Omega)}\right)$$

in the sense of the approximation theory.

Proof. First of all, note that we have

$$\operatorname{span}\left\{ A_n^{-1} Y_{n,j} \;\middle|\; n \in \mathbb{N}_0; j = -n,...,n \right\} \subseteq \mathcal{H}((B_n);\Omega)$$

because only finite linear combinations are possibly additionally inserted by the span. However, we know that

$$\left\{ A_n^{-1} Y_{n,j} \right\}_{n \in \mathbb{N}_0;\ j=-n,\dots,n}$$

is a basis of $\mathcal{H}((A_n);\Omega)$, see Theorem 3.2.5. Thus, it yields

$$\overline{\operatorname{span}\left\{ A_n^{-1} Y_{n,j} \;\middle|\; n \in \mathbb{N}_0;\ j=-n,\dots,n \right\}}^{\mathcal{H}((A_n);\Omega)} = \mathcal{H}((A_n);\Omega).$$

Hence, together with Theorem 3.2.7, i. e. $\mathcal{H}((B_n);\Omega) \subseteq \mathcal{H}((A_n);\Omega)$, we can conclude that

$$\overline{\mathcal{H}((B_n);\Omega)}^{\mathcal{H}((A_n);\Omega)} = \mathcal{H}((A_n);\Omega).$$

That means: for all $\varepsilon > 0$ and for all $f \in \mathcal{H}((A_n);\Omega)$, there exists a $g \in \mathcal{H}((B_n);\Omega)$ such that we have

$$\|f - g\|_{\mathcal{H}((A_n);\Omega)} \leq \frac{\varepsilon}{2}. \tag{3.11}$$

Note that, due to Theorem 3.2.10, there exists a $c \in \mathbb{R}^+$ such that it holds

$$\|f\|_{\mathcal{H}((A_n);\Omega)} \leq \max\{c,1\} \|f\|_{\mathcal{H}((B_n);\Omega)}$$

for all $f \in \mathcal{H}((B_n);\Omega)$. Thus, due to Theorem 3.2.15, to the particular $g \in \mathcal{H}((B_n);\Omega)$ from (3.11), there exists a linear combination

$$\sum_{i=0}^{I} \alpha_i k_{\mathcal{H}((B_n);\Omega)}\left(\chi^{(i)}, \cdot\right)$$

such that it holds

$$\left\| g - \sum_{i=0}^{I} \alpha_i k_{\mathcal{H}((B_n);\Omega)}\left(\chi^{(i)}, \cdot\right) \right\|_{\mathcal{H}((B_n);\Omega)} \leq \frac{\varepsilon}{2\max\{c,1\}}.$$

Therefore, we have

$$\begin{aligned} &\left\| g - \sum_{i=0}^{I} \alpha_i k_{\mathcal{H}((B_n);\Omega)}\left(\chi^{(i)}, \cdot\right) \right\|_{\mathcal{H}((A_n);\Omega)} \\ &\quad \leq \max\{c,1\} \left\| g - \sum_{i=0}^{I} \alpha_i k_{\mathcal{H}((B_n);\Omega)}\left(\chi^{(i)}, \cdot\right) \right\|_{\mathcal{H}((B_n);\Omega)} \leq \frac{\varepsilon}{2} \end{aligned}$$

All in all, for the closedness in the sense of the approximation theory, we obtain that for all $\varepsilon > 0$ and for all $f \in \mathcal{H}((A_n);\Omega)$, there exists a $g \in \mathcal{H}((B_n);\Omega)$ and a linear combination $\sum_{i=0}^{I} \alpha_i k_{\mathcal{H}((B_n);\Omega)}(\chi_i, \cdot)$ such that

$$\begin{aligned}
&\left\| f - \sum_{i=0}^{I} \alpha_i k_{\mathcal{H}((B_n);\Omega)}\left(\chi^{(i)}, \cdot\right) \right\|_{\mathcal{H}((A_n);\Omega)} \\
&\quad \leq \|f - g\|_{\mathcal{H}((A_n);\Omega)} + \left\| g - \sum_{i=0}^{I} \alpha_i k_{\mathcal{H}((B_n);\Omega)}\left(\chi^{(i)}, \cdot\right) \right\|_{\mathcal{H}((A_n);\Omega)} \\
&\quad < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.
\end{aligned}$$

□

3.3. From radial basis functions to low pass filters

The idea of using dictionary elements for approximating a signal enables us to use a mixture of different kind of trial functions. We have already introduced spherical harmonics and Slepian functions for this reason. In this section, we introduce radial basis functions which enlarge the spectrum of possible dictionary elements to local ones. For more details, the reader is referred to, e. g., Freeden et al. (1998); Freeden and Michel (2004a); Freeden and Schreiner (2009); Michel (2013) on which this section is based. Note that in some literature the term (scalar) zonal functions is also used for radial basis functions.

Definition 3.3.1. *A (spherical) radial basis function is defined as a kernel function* $k_\bullet \colon \Omega \times \Omega \to \mathbb{R}$ *with the property*

$$k_\bullet(\xi, \eta) := \hat{k}_\bullet(|\xi - \eta|) = \widetilde{k}_\bullet(\xi \cdot \eta)$$

for a fixed $\xi \in \Omega$ *and all* $\eta \in \Omega$.

The latter equality is well-defined as for all $\xi, \eta \in \Omega$ it holds

$$|\xi - \eta|^2 = |\xi|^2 + |\eta|^2 - 2\xi \cdot \eta = 2(1 - \xi \cdot \eta).$$

In Definition 3.3.1, a radial basis function k depends on a fixed unit normal vector $\xi \in \Omega$ and its domain is the sphere. Thus, we call ξ the centre of the radial basis function. Then its value at a point η depends only on the distance of η to the centre ξ of $k_\bullet$. In this way, $k_\bullet$ can also be considered as a function on $[-1, 1]$. Note that the $\bullet$ is a replacement character at this point: we can further develop radial basis functions by means of an additional weight or scale which acts as a control parameter for the localization.

Definition 3.3.2. *For a scale* $s \in]0, S]$, $S \in \mathbb{R}^+$, *a (scaled) radial basis function is defined as a kernel function* $k_s \colon \Omega \times \Omega \to \mathbb{R}$ *with the property*

$$k_s(\xi, \eta) := \hat{k}_s(|\xi - \eta|) = \widetilde{k}_s(\xi \cdot \eta)$$

for a fixed $\xi \in \Omega$ *and all* $\eta \in \Omega$.

Note that a scaled radial basis function is a generalization of a radial basis function as the scale is irrelevant in the latter case (hence, the use of $\bullet$ in Definition 3.3.1). Thus, we will consider scaled radial basis functions in this thesis. However, in the sequel, we will refer to them only as radial basis functions. First of all, we note the particular case in which the range of the scale function is bounded.

Lemma 3.3.3. *A radial basis function* $k_s\colon \Omega \times \Omega \to \mathbb{R}$ *can be reformulated as a function* $k\colon \mathbb{B}_S \times \Omega \to \mathbb{R}$ *on the ball* $\mathbb{B}_S$ *of radius* S *and centre* 0 *by*

$$k(x,\eta) \coloneqq k(s\xi,\eta) \coloneqq k_s(\xi,\eta)$$

for fixed $\xi \in \Omega$, $x = s\xi \in \mathbb{B}_S$ *and for all* $\eta \in \Omega$.

Proof. The reformulation is done using $x = s\xi$. □

A well-known example of a radial basis function is the Abel–Poisson kernel (see e. g. Freeden et al., 1998, p. 109).

Example 3.3.4. The Abel–Poisson kernel is defined as

$$K_h(\xi,\eta) \coloneqq \frac{1}{4\pi}\frac{1-h^2}{(1+h^2-2h\xi\cdot\eta)^{3/2}}$$

for all $h \in [0,1[$, and $\xi,\eta \in \Omega$. Equivalently, we can write

$$K(x,\eta) \coloneqq \frac{1}{4\pi}\frac{1-|x|^2}{(1+|x|^2-2x\cdot\eta)^{3/2}} \tag{3.12}$$

for all $x \in \mathring{\mathbb{B}}$ and $\eta \in \Omega$. Note the similarity to the kernel of the Fredholm integral operator of the first kind given in (2.28).
$\mathring{\mathbb{B}}$ emphasizes that the Abel–Poisson kernel is well-defined only in the interior of the ball. Examples of Abel–Poisson kernels are given in Figure 3.2. There, four Abel–Poisson kernels are presented. Blue colour stands for minimal values and red colour for maximal ones. All kernels have the same centre ξ, but have different scales. Shown are an Abel–Poisson kernel with scale 0.7, 0.8, 0.9 and 0.97 (from left to right). We notice that each kernel is a localized function. It attains only one extremum, has decreasing values in a neighbourhood of its centre and nearly vanishes on the outside of this region. In this way, a radial basis function has the form of a "hat" situated at its centre. Further, we see that the scale controls the level of localization. If we compare the plot on the left-hand side with the one on the right-hand side, we see that the higher h is the stronger the function is localized.

Remark 3.3.5. We emphasize the following aspects:

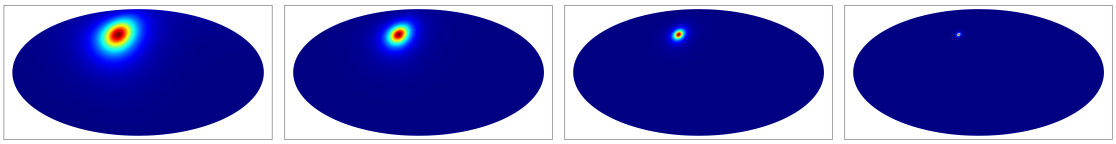


Figure 3.2.: Abel–Poisson kernels with a fixed centre ξ in all plots but different scales: $h = 0.7,\ 0.8,\ 0.9,\ 0.97$ (from left to right).

(a) Note the structure in our notation. A radial basis function $k_s(\xi, \cdot)$ is defined on the sphere Ω and with an explicit scale $s \in]0, S]$, $S \in \mathbb{R}^+$. If we consider its formulation on the ball, we use $k(x, \cdot)$ because the scale is implicitly given by its argument $x = s\xi \in \mathbb{B}_S$. For the specific Abel-Poisson kernel, we use the capital $K(x, \cdot)$.

(b) In the sequel, we are interested in square-integrable kernels. As we will see, the Abel–Poisson kernel is an example for such a kernel.

Next, we summarize an important property of radial basis functions which will be a recurring topic in the sequel. With radial basis functions, we are able to define a spherical convolution. We follow the line of Narcowich and Ward (1996) and Freeden and Michel (2004a, p. 26) for this.

Definition 3.3.6. *For functions $g,\ h \in \mathrm{L}^2([-1,1])$ and a function $f \in \mathrm{L}^2(\Omega)$, a spherical convolution is defined by*

$$(g * f)(\eta) := \int_\Omega g(\eta \cdot \chi) f(\chi)\, \mathrm{d}\omega(\chi),$$
$$(g * h)(\eta, \xi) := \int_\Omega g(\eta \cdot \chi) h(\xi \cdot \chi)\, \mathrm{d}\omega(\chi)$$

for all $\eta,\ \xi \in \Omega$. Further, the Legendre coefficient of g is given as

$$g^\wedge(n) := 2\pi \int_{-1}^{1} g(t) P_n(t)\, \mathrm{d}t. \tag{3.13}$$

For specific functions f, the following results can be derived (see e. g. Freeden and Gutting (2013, pp. 157-159) and Michel (2013, pp. 185-190)).

Theorem 3.3.7. *For $n \in \mathbb{N}_0$, we consider the n-th Legendre polynomial P_n as well as a spherical harmonic Y_n of degree n. Further, let $g,\ f \in \mathrm{L}^p([-1,1])$ and $\eta \in \Omega$. We have*

$$\begin{aligned}(g * P_n)(\eta, \xi) &= \int_\Omega g(\eta \cdot \chi) P_n(\xi \cdot \chi)\, \mathrm{d}\omega(\chi) = \langle g(\eta \cdot), P_n(\xi \cdot) \rangle_{\mathrm{L}^2(\Omega)} \\ &= g^\wedge(n) P_n(\eta \cdot \xi), \\ (g * Y_n)(\eta) &= \int_\Omega g(\eta \cdot \chi) Y_n(\chi)\, \mathrm{d}\omega(\chi) = \langle g(\eta \cdot), Y_n \rangle_{\mathrm{L}^2(\Omega)} \\ &= g^\wedge(n) Y_n(\eta),\end{aligned}$$

$$(g * f)(\eta, \xi) = \int_\Omega g(\eta \cdot \chi) f(\xi \cdot \chi) \, \mathrm{d}\omega(x) = \langle g(\eta \cdot), f(\xi \cdot) \rangle_{\mathrm{L}^2(\Omega)} = g^\wedge(n) f^\wedge(n),$$
$$(g * g)(\eta, \xi) = \int_\Omega g(\eta \cdot \chi) g(\xi \cdot \chi) \, \mathrm{d}\omega(x) = \langle g(\eta \cdot), g(\xi \cdot) \rangle_{\mathrm{L}^2(\Omega)} = \left(g^\wedge(n)\right)^2.$$

The first equation is also called the Funk-Hecke formula.

With this, we obtain the Legendre series of a radial basis function in spherical harmonics (see e. g. Freeden and Schreiner, 2009, pp. 340-341).

Theorem 3.3.8. *For all $\eta, \xi \in \Omega$ and a scale $s \in]0, S]$, $S \in \mathbb{R}^+$, a square-integrable radial basis function $k_s \colon \Omega \times \Omega \to \mathbb{R}$ has the expansion*

$$k_s(\xi, \eta) = \sum_{n=0}^{\infty} \frac{2n+1}{4\pi} k_s^\wedge(n) P_n(\xi \cdot \eta)$$

for the n-th Legendre polynomial P_n.

Proof. The Fourier expansion of k_s is given as follows.

$$\begin{aligned}
k_s(\xi, \eta) &= \sum_{n=0}^{\infty} \sum_{j=-n}^{n} \langle k_s(\xi, \cdot), Y_{n,j} \rangle_{\mathrm{L}^2(\Omega)} Y_{n,j}(\eta) \\
&= \sum_{n=0}^{\infty} \sum_{j=-n}^{n} \int_\Omega k_s(\xi, \chi) Y_{n,j}(\chi) \, \mathrm{d}\omega(\chi) Y_{n,j}(\eta) \\
&= \sum_{n=0}^{\infty} \int_\Omega k_s(\xi, \chi) \sum_{j=-n}^{n} Y_{n,j}(\chi) Y_{n,j}(\eta) \, \mathrm{d}\omega(\chi) \\
&= \sum_{n=0}^{\infty} \frac{2n+1}{4\pi} \int_\Omega k_s(\xi, \chi) P_n(\chi \cdot \eta) \, \mathrm{d}\omega(\chi) \\
&= \sum_{n=0}^{\infty} \frac{2n+1}{4\pi} k_s^\wedge(n) P_n(\xi \cdot \eta),
\end{aligned}$$

where the addition theorem for spherical harmonics from Theorem 2.2.4 and the Funk-Hecke formula from Theorem 3.3.7 are used. □

The expansion of an Abel–Poisson kernel in Legendre polynomials can be derived independently of the computation of $k_s^\wedge(n)$ as given in (3.13). This yields the following result (see e. g. Michel, 2013, Theorem 3.16 and Example 6.23).

Lemma 3.3.9. *For all $\eta \in \Omega$ and a point $x = h\xi \in \mathring{\mathbb{B}}$, an Abel–Poisson kernel $K(x, \eta)$ has the expansion*

$$K(x, \eta) = \frac{1}{4\pi} \frac{1 - |x|^2}{(1 + |x|^2 - 2x \cdot \eta)^{3/2}} = \sum_{n=0}^{\infty} \frac{2n+1}{4\pi} |x|^n P_n\left(\frac{x}{|x|} \cdot \eta\right)$$

for the n-th Legendre polynomial P_n.

Note that it obviously holds

$$K^\wedge(n) = |x|^n = h^n \tag{3.14}$$

for the Abel–Poisson kernel K. Thus, radial basis functions are built to be spatially localized functions, but, in general, are not well-localized in frequency. Hence, for practical purposes, the Abel–Poisson kernel has an advantage due to its closed form as given in Example 3.3.4.

Further characteristic: multiresolution analysis Polynomials like the spherical harmonics are global functions which are perfectly localized in the frequency domain. However, radial basis functions are generally not localized in the frequency domain as we have seen in Theorem 3.3.8. Nonetheless, they have some advantages for their use in approximating a signal. For instance, for a dictionary, we are interested in radial basis functions that are so-called low pass filters as they enable a multiresolution analysis. However, the construction of low pass filters takes several steps. We first define scaling functions.

Definition 3.3.10. *For $S \in \mathbb{R}^+$, let $\{k_s\}_{s\in]0,S]} \subseteq \mathrm{L}^2(\Omega \times \Omega)$ be a family of square-integrable radial basis functions. This family is called a scaling function if the Legendre coefficients $k_s^\wedge(n)$ fulfil:*
(S1) For $s_1 < s_2$, it holds $k_{s_1}^\wedge(n) \geq k_{s_2}^\wedge(n)$ for all $n \in \mathbb{N}_0$.
(S2) For all $n \in \mathbb{N}_0$, the limit of $(k_s^\wedge(n))_{s\in\,]0,S]}$ with respect to $s \searrow 0$ equals 1.
(S3) The Legendre coefficient $k_s^\wedge(n)$ is non-negative for all $n \in \mathbb{N}_0$ and $s \in \mathbb{R}^+$.

That means a scaling function is a particular system of radial basis functions with non-negative scales. Note that we call the whole system of functions a scaling function. In the sequel, we will not distinguish between a single function and the system of functions and will name both a scaling function. It will be emphasized which one is meant at the respective points.
The question arises how to construct a scaling function. For this, a generator of a scaling function is defined in, e. g., Freeden et al. (1998, Definition 11.1.1).

Definition 3.3.11. *A generator of a scaling function is a function $\kappa\colon \mathbb{R}_0^+ \to \mathbb{R}$ which is admissible, i. e.*

$$\sum_{n=0}^{\infty} \frac{2n+1}{4\pi} \left(\sup_{\tau\in[n,n+1[} |\kappa(\tau)| \right)^2 < \infty$$

and fulfils:
(GS1) $\kappa(0) = 1$.
(GS2) κ is continuous at 0.
(GS3) κ is monotonically decreasing.

Having a generator for a scaling function, we can construct the latter one as follows (see e. g. Freeden and Michel, 2004a, Section 3.5.2).

Theorem 3.3.12. *Let κ be a generator of a scaling function. Then a scaling function $\{k_s\}_{s\in]0,S]} \subseteq \mathrm{L}^2(\Omega\times\Omega)$, $S\in\mathbb{R}^+$, is obtained by setting its Legendre coefficients to*

$$k_s^\wedge(n) := \kappa(sn)$$

for all $n\in\mathbb{N}_0$ and $s\in\,]0,S]$.

We consider a first simple example of a scaling function, see, e. g., Schreiner (1996) or Freeden et al. (1998, p. 295).

Example 3.3.13. An example of a scaling function is the Cubic Polynomial Scaling Function which is constructed through the generator

$$\kappa(\tau) := \begin{cases} (1-\tau)^2(1+2\tau), & 0\le\tau<1,\\ 0, & 1\le\tau. \end{cases}$$

Its admissibility as well as property (GS1) is obvious. As the generator is a product of continuous functions on $[0,1[$, property (GS2) is also fulfilled. The monotonicity with respect to $\tau\in[0,1[$ needs to be clarified. For this, we consider the derivative of κ with respect to τ. We obtain

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\kappa(\tau) = \begin{cases} \tau(6\tau-6), & 0\le\tau<1,\\ 0, & 1\le\tau. \end{cases}$$

As $6\tau-6<0$ if $\tau\in[0,1[$, the gradient of κ is non-positive and, thus, κ is monotonically decreasing. All in all, κ is a generator of a scaling function.

If we are given a scaling function, we obtain a spherical multiresolution analysis (for the scale discrete case, see e. g. Michel, 2013, pp. 207-209).

Theorem 3.3.14. *For $S\in\mathbb{R}^+$, let $\{k_s\}_{s\in]0,S]}\subseteq\mathrm{L}^2(\Omega\times\Omega)$ be a scaling function. We define the sets*

$$\begin{aligned} V_s &:= \left\{ k_s * k_s * f \;\middle|\; f\in\mathrm{L}^2(\Omega)\right\}\\ &= \left\{ \sum_{n=0}^{\infty}\sum_{j=-n}^{n} \left(k_s^\wedge(n)\right)^2 f^\wedge(n,j)Y_{n,j} \;\middle|\; f\in\mathrm{L}^2(\Omega)\right\}. \end{aligned}$$

Then the spaces V_s, $s\in\,]0,S]$, represent a multiresolution analysis. That means, it holds:
(MRA1) For all $s_1,\ s_2\in\,]0,S]$ with $s_1<s_2$ it holds $V_{s_2}\subset V_{s_1}\subset\mathrm{L}^2(\Omega)$.
(MRA2) The union of the sets V_s is dense in $\mathrm{L}^2(\Omega)$, i. e.

$$\overline{\bigcup_{s\in]0,S]} V_s}^{\|\cdot\|_{\mathrm{L}^2(\Omega)}} = \mathrm{L}^2(\Omega).$$

Note that the multiresolution analysis also holds for $f \in \mathcal{H}_2(\Omega) \subseteq \mathrm{L}^2(\Omega)$. Furthermore, we want to emphasize that the convolution of two scaling functions k_{s_1} and k_{s_2}, $s_1,\ s_2 \in]0, S]$, can be obtained via their Legendre coefficients as it holds

$$(k_{s_1} * k_{s_2})^\wedge (n) = k_{s_1}^\wedge(n) k_{s_2}^\wedge(n),$$

see Theorem 3.3.7. Thus, we immediately obtain that the convolution $k_{s_1} * k_{s_2}$ of two scaling functions k_{s_1} and k_{s_2} is again a scaling function.
The multiresolution analysis gives us a hint on the particular scaling functions that are sensible in a dictionary. The space V_{s_2} contains the part of f that remains after applying the convolution with the scaling function k_{s_2} twice to f. Due to (MRA1), this part contains less information than if the convolution with the scaling function k_{s_1} is twice applied to f if $s_1 < s_2$. Thus, we see that the convolution $k_s * k_s$ acts as a filter on a signal f, e. g. the gravitational potential. For example, if the Legendre coefficients $k_s^\wedge(n)$ are nearly zero for all $n \in \mathbb{N}_0$ with $n \geq n_0 \in \mathbb{N}_0$, then the convolution represents a so-called low pass filter of a signal.

Definition 3.3.15. *For $S \in \mathbb{R}^+$, let $\{k_s\}_{s\in]0,S]} \subseteq \mathrm{L}^2(\Omega \times \Omega)$ be a scaling function. If for all $s \in]0, S]$ and $\varepsilon > 0$, there exists a $n_0 \in \mathbb{N}_0$ such that for all $n \geq n_0$ it holds*

$$k_s^\wedge(n) < \varepsilon,$$

then the system $\{k_s * k_s\}_{s\in]0,S]}$ is called a low pass filter.

Note that, again, we call the whole system as well as a member of this system a low pass filter in the sequel.

Theorem 3.3.16. *For all $\eta \in \Omega$, a centre $\xi \in \Omega$ and a scale $s \in]0, S]$, $S \in \mathbb{R}^+$, a low pass filter $k_s * k_s \colon \Omega \times \Omega \to \mathbb{R}$ has the expansion*

$$(k_s * k_s)(\xi, \eta) = \sum_{n=0}^{\infty} \frac{2n+1}{4\pi} \left(k_s^\wedge(n)\right)^2 P_n(\xi \cdot \eta)$$

for the n-th Legendre polynomial P_n.

Proof. Combine Theorems 3.3.7 and 3.3.8. □

Next, we give an example of a low pass filter which we will use in our dictionary (see e. g. Freeden and Michel, 2004a, p. 432).

Example 3.3.17. We consider the generator of a scaling function $\kappa \colon \mathbb{R}_0^+ \to \mathbb{R}$

$$\kappa(\tau) := \exp(-\tau).$$

It is clear that this is a generator of a scaling function. Due to Theorem 3.3.12, the Legendre coefficients of the scaling function equal

$$k_s^\wedge(n) = \exp(-sn) = (\exp(-s))^n$$

for $n \in \mathbb{N}_0$. As we have seen in (3.14), these are the Legendre coefficients of the Abel–Poisson kernel if we use

$$h := \exp(-s).$$

Therefore, a scaling function is given by

$$\{K(\exp(-s)\xi, \cdot)\}_{s \in \mathbb{R}^+}$$

for $\xi \in \Omega$. The respective low pass filters are obtained by

$$\left\{K\left(\exp\left(-s^2\right)\xi, \cdot\right)\right\}_{s \in \mathbb{R}^+} = \{K(\exp(-s)\xi, \cdot)\}_{s \in \mathbb{R}^+}$$

for $\xi \in \Omega$. Hence, we use the Abel–Poisson low pass filter given by

$$\{K(h\xi, \cdot)\}_{h \in [0,1[} = \{K(x, \cdot)\}_{x \in \mathring{\mathbb{B}}}$$

in the sequel. That means, for a dictionary element, we do not have to bear in mind the convolution of two radial basis functions in this case because the usual Abel–Poisson kernel is already a low pass filter.

Some aspects of low pass filters and Sobolev spaces We have seen in Theorem 3.2.13 that a Sobolev space $\mathcal{H}((A_n); \Omega)$ is a reproducing kernel Hilbert space with the reproducing kernel

$$k_{\mathcal{H}((A_n);\Omega)}(\chi, \eta) := \sum_{n=0}^{\infty} \frac{2n+1}{4\pi} A_n^{-2} P_n(\chi \cdot \eta)$$

for $\chi, \eta \in \Omega$ and if $(A_n)_{n \in \mathbb{N}_0}$ is summable and $A_n \neq 0$ for all $n \in \mathbb{N}_0$. If we compare this with Theorem 3.3.16, we see that a low pass filter is the reproducing kernel of the Sobolev space with

$$A_n = \left(k_s^\wedge(n)^2\right)^{-1/2} = \left(k_s^\wedge(n)\right)^{-1}, \qquad n \in \mathbb{N}_0,$$

for a fixed $s \in]0, S]$, $S \in \mathbb{R}^+$, if the sequence is summable as well as $k_s^\wedge(n) \neq 0$ and $A_n \neq 0$ for all $n \in \mathbb{N}_0$. The summability is always given for a scaling function obtained by a generator of a scaling function.
In the case of the Abel–Poisson low pass filters, we obtain due to Lemma 3.3.9 the spaces $\mathcal{H}((A_n); \Omega)$ with

$$A_n = h^{-n}.$$

Note that, in this case, the sequence $(A_n)_{n \in \mathbb{N}_0} = (h^{-n})_{n \in \mathbb{N}_0}$ is summable due to $h \in [0, 1[$ by definition. By virtue of Theorem 3.2.15, we obtain the following result

Lemma 3.3.18. *For $S \in \mathbb{R}^+$, let $s \in]0, S]$ be fixed and $\Xi \subset \Omega$ be dense and countable. If it holds*

$$k_s^\wedge(n) \neq 0$$

for all $n \in \mathbb{N}_0$ and the sequence $\left(\left(k_s^\wedge(n)\right)^{-1}\right)_{n\in\mathbb{N}_0}$ is summable, then the system of low pass filters

$$\{(k_s * k_s)(\xi, \cdot) \mid \xi \in \Xi\}$$

is closed in the space $(\mathrm{L}^2(\Omega), \|\cdot\|_{\mathrm{L}^2(\Omega)})$ in the sense of the approximation theory.

Proof. Apply Theorems 3.2.13, 3.2.15 and 3.3.16. □

In particular, we obtain the following result.

Lemma 3.3.19. *For a value $h \in [0, 1[$ and a dense and countable set $\Xi \subset \Omega$, the system*

$$\{K(h\xi, \cdot) \mid \xi \in \Xi\} = \left\{K(x, \cdot) \;\middle|\; \frac{x}{h} \in \Xi\right\}$$

of Abel–Poisson kernels is closed in the space $(\mathrm{L}^2(\Omega), \|\cdot\|_{\mathrm{L}^2(\Omega)})$ in the sense of the approximation theory.

Proof. Apply Lemma 3.3.18. □

At last, we consider the relation of a low pass filter and the particular space $\mathcal{H}_2(\Omega)$. In general, we obtain the following result.

Lemma 3.3.20. *For $S \in \mathbb{R}^+$, let $s \in]0, S]$ be fixed and $\Xi \subset \Omega$ be dense and countable. If it holds*

$$k_s^\wedge(n) \neq 0$$

for all $n \in \mathbb{N}_0$,

$$(n + 0.5)^2 \leq \left|\left(k_s^\wedge(n)\right)^{-1}\right|$$

for all $n \in \mathbb{N}_0$ with $n \geq n_0 \in \mathbb{N}_0$ and the sequence $\left(\left(k_s^\wedge(n)\right)^{-1}\right)_{n\in\mathbb{N}_0}$ is summable, then the system of low pass filters

$$\{(k_s * k_s)(\xi, \cdot) \mid \xi \in \Xi\}$$

is closed in the space $(\mathcal{H}_2(\Omega), \|\cdot\|_{\mathcal{H}_2(\Omega)})$ in the sense of the approximation theory.

Proof. Apply Theorems 3.2.13, 3.2.16 and 3.3.16. □

In particular for the Abel–Poisson kernels, this yields the following.

Lemma 3.3.21. *For a fixed $h \in [0, 1[$, the system of Abel–Poisson kernels*

$$\{K(h\xi, \cdot) \mid \xi \in \Xi\} = \left\{K(x, \cdot) \;\middle|\; \frac{x}{h} \in \Xi\right\},$$

where $\Xi \subset \Omega$ is dense and countable, is closed in the space $(\mathcal{H}_2(\Omega), \|\cdot\|_{\mathcal{H}_2(\Omega)})$ in the sense of the approximation theory.

Proof. Apply Lemma 3.3.20. □

3.4. Radial basis wavelets as band pass filters

In Section 3.3, we introduced low pass filters based on radial basis functions. Low pass filters extract the information of the signal which is below a certain threshold in the range of the frequency. However, it is also interesting to insert band pass filters in a dictionary as well because such filters yield the information of a signal which lies between two thresholds. Band pass filters are mathematically described by wavelets. This section is based on Freeden et al. (1998); Freeden and Michel (2004a); Michel (2013). For further literature on spherical wavelets, see, e. g., Freeden and Schreiner (1998); Freeden and Windheuser (1996); Windheuser (1995).

Definition 3.4.1. *Let $S \in \mathbb{R}^+$. Further, let $\{k_s\}_{s\in\,]0,S]} \subset \mathrm{L}^2(\Omega\times\Omega)$ be a scaling function. If the Legendre coefficients of the families $\{w_s\}_{s\in\,]0,S]}$ and $\{\widetilde{w}_s\}_{s\in\,]0,S]}$ fulfil*

$$w_s^\wedge(n)\widetilde{w}_s^\wedge(n) = \left(k_{s/2}^\wedge(n)\right)^2 - \left(k_s^\wedge(n)\right)^2$$

for all $n \in \mathbb{N}_0$ and $s \in]0,S]$, we call these families a (scaled) primal and dual radial basis wavelet, respectively. The radial basis wavelets w_S and $\widetilde{w}_S$ are called (scaled) radial basis mother wavelet. Further, we set $w_{2S} := \widetilde{w}_{2S} := k_S$. The solution

$$w_s^\wedge(n) := \widetilde{w}_s^\wedge(n) := \sqrt{\left(k_{s/2}^\wedge(n)\right)^2 - \left(k_s^\wedge(n)\right)^2}$$

for all $n \in \mathbb{N}_0$ and $s \in]0,S]$ is called the (scaled) radial basis P-wavelet. The solution

$$w_s^\wedge(n) := k_{s/2}^\wedge(n) - k_s^\wedge(n), \quad \widetilde{w}_s^\wedge(n) := k_{s/2}^\wedge(n) + k_s^\wedge(n)$$

for all $n \in \mathbb{N}_0$ and $s \in]0,S]$ is called the (scaled) radial basis M-wavelet.

Similar to radial basis functions, also radial basis wavelets can be constructed via a generating function (see e. g. Michel, 2013, Theorem 7.24).

Definition 3.4.2. *Let $\kappa\colon \mathbb{R}_0^+ \to \mathbb{R}$ be a generator of a scaling function. Further, let the functions $\psi,\ \widetilde{\psi}\colon \mathbb{R}_0^+ \to \mathbb{R}$ be admissible and fulfil*

$$\psi(\tau)\widetilde{\psi}(\tau) = \left(\kappa\left(\frac{\tau}{2}\right)\right)^2 - \left(\kappa(\tau)\right)^2$$

for all $\tau \in \mathbb{R}_0^+$. Then ψ and $\widetilde{\psi}$ are called generators of a primal and a dual radial basis mother wavelet, respectively.

Theorem 3.4.3. *Let $\psi,\ \widetilde{\psi}\colon \mathbb{R}_0^+ \to \mathbb{R}$ be generators of a primal and a dual radial basis mother wavelet, respectively, and $\kappa\colon \mathbb{R}_0^+ \to \mathbb{R}$ the related generator of a scaling function. Then, for $S \in \mathbb{R}^+$, the families $\{w_s\}_{s\in\,]0,S]}$ and $\{\widetilde{w}_s\}_{s\in\,]0,S]}$ obtained via*

$$w_s^\wedge(n) := \psi(sn), \quad \widetilde{w}_s^\wedge(n) := \widetilde{\psi}(sn), \quad w_{2S}^\wedge(n) := \widetilde{w}_{2S}^\wedge(n) := \kappa(Sn)$$

for all $n \in \mathbb{N}_0$ and $s \in \mathbb{R}^+$ are a primal and a dual radial basis wavelet, respectively.

Note that in our approach wavelets are based on a radial basis function. Thus, by construction, they are functions $w_s(\zeta,\eta),\ \widetilde{w}_s(\zeta,\eta) \in \mathbb{R}$ with a fixed argument $\zeta \in \Omega$ for all $\eta \in \Omega$. Further, their value at a point $\eta \in \Omega$ also depends only on the distance between ζ and η. Thus, in analogy to radial basis functions, we call ζ the centre of the primal and dual radial basis wavelet, respectively. Moreover, the wavelets inherit the dependence on the scale s from the corresponding radial basis function. The scale is still a measure for their localization. Again, we can reformulate the wavelets with respect to their domain.

Theorem 3.4.4. *For $S \in \mathbb{R}^+$, let the systems $\{w_s\}_{s\in\,]0,S]}$ and $\{\widetilde{w}_s\}_{s\in\,]0,S]}$ be a primal and dual radial basis wavelet, respectively. The wavelets can be written as functions $w,\ \widetilde{w}\colon \mathbb{B}_a \times \Omega \to \mathbb{R}$ on the ball $\mathbb{B}_a$ with radius $a \in \mathbb{R}^+$ by*

$$w(x,\eta) := w\left(s\zeta,\eta\right) := w_s(\zeta,\eta), \qquad \widetilde{w}(x,\eta) := \widetilde{w}\left(s\zeta,\eta\right) := \widetilde{w}_s(\zeta,\eta)$$

for all $\eta \in \Omega$.

Note that, in the sequel, we will use an analogous notation as we did with the radial basis functions, Remark 3.3.5 (a).

Towards band pass filters We have seen that the low pass filters enable a multiresolution analysis. Wavelets are defined as the difference of scaling functions such as low pass filters. Thus, we obtain a similar result here.

Definition 3.4.5. *For $S \in \mathbb{R}^+$, let the systems $\{w_s\}_{s\in\,]0,S]}$ and $\{\widetilde{w}_s\}_{s\in\,]0,S]}$ be a primal and dual radial basis wavelet, respectively. The spaces*

$$\begin{aligned} W_s &:= \left\{ \widetilde{w}_s * w_s * f \;\middle|\; f \in \mathrm{L}^2(\Omega) \right\} \\ &= \left\{ \sum_{n=0}^{\infty} \sum_{j=-n}^{n} \widetilde{w}_s^\wedge(n) w_s^\wedge(n) f^\wedge(n,j) Y_{n,j} \;\middle|\; f \in \mathrm{L}^2(\Omega) \right\} \end{aligned}$$

for $s \in \,]0,S]$ are called detail spaces.

Compare with Michel (2013, Theorem 7.29). With the detail spaces, we obtain the scale-step property of wavelets (see e. g. Michel, 2013, Theorem 7.30).

Theorem 3.4.6. ***(Scale-Step Property)*** *For $S \in \mathbb{R}^+$, let $\{k_s\}_{s\in\,]0,S]} \subset \mathrm{L}^2(\Omega\times\Omega)$ be a scaling function and the systems $\{w_s\}_{s\in\,]0,S]}$ and $\{\widetilde{w}_s\}_{s\in\,]0,S]}$ the corresponding primal and dual radial basis wavelet, respectively. Further, let $s \in \,]0,S]$ and $N \in \mathbb{N}$. For any $f \in \mathrm{L}^2(\Omega)$ it holds*

$$k_{s/2^N} * k_{s/2^N} * f = k_s * k_s * f + \sum_{n=0}^{N-1} \widetilde{w}_{s/2^n} * w_{s/2^n} * f$$

and

$$f = k_s * k_s * f + \sum_{n=0}^{\infty} \widetilde{w}_{s/2^n} * w_{s/2^n} * f$$

where the latter equality holds in the sense of $\mathrm{L}^2(\Omega)$.

Hence, in analogy to low pass filters, we define band pass filters.

Definition 3.4.7. *For* $S \in \mathbb{R}^+$, *let* $\{w_s\}_{s\in\,]0,S]}$ *and* $\{\widetilde{w}_s\}_{s\in\,]0,S]}$ *be a primal and dual radial basis wavelet, respectively. If, for all* $s \in]0, S]$ *and for all* $\varepsilon > 0$, *there exist* $n_1,\ n_2 \in \mathbb{N}_0$ *such that, for all* $n \leq n_1$, *it holds*

$$\left|w_s^\wedge(n)\right| < \varepsilon \qquad \text{as well as} \qquad \left|\widetilde{w}_s^\wedge(n)\right| < \varepsilon$$

and, for all $n \geq n_2$ *it holds*

$$\left|w_s^\wedge(n)\right| < \varepsilon \qquad \text{as well as} \qquad \left|\widetilde{w}_s^\wedge(n)\right| < \varepsilon,$$

then the system $\{\widetilde{w}_s * w_s\}_{s\in\,]0,S]}$ *is called a band pass filter. If the wavelets are P-wavelets, the band pass filters are called P-band pass filters. If the wavelets are M-wavelets, the band pass filters are called M-band pass filters.*

Note again that we call the systems as well as a single function a band pass filter. Band pass filters shall be used as possible dictionary elements. For band pass filters, we inherit a Legendre representation.

Theorem 3.4.8. *For all* $\eta \in \Omega$, *a fixed* $\zeta \in \Omega$ *and a primal and dual radial basis wavelet* $\{w_s\}_{s\in\,]0,S]}$ *and* $\{\widetilde{w}_s\}_{s\in]0,S]}$ *with* $S \in \mathbb{R}^+$, *respectively, the corresponding band pass filter has the expansion*

$$\left(\widetilde{w}_s * w_s\right)(\zeta, \eta) = \sum_{n=0}^{\infty} \frac{2n+1}{4\pi} \widetilde{w}_s^\wedge(n) w_s^\wedge(n) P_n(\zeta \cdot \eta)$$

for the n-th Legendre polynomial P_n.

Proof. Also for wavelets, the equalities from Theorem 3.3.7 hold. □

As an example for radial basis wavelets, we develop Example 3.3.17 further (see also e. g. Freeden et al., 1998, pp. 289-290).

Example 3.4.9. Again, we consider the generator of a scaling function $\kappa\colon \mathbb{R}_0^+ \to \mathbb{R}$ given by

$$\kappa(\tau) := \exp(-\tau).$$

The Legendre coefficients of the scaling function are given by

$$k_s^\wedge(n) = \exp\left(-sn\right) = \left(\exp\left(-s\right)\right)^n.$$

We have seen in Example 3.3.17 that this gives us the Abel–Poisson kernel or scaling function as well as a low pass filter. We want to construct the related P-band pass filter. The Legendre coefficients of the primal and dual radial basis wavelet are defined by

$$\begin{aligned} w_s^\wedge(n) \coloneqq \widetilde{w}_s^\wedge(n) &\coloneqq \sqrt{\left(k_{s/2}^\wedge(n)\right)^2 - \left(k_s^\wedge(n)\right)^2} \\ &= \sqrt{\left(\exp\left(-\frac{s}{2}n\right)\right)^2 - \left(\exp\left(-sn\right)\right)^2} \\ &= \sqrt{\exp\left(-s\right)^n - \exp\left(-2s\right)^n}. \end{aligned}$$

Similarly to the case of low pass filters, we need to consider the convolution of the primal and the dual radial basis wavelet to construct band pass filters. Only the convolutions yield the respective detail spaces and the scale-step property needed for a decomposition of the multiresolution analysis, see Theorem 3.4.6. For $\eta \in \Omega$ and a fixed $s\zeta \in \mathring{\mathbb{B}}$, we obtain in this case

$$\begin{aligned} W(x,\eta) \coloneqq \left(\widetilde{w} * w\right)(x,\eta) &= \sum_{n=0}^{\infty} \left(w_s^\wedge(n)\right)^2 \frac{2n+1}{4\pi} P_n(\zeta,\eta) \\ &= \sum_{n=0}^{\infty} \left(\exp\left(-s\right)^n - \exp\left(-2s\right)^n\right) \frac{2n+1}{4\pi} P_n(\zeta,\eta) \\ &= K\left(\exp\left(-s\right)\zeta,\eta\right) - K\left(\exp\left(-2s\right)\zeta,\eta\right). \end{aligned}$$

Thus, for $b \coloneqq \exp\left(-s\right)$, we have

$$W(x,\eta) \coloneqq W(b\zeta,\eta) \coloneqq K\left(b\zeta,\eta\right) - K\left(b^2\zeta,\eta\right).$$

As the Abel–Poisson kernel is defined for arbitrary $b \in [0,1[$, an Abel–Poisson P-band pass filter is equivalently defined and can be reformulated as follows:

$$W(x,\eta) \coloneqq K\left(x,\eta\right) - K\left(|x|x,\eta\right) \tag{3.15}$$

for $x = b\zeta \in \mathring{\mathbb{B}}$, $b \in [0,1[$ and $\zeta \in \Omega$. Note that the closed form of the Abel–Poisson kernel as seen in Example 3.3.4 makes also the respective P-band pass filters advantageous for practical purposes. Examples of Abel–Poisson P-band pass filters are given in Figure 3.3. In the plots, four Abel–Poisson P-band pass filters are shown. Blue colour stands for minimal values and red colour for maximal ones. All P-band pass filters are given a fixed centre ζ. However, they attain different scales. Presented are an Abel–Poisson P-band pass filters with scale 0.7, 0.8, 0.9 and 0.97 (from left to right). Each P-band pass filter is a localized function and depicts the usual wavelet form. Further, the scale inherits the control of the level of localization from the Abel–Poisson kernels. A comparison of the plot on the left-hand side with the one on the right-hand side shows that, for higher scales, the wavelets are stronger localized.

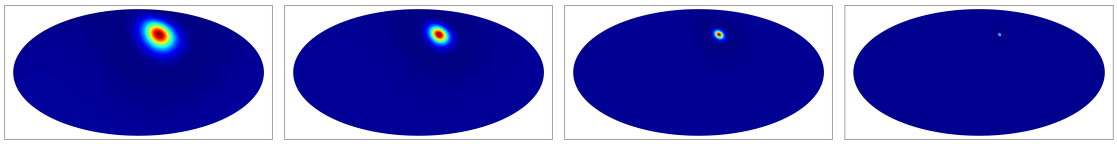


Figure 3.3.: Abel–Poisson P-band pass filters with a fixed centre ζ in all plots but different scales $b = 0.7,\ 0.8,\ 0.9,\ 0.97$ (left to right).

Some aspects of band pass filters and Sobolev spaces In Section 3.3, we derived that, under certain assumptions, a low pass filter is the reproducing kernel of a specific Sobolev space. As in our approach the band pass filters are tightly related to the low pass filters, the properties are obtained analogously in this case. The derivation with respect to low pass filters used the Legendre representation of a radial basis function. We considered the Legendre expansion of band pass filters in Theorem 3.4.8. Further, we defined the related wavelets by their Legendre coefficients. Thus, we obtain that the Sobolev space $\mathcal{H}((A_n);\Omega)$ related to a band pass filter $\{\widetilde{w}_s * w_s\}_{s\in\,]0,S]}$, $S \in \mathbb{R}^+$, is defined by

$$A_n = \left(\widetilde{w}_s^\wedge(n) w_s^\wedge(n)\right)^{-1/2}$$

if $\widetilde{w}_s^\wedge(n) w_s^\wedge(n) \neq 0$ for all $n \in \mathbb{N}_0$ and the sequence $(A_n)_{n\in\mathbb{N}_0}$ is summable. In the case of the Abel–Poisson P-band pass filter, we obtain

$$A_n = \left(b^n - b^{2n}\right)^{-1/2}.$$

Note that, again, this sequence is summable due to $b \in [0,1[$. All in all, we obtain similar results as before.

Lemma 3.4.10. *Let $s \in]0,S]$, $S \in \mathbb{R}^+$, be fixed and $Z \subseteq \Omega$ be dense and countable. If it holds*

$$\widetilde{w}_s^\wedge(n) w_s^\wedge(n) \neq 0$$

for all $n \in \mathbb{N}_0$ and the sequence $((\widetilde{w}_s^\wedge(n) w_s^\wedge(n))^{-1/2})_{n\in\mathbb{N}_0}$ is summable, then the system of band pass filters

$$\{\widetilde{w}_s(\zeta,\cdot) * w_s(\zeta,\cdot) \mid \zeta \in Z\}$$

is closed in the space $(\mathrm{L}^2(\Omega), \|\cdot\|_{\mathrm{L}^2(\Omega)})$ in the sense of the approximation theory.

Proof. Apply Theorems 3.2.13, 3.2.15 and 3.4.8. □

Lemma 3.4.11. *For a value $b \in [0,1[$ and a dense and countable set $Z \in \Omega$, the system of Abel–Poisson P-band pass filters*

$$\{W(b\zeta,\cdot) \mid \zeta \in Z\} = \left\{W(x,\cdot) \;\middle|\; \frac{x}{b} \in Z\right\}$$

is closed in the space $(\mathrm{L}^2(\Omega), \|\cdot\|_{\mathrm{L}^2(\Omega)})$ in the sense of the approximation theory.

Proof. Apply Lemma 3.4.10. □

Lemma 3.4.12. *Let $s \in]0, S]$, $S \in \mathbb{R}^+$, be fixed and $Z \subseteq \Omega$ be dense and countable. If it holds*

$$\widetilde{w}_s^\wedge(n) w_s^\wedge(n) \neq 0$$

for all $n \in \mathbb{N}_0$,

$$(n+0.5)^2 \leq \left(w_s^\wedge(n)\widetilde{w}_s^\wedge(n)\right)^{-1/2}$$

for all $n \in \mathbb{N}_0$ with $n \geq n_0 \in \mathbb{N}_0$ and the sequence $((\widetilde{w}_s^\wedge(n) w_s^\wedge(n))^{-1/2})_{n\in\mathbb{N}_0}$ is summable, then the system of band pass filters

$$\{\widetilde{w}_s(\zeta, \cdot) * w_s(\zeta, \cdot) \mid \zeta \in Z\},$$

is closed in the space $(\mathcal{H}_2(\Omega), \|\cdot\|_{\mathcal{H}_2(\Omega)})$ in the sense of the approximation theory.

Proof. Apply Theorems 3.2.13, 3.2.16 and 3.4.8. □

Lemma 3.4.13. *For a value $b \in [0, 1[$ and a dense and countable set $Z \in \Omega$, the system of Abel–Poisson P-band pass filters*

$$\{W(b\zeta, \cdot) \mid \zeta \in Z\} = \left\{W(x, \cdot) \;\middle|\; \frac{x}{b} \in Z\right\}$$

is closed in the space $(\mathcal{H}_2(\Omega), \|\cdot\|_{\mathcal{H}_2(\Omega)})$ in the sense of the approximation theory.

Proof. Apply Lemma 3.4.12. □

3.5. Inner products and upward continued values

For practical purposes, we consider some particular inner products and the upward continued values of the introduced trial functions at this point.
At first, we consider the $\mathrm{L}^2(\Omega)$-inner products where one argument is a spherical harmonic. We obtain the following results.
For two spherical harmonics, we already pointed out (see (2.12)) that it holds

$$\left\langle Y_{m,k}, Y_{n,j}\right\rangle_{\mathrm{L}^2(\Omega)} = \delta_{n,m}\delta_{j,k}$$

for $n,\ m,\ k,\ j \in \mathbb{N}_0$ with $-m \leq k \leq m$ and $-n \leq j \leq n$. For an Abel–Poisson kernel and a spherical harmonic, we obtain

$$\left\langle K(x, \cdot), Y_{n,j}\right\rangle_{\mathrm{L}^2(\Omega)} = K(x, \cdot)^\wedge(n) Y_{n,j}\left(\frac{x}{|x|}\right) = |x|^n Y_{n,j}\left(\frac{x}{|x|}\right)$$

for a $x \in \mathring{\mathbb{B}}$ and an $n,\ j \in \mathbb{N}_0$ with $-n \leq j \leq n$ due to (3.14). Using this result, we obtain for an Abel–Poisson P-band pass filter that it holds

$$\begin{aligned}\left\langle W(x,\cdot), Y_{n,j}\right\rangle_{\mathrm{L}^2(\Omega)} &= \left\langle K\left(x,\cdot\right), Y_{n,j}\right\rangle_{\mathrm{L}^2(\Omega)} - \left\langle K\left(|x|x,\cdot\right), Y_{n,j}\right\rangle_{\mathrm{L}^2(\Omega)} \\ &= \left(|x|^n - |x|^{2n}\right) Y_{n,j}\left(\frac{x}{|x|}\right)\end{aligned}$$

for a $x \in \mathring{\mathbb{B}}$ and an $n,\ j \in \mathbb{N}_0$ with $-n \leq j \leq n$. At last, for a Slepian function $g^{(k,N)}$ with band-limit N and a spherical harmonic, we have

$$\left\langle g^{(k,N)}(R,\cdot), Y_{n,j}\right\rangle_{\mathrm{L}^2(\Omega)} = \begin{cases} g_{n,j}^{(k,N)}(R), & n \leq N, \\ 0, & else \end{cases}$$

by construction. With these inner product, we consider the upward continuation of the specified trial functions, see (2.26). Again, we already showed in Lemma 2.4.8 that it holds

$$\mathcal{T} Y_{m,k} = \sigma^{-m-1} Y_{m,k}$$

for $m,\ k \in \mathbb{N}_0$ with $-m \leq k \leq m$. For the upward continuation of an Abel–Poisson kernel, we obtain

$$\begin{aligned}\mathcal{T}K(x,\cdot) &= \sum_{n=0}^{\infty}\sum_{j=-n}^{n} \left\langle Y_{n,j}, K(x,\cdot)\right\rangle_{\mathrm{L}^2(\Omega)} \sigma^{-n-1} Y_{n,j} = \sum_{n=0}^{\infty}\sum_{j=-n}^{n} \frac{|x|^n}{\sigma^{n+1}} Y_{n,j}\left(\frac{x}{|x|}\right) Y_{n,j} \\ &= \sum_{n=0}^{\infty} \frac{|x|^n}{\sigma^{n+1}} \frac{2n+1}{4\pi} P_n\left(\frac{x}{|x|}\cdot\right) = \frac{1}{\sigma} K\left(\frac{x}{\sigma},\cdot\right) \end{aligned} \tag{3.16}$$

for a $x \in \mathring{\mathbb{B}}$. Thus, for an Abel–Poisson band pass filter, it yields

$$\mathcal{T}W(x,\cdot) = \mathcal{T}K\left(x,\cdot\right) - \mathcal{T}K\left(|x|x,\cdot\right) = \frac{1}{\sigma} K\left(\frac{x}{\sigma},\cdot\right) - \frac{1}{\sigma} K\left(\frac{|x|x}{\sigma},\cdot\right) \tag{3.17}$$

for a $x \in \mathring{\mathbb{B}}$ as $\mathcal{T}$ is linear. Last but not least, the upward continued value of a Slepian function of band-limit N is derived as

$$\begin{aligned}\mathcal{T} g^{(k,N)}(R,\cdot) &= \sum_{n=0}^{\infty}\sum_{j=-n}^{n} \left\langle g^{(k,N)}(R,\cdot), Y_{n,j}\right\rangle_{\mathrm{L}^2(\Omega)} \sigma^{-n-1} Y_{n,j} \\ &= \sum_{n=0}^{N}\sum_{j=-n}^{n} g_{n,j}^{(k,N)}(R)\sigma^{-n-1} Y_{n,j}. \end{aligned} \tag{3.18}$$

At last, for a penalty term in our computations, we need the values of the $\mathcal{H}_2(\Omega)$-inner products of all combinations of the introduced trial functions. We derive

them at this point. Let $n,\ m,\ j,\ k,\ o,\ p,\ L \in \mathbb{N}_0$ with $-n \leq j \leq n,\ -m \leq k \leq m$ and $o,\ p \leq (L+1)^2$. Further, let $x,\ x' \in \mathring{\mathbb{B}}$ and $R,\ R' \subseteq \Omega$.

$$\begin{aligned}
&\left\langle Y_{n,j}, Y_{m,k} \right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = A_n^2 \delta_{n,m} \delta_{j,k}
\end{aligned} \tag{3.19}$$

$$\begin{aligned}
&\left\langle K(x,\cdot), Y_{m,k} \right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \sum_{n=0}^{\infty} \sum_{j=-n}^{n} A_n^2 \left\langle K(x,\cdot), Y_{n,j} \right\rangle_{\mathrm{L}^2(\Omega)} \left\langle Y_{m,k}, Y_{n,j} \right\rangle_{\mathrm{L}^2(\Omega)} \\
&\quad = \sum_{n=0}^{\infty} \sum_{j=-n}^{n} A_n^2 |x|^n Y_{n,j}\left(\frac{x}{|x|}\right) \delta_{n,m} \delta_{j,k} \\
&\quad = A_m^2 |x|^m Y_{m,k}\left(\frac{x}{|x|}\right)
\end{aligned} \tag{3.20}$$

$$\begin{aligned}
&\left\langle W(x,\cdot), Y_{m,k} \right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \left\langle K(x,\cdot), Y_{m,k} \right\rangle_{\mathcal{H}_2(\Omega)} - \left\langle K(|x|x,\cdot), Y_{m,k} \right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = A_m^2 \left(|x|^m - |x|^{2m} \right) Y_{m,k}\left(\frac{x}{|x|}\right)
\end{aligned} \tag{3.21}$$

$$\begin{aligned}
&\left\langle g^{(o,L)}(R,\cdot), Y_{m,k} \right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \sum_{l=0}^{L} \sum_{i=-l}^{l} g_{l,i}^{(o,L)}(R) \left\langle Y_{l,i}, Y_{m,k} \right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \begin{cases} A_m^2 g_{m,k}^{(o,L)}(R), & m \leq L \\ 0, & else \end{cases}
\end{aligned} \tag{3.22}$$

$$\begin{aligned}
&\left\langle K(x,\cdot), K\left(x',\cdot\right) \right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \sum_{n=0}^{\infty} \sum_{j=-n}^{n} A_n^2 \left\langle K(x,\cdot), Y_{n,j} \right\rangle_{\mathrm{L}^2(\Omega)} \left\langle K\left(x',\cdot\right), Y_{n,j} \right\rangle_{\mathrm{L}^2(\Omega)} \\
&\quad = \sum_{n=0}^{\infty} A_n^2 \left(|x| \left|x'\right| \right)^n \sum_{j=-n}^{n} Y_{n,j}\left(\frac{x}{|x|}\right) Y_{n,j}\left(\frac{x'}{|x'|}\right) \\
&\quad = \sum_{n=0}^{\infty} A_n^2 \left(|x| \left|x'\right| \right)^n \frac{2n+1}{4\pi} P_n\left(\frac{x}{|x|} \cdot \frac{x'}{|x'|}\right)
\end{aligned} \tag{3.23}$$

$$
\begin{aligned}
&\left\langle W(x,\cdot), K\left(x',\cdot\right)\right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \left\langle K\left(x,\cdot\right), K\left(x',\cdot\right)\right\rangle_{\mathcal{H}_2(\Omega)} - \left\langle K\left(|x|x,\cdot\right), K\left(x',\cdot\right)\right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \sum_{n=0}^{\infty} A_n^2 \left(|x|\left|x'\right|\right)^n \frac{2n+1}{4\pi} P_n\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right) \\
&\qquad - \sum_{n=0}^{\infty} A_n^2 \left(|x|^2\left|x'\right|\right)^n \frac{2n+1}{4\pi} P_n\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right) \\
&\quad = \sum_{n=0}^{\infty} A_n^2 \left(|x|^n - |x|^{2n}\right)\left|x'\right|^n \frac{2n+1}{4\pi} P_n\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right)
\end{aligned} \tag{3.24}
$$

$$
\begin{aligned}
&\left\langle g^{(o,L)}(R,\cdot), K\left(x',\cdot\right)\right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \sum_{l=0}^{L}\sum_{i=-l}^{l} g_{l,i}^{(o,L)}(R)\left\langle Y_{l,i}, K\left(x',\cdot\right)\right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \sum_{l=0}^{L}\sum_{i=-l}^{l} A_l^2 g_{l,i}^{(o,L)}(R)\left|x'\right|^l Y_{l,i}\left(\frac{x'}{|x'|}\right)
\end{aligned} \tag{3.25}
$$

$$
\begin{aligned}
&\left\langle W(x,\cdot), W\left(x',\cdot\right)\right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \sum_{n=0}^{\infty}\sum_{j=-n}^{n} A_n^2 \left\langle W(x,\cdot), Y_{n,j}\right\rangle_{\mathrm{L}^2(\Omega)} \left\langle W\left(x',\cdot\right), Y_{n,j}\right\rangle_{\mathrm{L}^2(\Omega)} \\
&\quad = \sum_{n=0}^{\infty} A_n^2 \left(|x|^n - |x|^{2n}\right)\left(\left|x'\right|^n - \left|x'\right|^{2n}\right)\frac{2n+1}{4\pi} P_n\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right)
\end{aligned} \tag{3.26}
$$

$$
\begin{aligned}
&\left\langle g^{(o,L)}(R,\cdot), W\left(x',\cdot\right)\right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \sum_{l=0}^{L}\sum_{i=-l}^{l} g_{l,i}^{(o,L)}(R)\left\langle Y_{l,i}, W\left(x',\cdot\right)\right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \sum_{l=0}^{L}\sum_{i=-l}^{l} A_l^2 g_{l,i}^{(o,L)}(R)\left(\left|x'\right|^l - \left|x'\right|^{2l}\right) Y_{l,i}\left(\frac{x'}{|x'|}\right)
\end{aligned} \tag{3.27}
$$

$$
\begin{aligned}
&\left\langle g^{(o,L)}(R,\cdot), g^{(p,L)}\left(R',\cdot\right)\right\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \sum_{n=0}^{\infty}\sum_{j=-n}^{n} A_n^2 \left\langle g^{(o,L)}(R,\cdot), Y_{n,j}\right\rangle_{\mathrm{L}^2(\Omega)} \left\langle g^{(p,L)}\left(R',\cdot\right), Y_{n,j}\right\rangle_{\mathrm{L}^2(\Omega)} \\
&\quad = \sum_{n=0}^{L}\sum_{j=-n}^{n} A_n^2 g_{n,j}^{(o,L)}(R) g^{(p,L)}\left(R'\right)_{n,j}
\end{aligned} \tag{3.28}
$$

Note that for those values which are represented by a series, an approximate value can be obtained by truncating the series. The truncated series can be computed with the use of the Clenshaw algorithm, see Appendix A.

4. An algorithmic approach: matching pursuits

In Chapter 2, we have seen that the downward continuation of satellite data is an example of an ill-posed inverse problem. Such problems cannot be solved easily in general. With the use of the generalized inverse, we obtain the unique minimum-norm solution. However, stability of this solution can usually only be achieved with regularization strategies. Such strategies give us an approximate well-posed problem to solve instead. As an example, we introduced the Tikhonov-Philipps regularization in Example 2.5.17. In this particular strategy, we have to solve either the regularized normal equation or a minimization problem to obtain a solution. In the first case, traditionally, the problem is reformulated to a system of linear equations. In the latter case, one particular approach is given by certain **i**nverse **p**roblem **m**atching **p**ursuit (IPMP) algorithms. In its most general sense, such methods approximate a solution of a given inverse problem with the use of a dictionary. In contrast to the solution of a system of linear equations, this approximation is given as a function and not as discretized values. In this chapter, we introduce the reader to matching pursuits in general and the IPMP algorithms in particular.

4.1. An introduction to matching pursuits

We consider an approximation problem: we are given measurements of a signal at discrete points of a certain domain. Our aim is to determine a representation of this signal as a function. In general, such an approximation task can be solved by well-known methods which expand the solution in an orthonormal basis, splines or wavelets (see e. g. Freeden et al., 1998; Hanke-Bourgeois, 2009; Magnus et al., 1966; Michel, 2013; Schwarz and Köckler, 2011; Szegö, 1975). A matching pursuit, however, follows a different route. It assumes that the signal can be well represented as a linear combination of so-called dictionary elements.
This section is based on Mallat and Zhang (1993); Pati et al. (1993); Vincent and Bengio (2002) for the basic matching pursuits and Fischer and Michel (2013a); Gutting et al. (2017); Kontak (2018); Kontak and Michel (2018); Michel (2015a, 2020); Michel and Orzlowski (2017); Michel and Telschow (2016); Telschow (2014) for the functional matching pursuits. First of all, we express the idea of using dictionaries.

Why use a dictionary? "We can express a wide range of ideas and at the same time easily communicate subtle difference between close concepts, because natural languages have large vocabularies, that include words with close meanings. [...] Such decompositions are similar to a text written with a small vocabulary. Although this vocabulary might be sufficient to express all ideas, it requires to use circumvolutions that replace unavailable words by full sentences." – Mallat and Zhang (1993)
This description from Mallat and Zhang points out one of the main advantages of a matching pursuit. It might be a useful tool for a better understanding to represent a signal in a given (possibly orthonormal) basis. However, this concept is not necessarily a very natural one. With a broader vocabulary, one can be more accurate with less words and more precise due to using more suitable words. A matching pursuit aims to incorporate this idea into mathematics.

Similar works with different names The term matching pursuit was originally coined in works related to signal processing. Similar approaches have been developed in statistics and are known as projection pursuit (regression) (see e. g. Friedman and Stuetzle, 1981; Huber, 1985; Jones, 1987). Further, in approximation theory, similar algorithms were called greedy algorithms (see e. g. Temlyakov, 2011). Note that a matching pursuit indeed has a greedy component as we will see. However, it is no greedy algorithm in the sense of computer science, i. e. it may not produce the global solution of an optimization problem. We will discuss the difference of such problems to a matching pursuit in a bit more detail in Chapter 6.

4.1.1. The classical matching pursuit

We start with some notations needed for discussing any matching pursuit. Further, we also formulate the classical matching pursuits with a (discretization) operator.

Remark 4.1.1. At first, we consider the following approximation problem: for a set of grid points

$$\eta^{(i)} \in \mathbb{R}^3,\ i = 1, ..., \ell \qquad \text{with} \qquad \ell \in \mathbb{N},$$

let

$$y = (y_i)_{i=1,...,\ell} \in \mathbb{R}^\ell \qquad \text{with} \qquad y_i = f\left(\eta^{(i)}\right) \in \mathbb{R} \tag{4.1}$$

denote measurements of a given signal . We are interested in the function $f \in H$ for a Hilbert space $(H, \langle \cdot, \cdot \rangle_H)$.

In order to use a consistent notation, we immediately relate the approximation problem to an inverse problem.

Remark 4.1.2. For each y_i, $i = 1,..,\ell$ from Remark 4.1.1, we can write $y_i = (If)(\eta^{(i)})$ with the identity operator I. We exchange the operator I by a system of (discretization) functionals $I_{\urcorner}^i$ such that $I_{\urcorner}^i f = (If)(\eta^{(i)}) = y_i$ holds for $i = 1,...,\ell$. The Hebrew letter $\urcorner$ is used to emphasize that the system $(I_{\urcorner}^i)_{i=1,...,\ell}$ is a discretization of I. Then we can write $I_{\urcorner} := (I_{\urcorner}^i)_{i=1,...,\ell}$ and have the approximation problem in the familiar formulation

$$I_{\urcorner} f = y.$$

Note that, by using the operator $I_{\urcorner}$, we obtain $I_{\urcorner} f \in \mathbb{R}^\ell$.

The usual approach for this approximation problem is to represent the unknown signal f by its truncated Fourier series

$$f \approx \sum_{n=0}^{N} \langle f, g_n \rangle_H \, g_n$$

for some basis g_n of $(H, \langle \cdot, \cdot \rangle_H)$. However, we have seen in the last chapters that diverse bases have their particular (dis-)advantages and, thus, it may be difficult to determine a suitable one. Further, using the truncated Fourier series gives rise to numerically difficult computations, for instance, if we have to solve a large system of linear equations. In a matching pursuit, we exchange the basis functions with dictionary elements.

Definition 4.1.3. *Let $(H, \langle \cdot, \cdot \rangle_H)$ be a Hilbert space of functions. A dictionary is a family*

$$\mathcal{D} \subseteq H \backslash \ker I_{\urcorner} \qquad \text{with} \qquad \ker I_{\urcorner} = \{d \in \mathcal{D} \mid I_{\urcorner} d = 0\},$$

i. e. a dictionary is an arbitrarily large set of functions in which each one does not map to zero under $I_{\urcorner}$. Each $d \in \mathcal{D}$ is called a dictionary element. Further, we set

$$\begin{aligned} V &:= V_{\mathcal{D}} := \overline{\operatorname{span} \mathcal{D}}^{\|\cdot\|_H}, \\ \mathcal{V} &:= \mathcal{V}_{\mathcal{D}} := \overline{\operatorname{span} \{ I_{\urcorner} d \mid d \in \mathcal{D} \}}^{\|\cdot\|_{\mathbb{R}^\ell}} \end{aligned} \tag{4.2}$$

and

$$\begin{aligned} V_{\mathcal{D}_N} &:= \operatorname{span} \mathcal{D}_N, & \mathcal{D}_N \subseteq \mathcal{D},\ |\mathcal{D}_N| = N, \\ \mathcal{V}_{\mathcal{D}_N} &:= \operatorname{span}\{I_{\urcorner} \mathcal{D}_N\}, & I_{\urcorner} \mathcal{D}_N := \{ I_{\urcorner} d \mid d \in \mathcal{D}_N \} \end{aligned} \tag{4.3}$$

for $N \in \mathbb{N}$. We say a dictionary $\mathcal{D}$ is complete if and only if $V = H$. We say a dictionary $\mathcal{D}$ is overcomplete if and only if there exists a proper subset $\mathcal{D}^\star \subset \mathcal{D}$ which is complete.

Note that the smallest possible complete dictionary consists of a basis of H. The largest possible complete dictionary is naturally H itself (see Mallat and Zhang, 1993).

Remark 4.1.4. For a closed linear subspace $U \subseteq X$ ($X = H$ for a Hilbert space of functions $(H, \langle \cdot, \cdot \rangle_H)$ or $X = \mathbb{R}^\ell$), we denote the orthogonal projection of $x \in X$ onto U by $\mathcal{P}_U x$ and the orthogonal complement of U with respect to the inner product $\langle \cdot, \cdot \rangle_X$ as $U^\perp$. Then, for $U \subseteq X$ and any $x \in X$, we have

$$x = \mathcal{P}_U x + \mathcal{P}_{U^\perp} x. \tag{4.4}$$

Further, let the elements of $\mathcal{V}_N \subseteq \mathbb{R}^\ell$, $N \in \mathbb{N}$ be pairwise orthogonal and enumerated. Then we have

$$\mathcal{P}_{\mathcal{V}_N} I\neg g = \sum_{n=1}^{N} \frac{\langle I\neg g, I\neg d_n \rangle_{\mathbb{R}^\ell}}{\| I\neg d_n \|^2_{\mathbb{R}^\ell}} \, I\neg d_n \tag{4.5}$$

for any $g \in H$ and dictionary elements $d_n \in \mathcal{D}_N$.

Definition 4.1.5. *For a Hilbert space $(H, \langle \cdot, \cdot \rangle_H)$ of functions, the aim of the classical matching pursuit is to solve an approximation problem from given data $y \in \mathbb{R}^\ell$. For this, it constructs an approximation f_N, $N \in \mathbb{N}$, of the solution $f \in H$. From now on, $N \in \mathbb{N}$ will always be the current iteration. The approximation f_N is given as a decomposition in a best basis $\mathcal{D}_N = \{d_n \mid d_n \in \mathcal{D},\ n = 1, ..., N\}$, i. e. in the form*

$$f_N \coloneqq \sum_{n=1}^{N} \alpha_n d_n \tag{4.6}$$

for dictionary elements $d_n \in \mathcal{D} \subseteq H \backslash \ker I\neg$ and real coefficients α_n. The residual is denoted by

$$R^N \coloneqq y - I\neg f_N \in \mathbb{R}^\ell \tag{4.7}$$

for a vector of measurements y as in (4.1)*. For an approximation f_N, it should hold*

$$\lim_{N \to \infty} \left\| R^N \right\|_{\mathbb{R}^\ell} = 0.$$

The system

$$\{(\alpha_n, d_n)\}_{n=1,...,N} \tag{4.8}$$

is called a structure book.

We have to determine the coefficients α_n and the dictionary elements d_n of the decomposition (4.6) for $n = 1, ..., N$, i. e. the best basis (or its structure book). In the classical matching pursuit (see e. g. Mallat and Zhang, 1993; Pati et al., 1993; Vincent and Bengio, 2002), this is done by iteratively minimizing the norm of the residual R^N: we first assume that f_0 is some initial approximation. In most

cases, we do not have a sophisticated guess for f_0 and, thus, set $f_0 \equiv 0$. Then we iteratively add dictionary elements. We set

$$f_{N+1} \coloneqq f_N + \alpha_{N+1} d_{N+1}$$

for iteration N such that

$$\left\| R^{N+1} \right\|_{\mathbb{R}^\ell} \leq \left\| R^N \right\|_{\mathbb{R}^\ell}. \tag{4.9}$$

Thus, the questions is how to determine (α_{N+1}, d_{N+1}) in each iteration. With (4.9), we aim to minimize

$$\begin{aligned} \left\| R^{N+1} \right\|_{\mathbb{R}^\ell}^2 &= \| y - I\neg f_{N+1} \|_{\mathbb{R}^\ell}^2 \\ &= \| y - I\neg f_N - \alpha_{N+1} I\neg d_{N+1} \|_{\mathbb{R}^\ell}^2 \\ &= \left\| R^N - \alpha_{N+1} I\neg d_{N+1} \right\|_{\mathbb{R}^\ell}^2 \end{aligned} \tag{4.10}$$

and choose

$$(\alpha_{N+1}, d_{N+1}) = \operatorname*{arg\,min}_{(\alpha,d)\in\mathbb{R}\times\mathcal{D}} \left\| R^N - \alpha I\neg d \right\|_{\mathbb{R}^\ell}^2. \tag{4.11}$$

Note that, for a finite dictionary, this minimum is surely attained. If the dictionary is infinite, we actually cannot be sure that the minimum exists and, thus, should only consider the infimum. However, in order to stay in accordance to the literature, here, we take over the notation with the minimum.
Hence, the finite set of in this way chosen dictionary elements $\mathcal{D}_N$ is called a best basis as an analogy to a representation in a traditional basis. However, it is a best basis because each dictionary element $d_n \in \mathcal{D}_N$ is chosen as the minimizer of the norm of R^n.
For practical purposes, we consider the quadratic term of (4.11) in more detail and can determine a coefficient $\alpha \in \mathbb{R}$ in dependence of a dictionary element $d \in \mathcal{D}$ as follows:

$$\begin{aligned} 0 &= \frac{\partial}{\partial\alpha} \left\| R^N - \alpha I\neg d \right\|_{\mathbb{R}^\ell}^2 = \frac{\partial}{\partial\alpha} \left(\left\| R^N \right\|_{\mathbb{R}^\ell}^2 - 2\alpha \left\langle R^N, I\neg d \right\rangle_{\mathbb{R}^\ell} + \alpha^2 \| I\neg d \|_{\mathbb{R}^\ell}^2 \right) \\ &= -2 \left\langle R^N, I\neg d \right\rangle_{\mathbb{R}^\ell} + 2\alpha \| I\neg d \|_{\mathbb{R}^\ell}^2. \end{aligned}$$

Thus, we eliminate the factor 2 and obtain

$$\alpha = \frac{\left\langle R^N, I\neg d \right\rangle_{\mathbb{R}^\ell}}{\| I\neg d \|_{\mathbb{R}^\ell}^2}. \tag{4.12}$$

Hence, the corresponding coefficient α_n is fixed for any choice of basis element $d_n \in \mathcal{D}_N$ and it suffices to choose d_{N+1} such that $\|R^{N+1}\|$ is minimized. For this, we insert (4.12) into $\|R^N - \alpha I\neg d\|^2_{\mathbb{R}^\ell}$ (confer (4.10) and (4.11)) and obtain

$$\left\|R^N - \alpha I\neg d\right\|^2_{\mathbb{R}^\ell} = \left\|R^N\right\|^2_{\mathbb{R}^\ell} - \left(\frac{\left\langle R^N, I\neg d\right\rangle_{\mathbb{R}^\ell}}{\|I\neg d\|_{\mathbb{R}^\ell}}\right)^2.$$

Hence, in each iteration N, the norm of the residual R^{N+1} is minimized if we choose the next best basis element $d_{N+1} \in \mathcal{D}$ such that

$$d_{N+1} := \arg\max_{d\in\mathcal{D}} \left(\frac{\left\langle R^N, I\neg d\right\rangle_{\mathbb{R}^\ell}}{\|I\neg d\|_{\mathbb{R}^\ell}}\right)^2. \tag{4.13}$$

Note that there might exist more than one dictionary element that maximizes this quotient. Moreover, similarly as before, for an infinite dictionary, we actually should seek a supremum to be precise. Again, in order to be in accordance with the literature, we maintain the formulation with the maximum. Further, note that this maximization also means that the normalized vector of values $I\neg d_{N+1}$ of the dictionary element $d_{N+1} \in \mathcal{D}$ with respect to grid points $\eta^{(i)}$, $i = 1, ..., \ell$, is chosen which is most collinear in the Euclidean sense to the current residual R^N (see e. g. Vincent and Bengio, 2002). We obtain a pseudo-code for this classical matching pursuit as given in Algorithm 1. Note that for practical purposes, the values $\langle I\neg d_i, I\neg d_j\rangle_{\mathbb{R}^\ell}$, d_i, $d_j \in \mathcal{D}$, can be preprocessed and used in the update rule

$$\left\langle R^{N+1},\ I\neg d\right\rangle_{\mathbb{R}^\ell} = \left\langle R^N,\ I\neg d\right\rangle_{\mathbb{R}^\ell} - \alpha_{N+1}\left\langle I\neg d_{N+1},\ I\neg d\right\rangle_{\mathbb{R}^\ell}$$

of the N-th iteration in order to improve the efficiency. Note that, in practice, the algorithm needs at least one termination criterion. We will discuss some possibilities in Section 4.5 after we introduced the IPMP algorithms. Further, note that the classical matching pursuit does not need to solve a large system of linear equations for the approximation task.
However, a classical matching pursuit faces two problems (see e. g. Pati et al., 1993; Vincent and Bengio, 2002): on the one hand, the choice of the structure book $\{(\alpha_n, d_n)\}_{n=1,...,N}$ does not represent an optimal N-term approximation in general because the coefficients α_n, $n = 1, ..., N$, of previous iterations are fixed in the current step as well as in future steps. On the other hand, if the coefficients α_n, $n = 1, ..., N$, are recomputed in each iteration N such that they are optimal with respect to the chosen dictionary elements d_n, $n = 1, ..., N+1$, and the coefficient α_{N+1}, then at least the latest chosen dictionary element d_{N+1} may not be the optimal choice any more. To solve these problems, the orthogonal matching pursuit was developed.

Data: $y \in \mathbb{R}^\ell$
Result: approximation f_N
initialization: $\mathcal{D}$, f_0, $R^0 := y - I\neg f_0$;
$N = 0$;
while *(stopping criteria not fulfilled)* **do**

$$d_{N+1} := \arg\max_{d\in\mathcal{D}} \left(\frac{\left\langle R^N,\, I\neg d \right\rangle_{\mathbb{R}^\ell}}{\| I\neg d\|_{\mathbb{R}^\ell}} \right)^2;$$

$$\alpha_{N+1} := \frac{\left\langle R^N,\, I\neg d_{N+1} \right\rangle_{\mathbb{R}^\ell}}{\|I\neg d_{N+1}\|^2_{\mathbb{R}^\ell}};$$

$$R^{N+1} := R^N - \alpha_{N+1} I\neg d_{N+1};$$

N be increased by 1;

end
return $f_N = \sum_{n=1}^N \alpha_n d_n$;

Algorithm 1: Pseudo-code for the matching pursuit.

4.1.2. The orthogonal matching pursuit

For the first problem, the classical matching pursuit was extended such that it simulates a choice of best coefficients $\alpha_1, ..., \alpha_{N+1}$ after the dictionary element d_{N+1} is chosen as the maximizer of (4.13) in the current iteration N. As the dictionary element is chosen first, the technique was called back-fitting (see e. g. Mallat and Zhang, 1993; Vincent and Bengio, 2002).
The second problem is dealt with the so-called pre-fitting technique (see e. g. Vincent and Bengio, 2002). As it mimics the simultaneous choice of an optimal dictionary element d_{N+1} and optimal coefficients $\alpha_1, ..., \alpha_{N+1}$, we will concentrate on this method in more detail next. Note that the back-fitting and the pre-fitting mechanisms both rely on an orthogonal projection. Thus, a matching pursuit is called orthogonal matching pursuit if one of these techniques is included. In the sequel, we consider matching pursuits with pre-fitting (which includes back-fitting naturally) as the orthogonal matching pursuit.
The idea of the orthogonal matching pursuit is as follows. With the pre-fitting technique, we want to choose a dictionary element d_{N+1} simultaneously with an optimal set of coefficients $\alpha_1, ..., \alpha_{N+1}$. Hence, we consider

$$\left(\alpha^{(N+1)}, d_{N+1}\right) = \underset{(\alpha,d)\in\mathbb{R}^{N+1}\times\mathcal{D}}{\arg\min} \left\| y - \sum_{n=1}^N \alpha_n I\neg d_n + \alpha_{N+1} I\neg d \right\|^2_{\mathbb{R}^\ell}$$

for $\alpha = (\alpha_1, ..., \alpha_{N+1}) \in \mathbb{R}^{N+1}$. Similar as in the non-orthogonal case, it would be more precise to speak of an infimum instead of the minimum, but we again

maintain the formulation with the minimum as was usually done in the literature. In each iteration N, we re-compute the coefficients $\alpha_1, ..., \alpha_N$. Hence, we introduce some additional superscripts to emphasize that the value after a certain iteration is meant. For instance, $\alpha_1^{(N)}$ stands for the value of the coefficient α_1 after the update in the N-th iteration. In general, the value of the superscript must always be equal or higher than the value of the subscript. If they are equal, then the coefficient is not updated yet, i. e.

$$\alpha_{N+1}^{(N+1)} = \alpha_{N+1}.$$

With respect to the given data $y \in \mathbb{R}^\ell$ and, in iteration N, the set $\mathcal{V}_N := \mathcal{V}_{\mathcal{D}_N}$ with $\mathcal{D}_N$ consisting of the N previously chosen dictionary elements, the best approximation $f_{N+1}^{(N+1)}$we can get fulfils

$$I\neg f_{N+1}^{(N+1)} := \sum_{n=1}^{N+1} \alpha_n^{(N+1)} I\neg d_n = \mathcal{P}_{\mathcal{V}_{N+1}} y.$$

We consider this in more detail. With the use of (4.4), we have for the N-th iteration (compare with Telschow, 2014)

$$\begin{aligned}
\mathcal{P}_{\mathcal{V}_{N+1}} y &= \mathcal{P}_{\mathcal{V}_N} y + \alpha_{N+1}^{(N+1)} \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d_{N+1} \\
&= \sum_{n=1}^{N} \alpha_n^{(N)} I\neg d_n + \alpha_{N+1}^{(N+1)} \left(I\neg d_{N+1} - \mathcal{P}_{\mathcal{V}_N} I\neg d_{N+1} \right) \\
&= \sum_{n=1}^{N} \alpha_n^{(N)} I\neg d_n - \alpha_{N+1}^{(N+1)} \sum_{n=1}^{N} \beta_n^{(N)} \left(d_{N+1} \right) I\neg d_n + \alpha_{N+1}^{(N+1)} I\neg d_{N+1} \\
&= \sum_{n=1}^{N} \left(\alpha_n^{(N)} - \alpha_{N+1}^{(N+1)} \beta_n^{(N)} \left(d_{N+1} \right) \right) I\neg d_n + \alpha_{N+1}^{(N+1)} I\neg d_{N+1}
\end{aligned} \tag{4.14}$$

with projection coefficients $\beta_n^{(N)}(d_{N+1})$, $n = 1, ..., N$, of the N-th iteration obtained from the Gram-Schmidt orthonormalization (compare with (4.5)). Note that these projection coefficients are generally not unique. A more detailed look on them will be provided with respect to the IPMP algorithms, confer Theorem 4.4.2 and Theorem 4.4.3. With the update rule for the coefficients

$$\alpha_n^{(N+1)} := \alpha_n^{(N)} - \alpha_{N+1}^{(N+1)} \beta_n^{(N)} \left(d_{N+1} \right), \tag{4.15}$$

we can write

$$\mathcal{P}_{\mathcal{V}_{N+1}} y = \sum_{n=1}^{N} \alpha_n^{(N+1)} I\neg d_n + \alpha_{N+1}^{(N+1)} I\neg d_{N+1} = I\neg f_{N+1}^{(N+1)}.$$

Hence, we define the residual as

$$\begin{aligned} R^{N+1} &:= y - I\neg f_{N+1}^{(N+1)} = y - I\neg f_N^{(N)} - \alpha_{N+1}^{(N+1)} \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d_{N+1} \\ &= R^N - \alpha_{N+1}^{(N+1)} \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d_{N+1} \end{aligned}$$

We summarize these considerations.

Definition 4.1.6. *For a Hilbert space $(H, \langle\cdot,\cdot\rangle_H)$ of functions, the orthogonal matching pursuit constructs an approximation*

$$I\neg f_N^{(N)} := \mathcal{P}_{\mathcal{V}_N} y = \sum_{n=1}^{N} \alpha_n^{(N)} I\neg d_n \tag{4.16}$$

for coefficients $\alpha_n^{(N)} \in \mathbb{R}, n = 1, ..., N$, and dictionary elements $d_n \in \mathcal{D} \subseteq H \backslash \ker I\neg$. Therefore, the residual is denoted by

$$R^{N+1} := y - I\neg f_{N+1}^{(N+1)} = R^N - \alpha_{N+1}^{(N+1)} \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d_{N+1}.$$

Then, in analogy to the classical matching pursuit, the aim of the N-th iteration of the orthogonal matching pursuit is to determine

$$\left(\alpha_{N+1}^{(N+1)}, d_{N+1}\right) = \underset{(\alpha,d)\in\mathbb{R}\times\mathcal{D}}{\arg\min} \left\| R^N - \alpha \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\|_{\mathbb{R}^\ell}^2, \tag{4.17}$$

where we again maintain the formulation with the minimum as was done in the literature instead of the (more precise) infimum. Similar as before, we consider the derivation with respect to α first:

$$\begin{aligned} 0 &= \frac{\partial}{\partial\alpha} \left(\left\| R^N - \alpha \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\|_{\mathbb{R}^\ell}^2 \right) \\ &= \frac{\partial}{\partial\alpha} \left(\left\| R^N \right\|_{\mathbb{R}^\ell}^2 - 2\alpha \left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\rangle_{\mathbb{R}^\ell}^2 + \alpha^2 \left\| \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\|_{\mathbb{R}^\ell}^2 \right) \\ &= -2 \left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\rangle_{\mathbb{R}^\ell}^2 + 2\alpha \left\| \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\|_{\mathbb{R}^\ell}^2. \end{aligned}$$

Thus, we have with

$$\alpha = \frac{\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\|_{\mathbb{R}^\ell}^2} \tag{4.18}$$

a dependence on the dictionary element $d \in \mathcal{D}$ for a coefficient $\alpha \in \mathbb{R}$ and can concentrate on the determination of the dictionary element d_{N+1}. Inserting (4.18) into (4.17) , yields

$$\left\| R^{N+1} \right\|^2 = \left\| R^N \right\|^2 - \left(\frac{\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\|_{\mathbb{R}^\ell}} \right)^2.$$

Data: $y \in \mathbb{R}^\ell$
Result: approximation $f_N^{(N)}$
initialization: $\mathcal{D}$, f_0, $R^0 = y - I\neg f_0$;
$N = 0$;
while *(stopping criteria not fulfilled)* **do**

$$d_{N+1} := \arg\max_{d\in\mathcal{D}} \left(\frac{\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\|_{\mathbb{R}^\ell}} \right)^2;$$

$$\alpha_{N+1}^{(N+1)} := \frac{\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d_{N+1} \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d_{N+1} \right\|_{\mathbb{R}^\ell}^2};$$

$$R^{N+1} := R^N - \alpha_{N+1}^{(N+1)} \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d_{N+1};$$

if $(N \geq 1)$ **then**
  $\alpha_n^{(N+1)} = \alpha_n^{(N)} - \alpha_{N+1}^{(N+1)} \beta_n^{(N)} (d_{N+1})$, $n = 1, ..., N$;
end
N be increased by 1;
end
return $f_N^{(N)} = \sum_{n=1}^N \alpha_n^{(N)} d_n$;

Algorithm 2: Pseudo-code for the orthogonal matching pursuit.

Again, we obtained an equivalent maximization problem: instead of minimizing $\|R^{N+1}\|^2$, we can equivalently seek the (generally not unique) dictionary element d_{N+1} by

$$d_{N+1} := \arg\max_{d\in\mathcal{D}} \left(\frac{\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_N^\perp} I\neg d \right\|_{\mathbb{R}^\ell}} \right)^2 .$$

Analogously as before, we maintain the notation with the maximum instead of a supremum. Note that, again, the next chosen dictionary element is the one which is most collinear to the residual. In the orthogonal matching pursuit, it is, simultaneously, also orthogonal to the previously chosen basis elements (see e. g. Telschow, 2014). Due to the latter property, each dictionary element can only be chosen once in an orthogonal matching pursuit. We obtain the pseudo-code for the orthogonal matching pursuit as given in Algorithm 2. Note that the algorithm allows certain update rules for an efficient implementation (e. g. with respect to the projection coefficients). We refer to Vincent and Bengio (2002) for this.

4.2. A particular dictionary and problem notation

Before we introduce the IPMP algorithms, we consider a particular dictionary in more detail. For spherical inverse problems with a global signal, the trial functions introduced in Chapter 2 and Chapter 3 are possible (and sensible) dictionary elements. Next, we will introduce a novel notation (compare Michel and Schneider, 2020) of a dictionary for spherical inverse problems of scalar signals including these trial functions in order to standardize the declaration of a dictionary for future publications. At first, we define sets of indices (compare with Freeden et al., 1998, p. 58).

Definition 4.2.1. *The set of pairs of indices of degree n and order j is given by*

$$\mathcal{N} := \{(n, j) \mid n \in \mathbb{N}_0,\ j = -n, ..., n\}.$$

Further, the set of pairs of a band-limit L and numbering k is defined as

$$\mathcal{L} := \left\{(k, L) \mid L \in \mathbb{N}_0,\ k = 1, ..., (L+1)^2\right\}.$$

The set $\mathcal{N}$ is used for the indices of spherical harmonics whereas the set $\mathcal{L}$ is used for the superscript indices of Slepian functions. Further, note that the group $\mathrm{SO}(3)$ has a universal 2-sheeted covering that is diffeomorphic with

$$\mathbb{S}^3 := \left\{ x \in \mathbb{R}^4 \;\middle|\; |x| = 1 \right\}$$

(see e. g. Bröcker and tom Dieck, 1985; Grafarend and Kühnel, 2011). As

$$[-1, 1] \times \mathbb{S}^3 = \mathbb{B}^4 := \left\{ x \in \mathbb{R}^4 \;\middle|\; |x| \leq 1 \right\}$$

we will use here the abbreviation $\mathbb{B}^4$ instead of $[-1, 1] \times \mathrm{SO}(3)$ for denoting a dictionary. Then we define a dictionary consisting of spherical harmonics, Slepian functions as well as Abel–Poisson low and band pass filters as follows.

Definition 4.2.2. *Let $N \subseteq \mathcal{N}$, $S \subseteq \mathbb{B}^4 \times \mathcal{L}$ and $B_K,\ B_W \subseteq \mathring{\mathbb{B}}$ for the open unit ball $\mathring{\mathbb{B}}$. We define the following (spherical scalar) trial function classes $[\cdot]_\bullet$ by*

$$[N]_{\mathrm{SH}} := \left\{ Y_{n,j} \;\middle|\; (n, j) \in N \right\}$$

*for **s**pherical **h**armonics,*

$$[S]_{\mathrm{SL}} := \left\{ g^{(k,L)} \left(\left(c, A\varepsilon^3 \right), \cdot \right) \;\middle|\; ((c, A), (k, L)) \in S \right\}$$

*for **Sl**epian functions,*

$$[B_K]_{\mathrm{APK}} := \{ K(x, \cdot) \mid x \in B_K \}$$

*for **A**bel **P**oisson low pass filters $K(x,\cdot)$ and*

$$[B_W]_{\mathrm{APW}} := \{W(x,\cdot) \mid x \in B_W\}$$

*for **A**bel **P**oisson band pass filters $W(x,\cdot)$. Then a (spherical scalar) dictionary $\mathcal{D} \subseteq \mathcal{H}_2(\Omega)\backslash\{0\} \subseteq \mathrm{L}^2(\Omega)\backslash\{0\}$ is defined as the union of trial function classes, i. e.*

$$\begin{aligned}\mathcal{D} &:= [N]_{\mathrm{SH}} + [S]_{\mathrm{SL}} + [B_K]_{\mathrm{APK}} + [B_W]_{\mathrm{APW}} \\ &:= [N]_{\mathrm{SH}} \cup [S]_{\mathrm{SL}} \cup [B_K]_{\mathrm{APK}} \cup [B_W]_{\mathrm{APW}}. \end{aligned} \tag{4.19}$$

Note that the notation of a trial function class is similar to the notation of an equivalence class in basic (linear) algebra. Further, note that, with respect to this definition, a dictionary can be finite as well as infinite.
If not all four trial function classes are considered, the dictionary is defined as the union of the actually used trial function classes. The notation can easily be transferred to other inverse problems. For instance, for spherical vectorial problems, we can use lower-case instead of capital letters for the subscript in the definition of a trial function class.

Remark 4.2.3. We immediately obtain that the dictionary from Definition 4.2.2 is complete in $\mathrm{L}^2(\Omega)$ and $\mathcal{H}_2(\Omega)$ if $[B_K]_{\mathrm{APK}}$ or $[B_W]_{\mathrm{APW}}$ fulfils the assumptions of either Lemma 3.3.19, 3.3.21, 3.4.11 or 3.4.13, i. e. $[B_K]_{\mathrm{APK}}$ or $[B_W]_{\mathrm{APW}}$ contains a basis system of the respective function space.

Further, we reckon a particular advantage of a matching pursuit if a dictionary like (4.19) is used. It is well-known that none of the introduced trial function classes is perfect for the approximation of a signal on its own. Each one was developed for a particular use. Most importantly, the functions of the diverse classes are distinct in their degree of localization. Thus, it seems sensible to use spherical harmonics to approximate global structures of a signal. However, adding global functions to an approximation has a global impact. Thus, localized functions produce better results if only local structures need to be added. However, if only local functions are used, the approximation can be very inaccurate in between the grid points at which we have given measurements. In this way, the combination of diverse trial function classes in one dictionary and an algorithm which automatically selects one suitable trial function after the other seems to be a promising approach for better results.

How to extend a matching pursuit to inverse problems – the idea We introduced an (orthogonal) matching pursuit in order to solve an approximation problem. However, in geomathematics, we are actually interested in inverse problems $Tx = y$. As shown before, if $T = I$ is the identity operator or $T = I\beth$ is a discretization operator (thus, the Hebrew letter $\beth$), an approximation problem can be viewed as a particular inverse problem. In that sense, we are interested in generalizations of the (orthogonal) matching pursuit to inverse problems.

For this, we obviously have to take the operator T as well as its ill-posedness into account. The latter one is usually treated with the use of a regularization like the Tikhonov-Philipps regularization, see Example 2.5.17. Both matching pursuits introduced before aim to minimize the norm of the current residual in each iteration. If we compare this approach with the Tikhonov-Philipps regularization, we see that each of the matching pursuits aims to minimize only a data fidelity term. Thus, the matching pursuits for inverse problem aim to minimize both a data fidelity as well as a penalty term in each iteration in order to solve the problem and treat the ill-posedness as well.
We introduce some (to Definition 4.1.3, Remark 4.1.4 and Definition 4.1.5) additional notation that is needed when discussing matching pursuits for inverse problems (compare with Michel and Schneider, 2020).

Remark 4.2.4. We consider a spherical scalar inverse problem where the right-hand side $y \in \mathbb{R}^\ell$ is a vector of measurements from a physical experiment. We set $X = \mathcal{H}_2(\Omega)$ for the function space of the solution as well as the approximation. Thus, for a better readability, we write $f \in \mathcal{H}_2(\Omega)$ instead of $x \in X$. Note that in previous publications, the choice $\mathrm{L}^2(\Omega)$ was also used. However, as we pointed out in Lemma 3.2.8 and Remark 3.2.9, we obtain a smoother (and, in practice, better) approximation if we demand $f \in \mathcal{H}_2(\Omega)$. Note that it also holds $\mathcal{D} \subseteq \mathcal{H}_2(\Omega)\backslash\{0\}$ for $\mathcal{D}$ as defined in Definition 4.2.2 (see Remark 4.2.3).
Further, we have $Y = \mathbb{R}^\ell$ for $\ell \in \mathbb{N}$ and the right-hand side $y \in \mathbb{R}^\ell$ similar to Remark 4.1.1. However, now, each y_i, $i = 1, .., \ell$, depends on a grid point $\sigma\eta^{(i)} \in \mathbb{R}^3$ with $1 < \sigma \in \mathbb{R}$ and $\eta^{(i)} \in \Omega$. In particular, we have $y_i = (Tf)(\sigma\eta^{(i)})$. The operator T is exchanged by a system of (discretization) functionals $T\tilde{\gamma}^i$ such that $T\tilde{\gamma}^i f = (Tf)(\sigma\eta^{(i)}) = y_i$ holds for $i = 1, ..., \ell$. Again, the Hebrew letter $\gimel$ is used to emphasize that the system $(T\tilde{\gamma}^i)_{i=1,...,\ell}$ is a discretization of T. Then we can write $T\gimel := (T\tilde{\gamma}^i)_{i=1,...,\ell}$ and have the linear inverse problem in the familiar formulation

$$T\gimel f = y.$$

Furthermore, the Tikhonov-Philipps functional is given by

$$\mathcal{J}\left(f; T\gimel, \lambda\left(\delta, y^\delta\right), y^\delta\right) := \left\|y^\delta - T\gimel f\right\|^2_{\mathbb{R}^\ell} + \lambda\left(\delta, y^\delta\right)\|f\|^2_{\mathcal{H}_2(\Omega)} \tag{4.20}$$

for perturbed data y^δ and a regularization parameter $\lambda(\delta, y^\delta)$ dependent on the perturbed data and the noise level δ.

Next, we introduce the IPMP algorithms in detail. In particular, we explain methods that are advancements of the classical matching pursuit as well as the orthogonal matching pursuit.

4.3. The regularized functional matching pursuit

We start with the introduction of the **r**egularized **f**unctional **m**atching **p**ursuit (RFMP) algorithm. This algorithm is the advancement of the classical matching pursuit to inverse problems. First, we recall details of its derivation. Then we summarize its theoretical results. We end this section with some aspects of the algorithm in practice. This section is based on Kontak (2018); Michel (2015a); Michel and Orzlowski (2017); Telschow (2014), although there exists a far wider range of literature on the algorithm (see e. g. Berkel et al., 2011; Fischer, 2011; Fischer and Michel, 2012, 2013a,b; Gutting et al., 2017; Kontak and Michel, 2019, 2018; Michel and Telschow, 2014).
We use the notation introduced in Definitions 4.1.3 and 4.1.5 (with $T\neg$ instead of $I\neg$) as well as Remarks 4.1.4 and 4.2.4.

Derivation of the RFMP algorithm In the classical matching pursuit, we considered the minimization of $\|R^{N+1}\|_{\mathbb{R}^\ell}$ with respect to the next coefficient α_{N+1} and dictionary element d_{N+1} for an approximation f_{N+1}. Now, we do the same but for the Tikhonov-Philipps functional of our discretized setting, see (4.20). Again, we aim to approximate the solution f iteratively by a linear combination of dictionary elements $d \in \mathcal{D}$. Thus, we consider

$$\begin{aligned}\mathcal{J}\left(f_N + \alpha d; T\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) \\ := \left\|y^\delta - T\neg(f_N + \alpha d)\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\|f_N + \alpha d\|_{\mathcal{H}_2(\Omega)}^2 . \quad (4.21)\end{aligned}$$

in the N-th iteration. Then the objective is to determine

$$\begin{aligned}(\alpha_{N+1}, d_{N+1}) &:= \underset{(\alpha,d)\in\mathbb{R}\times\mathcal{D}}{\arg\min}\left(\mathcal{J}\left(f_N + \alpha d; T\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right)\right) \\ &= \underset{(\alpha,d)\in\mathbb{R}\times\mathcal{D}}{\arg\min}\left(\left\|y^\delta - T\neg(f_N + \alpha d)\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\|f_N + \alpha d\|_{\mathcal{H}_2(\Omega)}^2\right) \\ &= \underset{(\alpha,d)\in\mathbb{R}\times\mathcal{D}}{\arg\min}\left(\left\|R^N - \alpha T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\|f_N + \alpha d\|_{\mathcal{H}_2(\Omega)}^2\right) \quad (4.22)\end{aligned}$$

for a dictionary $\mathcal{D}$ and the residual R^N given as in (4.7) (exchange $I\neg$ with $T\neg$ there). Note that we take over the notation with the minimum instead of an infimum from the non-regularized case. Again, we consider the optimal α first:

$$0 = \frac{\partial}{\partial\alpha}\left(\left\|R^N - \alpha T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\|f_N + \alpha d\|_{\mathcal{H}_2(\Omega)}^2\right)$$

$$
\begin{aligned}
&= \frac{\partial}{\partial \alpha}\left(\left\|R^N\right\|_{\mathbb{R}^\ell}^2 - 2\alpha\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell} + \alpha^2\left\|T\neg d\right\|_{\mathbb{R}^\ell}^2\right.\\
&\qquad \left.+\lambda\left(\delta, y^\delta\right)\left\|f_N\right\|_{\mathcal{H}_2(\Omega)}^2 + 2\alpha\lambda\left(\delta, y^\delta\right)\left\langle f_N, d\right\rangle_{\mathcal{H}_2(\Omega)} + \alpha^2\lambda\left(\delta, y^\delta\right)\left\|d\right\|_{\mathcal{H}_2(\Omega)}^2\right)\\
&= -2\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell} + 2\alpha\left\|T\neg d\right\|_{\mathbb{R}^\ell}^2 + 2\lambda\left\langle f_N, d\right\rangle_{\mathcal{H}_2(\Omega)} + 2\alpha\lambda\|d\|_{\mathcal{H}_2(\Omega)}^2
\end{aligned}
$$

with the abbreviation $\lambda := \lambda\left(\delta, y^\delta\right)$ in the last equation. Similar as before, we obtain a formula for the coefficient α dependent on a dictionary element d:

$$
\alpha = \frac{\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle f_N, d\right\rangle_{\mathcal{H}_2(\Omega)}}{\left\|T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|d\right\|_{\mathcal{H}_2(\Omega)}^2}. \tag{4.23}
$$

Hence, again, it is sufficient to consider the minimization of (4.22) only with respect to the dictionary element d. If we insert (4.23) into (4.21), we obtain

$$
\begin{aligned}
&\left\|R^{N+1}\right\|^2 + \lambda\left(\delta, y^\delta\right)\left\|f_{N+1}\right\|_{\mathcal{H}_2(\Omega)}^2\\
&= \left\|R^N - \alpha T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|f_N + \alpha d\right\|_{\mathcal{H}_2(\Omega)}^2\\
&= \left\|R^N\right\|_{\mathbb{R}^\ell}^2 - 2\alpha\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell} + \alpha^2\left\|T\neg d\right\|_{\mathbb{R}^\ell}^2\\
&\quad + \lambda\left(\delta, y^\delta\right)\left\|f_N\right\|_{\mathcal{H}_2(\Omega)}^2 + 2\alpha\lambda\left(\delta, y^\delta\right)\left\langle f_N, d\right\rangle_{\mathcal{H}_2(\Omega)} + \alpha^2\lambda\left(\delta, y^\delta\right)\left\|d\right\|_{\mathcal{H}_2(\Omega)}^2\\
&= \left\|R^N\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|f_N\right\|_{\mathcal{H}_2(\Omega)}^2\\
&\quad - 2\frac{\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle f_N, d\right\rangle_{\mathcal{H}_2(\Omega)}}{\left\|T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|d\right\|_{\mathcal{H}_2(\Omega)}^2}\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell}\\
&\quad + 2\frac{\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle f_N, d\right\rangle_{\mathcal{H}_2(\Omega)}}{\left\|T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|d\right\|_{\mathcal{H}_2(\Omega)}^2}\lambda\left(\delta, y^\delta\right)\left\langle f_N, d\right\rangle_{\mathcal{H}_2(\Omega)}\\
&\quad + \left(\frac{\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle f_N, d\right\rangle_{\mathcal{H}_2(\Omega)}}{\left\|T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|d\right\|_{\mathcal{H}_2(\Omega)}^2}\right)^2\left\|T\neg d\right\|_{\mathbb{R}^\ell}^2\\
&\quad + \left(\frac{\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle f_N, d\right\rangle_{\mathcal{H}_2(\Omega)}}{\left\|T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|d\right\|_{\mathcal{H}_2(\Omega)}^2}\right)^2\lambda\left(\delta, y^\delta\right)\left\|d\right\|_{\mathcal{H}_2(\Omega)}^2\\
&= \left\|R^N\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|f_N\right\|_{\mathcal{H}_2(\Omega)}^2 - 2\frac{\left(\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle f_N, d\right\rangle_{\mathcal{H}_2(\Omega)}\right)^2}{\left\|T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|d\right\|_{\mathcal{H}_2(\Omega)}^2}\\
&\quad + \frac{\left(\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle f_N, d\right\rangle_{\mathcal{H}_2(\Omega)}\right)^2}{\left\|T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|d\right\|_{\mathcal{H}_2(\Omega)}^2}
\end{aligned}
$$

$$= \left\| R^N \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \|f_N\|_{\mathcal{H}_2(\Omega)}^2 - \frac{\left(\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right) \langle f_N, d\rangle_{\mathcal{H}_2(\Omega)}\right)^2}{\|T\neg d\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \|d\|_{\mathcal{H}_2(\Omega)}^2}. \tag{4.24}$$

Thus, the minimization of (4.22), i. e. of the Tikhonov-Philipps functional of the N-th iteration, is equivalent to choosing

$$d_{N+1} \coloneqq \arg\max_{d\in\mathcal{D}} \frac{\left(\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right) \langle f_N, d\rangle_{\mathcal{H}_2(\Omega)}\right)^2}{\|T\neg d\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \|d\|_{\mathcal{H}_2(\Omega)}^2} \tag{4.25}$$

and α_{N+1} via (4.23) with $d = d_{N+1}$. In analogy to above, we use the maximum instead of the supremum here. Note that this derivation is independent of whether the dictionary is finite or infinite. Nonetheless, a solution might not be unique. However, it is obvious that, if we are given an infinite dictionary, the determination of the maximizer of (4.25) among the dictionary is not trivial. Thus, in previous works, only finite dictionaries were feasible for practical purposes.

Remark 4.3.1. In this thesis, the RFMP algorithm is always used with a finite dictionary. Then the value of the maximization objective (4.25) can be computed and compared for all dictionary elements $d \in \mathcal{D}$ in order to obtain the next basis element d_{N+1} for the approximation f_{N+1}.

A pseudo-code of the RFMP algorithm is given in Algorithm 3. Note that for $\lambda(\delta, y^\delta) = 0$, the RFMP algorithm is also called **f**unctional **m**atching **p**ursuit (FMP) algorithm and coincides with the classical matching pursuit if $T\neg = I\neg$. Further note that, similar to the classical matching pursuit, the RFMP algorithm (with a finite dictionary) can be implemented very efficiently (see Michel, 2015a; Telschow, 2014). The values $T\neg d$ as well as $\langle d, \tilde{d}\rangle_{\mathcal{H}_2(\Omega)}$ for $d,\ \tilde{d} \in \mathcal{D}$ can be computed before the actual iteration process. Furthermore, the additional update rule

$$\langle f_{N+1}, d\rangle_{\mathcal{H}_2(\Omega)} = \langle f_N, d\rangle_{\mathcal{H}_2(\Omega)} + \alpha_{N+1}\langle d_{N+1}, d\rangle_{\mathcal{H}_2(\Omega)} \tag{4.26}$$

for $d \in \mathcal{D}$ can be used for the computation of the nominators of (4.23) and (4.25), respectively. For an overview on an implementation routine including those preprocessing tools, see also Appendix B.

A summary of theoretical results of the RFMP algorithm There exist several theoretical results for the RFMP algorithm. We summarize them at this point. As we generalize the RFMP algorithm to a learning algorithm in Chapter 7, we will state the results in a theoretical view on the learning algorithm in Chapter 8 in more detail.
For the RFMP algorithm, the convergence of the sequence of Tikhonov functionals (confer (4.21)), the coefficients and the residual is known. With these results, the convergence of the approximation dependent on the dictionary can be

Data: $y \in \mathbb{R}^\ell$
Result: approximation f_N
initialization: $\mathcal{D}$, f_0, $R^0 = y - T\neg f_0$;
$N = 0$;
while *(stopping criteria not fulfilled)* **do**

$$d_{N+1} := \arg\max_{d\in\mathcal{D}} \frac{\left(\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\langle f_N, d\rangle_{\mathcal{H}_2(\Omega)}\right)^2}{\|T\neg d\|^2_{\mathbb{R}^\ell} + \lambda\left(\delta, y^\delta\right)\|d\|^2_{\mathcal{H}_2(\Omega)}};$$

$$\alpha_{N+1} := \frac{\left\langle R^N, T\neg d_{N+1}\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\langle f_N, d_{N+1}\rangle_{\mathcal{H}_2(\Omega)}}{\|T\neg d_{N+1}\|^2_{\mathbb{R}^\ell} + \lambda\left(\delta, y^\delta\right)\|d_{N+1}\|^2_{\mathcal{H}_2(\Omega)}};$$

$$R^{N+1} := R^N - \alpha_{N+1} T\neg d_{N+1};$$

N be increased by 1;

end
return $f_N = \sum_{n=1}^N \alpha_n d_n$;

Algorithm 3: Pseudo-code for the regularized functional matching pursuit (RFMP).

proven. Furthermore, there are results for the stability of the approximation and the convergence of the regularization. At last, the convergence rate of the residuals as well as the approximation can also be estimated. These results are published, for instance, in Kontak (2018); Kontak and Michel (2019); Michel (2015b); Michel and Orzlowski (2017).

4.4. The regularized orthogonal functional matching pursuit

Though the RFMP algorithm works well in practice, it has the same optimality problems as the classical matching pursuit: again, it would probably be better if the next dictionary element d_{N+1} and all coefficients α_n, $n = 1, ..., N+1$, were chosen simultaneously. Thus, on the basis of the orthogonal matching pursuit, the **r**egularized **o**rthogonal **f**unctional **m**atching **p**ursuit (ROFMP) algorithm was developed for spherical inverse problems. Analogously as before, we derive the orthogonal IPMP algorithm in this section. Then we consider the projection coefficients, which were first used in (4.14) and will also play a role in the ROFMP algorithm, in more detail. At last, we summarize the theoretical results of the ROFMP algorithm and make some comments on its practical realization. This section is based on Michel and Telschow (2016); Telschow (2014).

Derivation of the ROFMP algorithm First of all, note that we use the same notation for the inverse problem as given in Definitions 4.1.3 and 4.1.5 (with $T\neg$ instead of $I\neg$) as well as Remarks 4.1.4 and 4.2.4. Further, as we have $T\neg d \in \mathbb{R}^\ell$ for all dictionary elements $d \in \mathcal{D}$, we rewrite (4.2) and (4.3) as

$$\mathcal{V} = \mathcal{V}_\mathcal{D} = \overline{\operatorname{span}\{T\neg d \mid d \in \mathcal{D}\}}^{\mathbb{R}^\ell}$$

and

$$\mathcal{V}_N = \mathcal{V}_{\mathcal{D}_N} = \overline{\operatorname{span}\{T\neg \mathcal{D}_N\}}^{\mathbb{R}^\ell},\ T\neg\mathcal{D}_N \coloneqq \{T\neg d_n \mid d_n \in \mathcal{D},\ n = 1,...,N\}. \quad (4.27)$$

Similarly, we use $\mathcal{V}^\perp$ and $\mathcal{V}_N^\perp$ for their Euclidean orthogonal complements. To extend the orthogonal matching pursuit to inverse problems, we use a similar approach as with the RFMP algorithm. However, we now consider (confer (4.22))

$$\begin{aligned}\left(\alpha^{(N+1)}, d_{N+1}\right) \coloneqq \operatorname*{arg\,min}_{(\alpha,d)\in\mathbb{R}^{N+1}\times\mathcal{D}} \Bigg(&\left\| y^\delta - \sum_{n=1}^N \alpha_n T\neg d_n - \alpha_{N+1} T\neg d_{N+1} \right\|_{\mathbb{R}^\ell}^2 \\ &+\lambda\left(\delta, y^\delta\right) \left\| \sum_{n=1}^N \alpha_n T\neg d_n + \alpha_{N+1} T\neg d_{N+1} \right\|_{\mathcal{H}_2(\Omega)}^2 \Bigg) \end{aligned} \quad (4.28)$$

for a vector of coefficients

$$\alpha = (\alpha_1, ..., \alpha_{N+1}) \in \mathbb{R}^{N+1}.$$

Also here, we maintain the formulation with the minimum as was done in the literature instead of using the infimum. From the orthogonal matching pursuit, we obtain that the first term in the sum of (4.28) can be rearranged to

$$\left\| R^N - \alpha \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\|_{\mathbb{R}^\ell}^2$$

for $\alpha \in \mathbb{R}$ and $d \in \mathcal{D}$ (confer (4.17)). The second term can then be written as follows in order to assure that the images of chosen dictionary elements under $T\neg$ are orthogonal in $\mathbb{R}^\ell$. Assume we are in the N-th iteration, i. e. the approximation $f_N^{(N)}$ (confer (4.16)) is given as an orthogonal expansion. We aim to obtain

$$f_{N+1}^{(N+1)} = \sum_{n=1}^{N+1} \alpha_n^{(N+1)} d_n,$$

with updated coefficients $\alpha_n^{(N+1)},\ n = 1, ..., N+1$. Due to (4.15), (4.16) and with the abbreviation

$$b_N^{(N)}(d_{N+1}) \coloneqq \sum_{n=1}^N \beta_n^{(N)}(d_{N+1}) d_n,$$

this is equivalent to

$$\begin{aligned} f_{N+1}^{(N+1)} &= \sum_{n=1}^{N} \alpha_n^{(N)} d_n - \alpha_{N+1} \sum_{n=1}^{N} \beta_n^{(N)} (d_{N+1}) d_n + \alpha_{N+1}^{(N+1)} d_{N+1} \\ &= f_N^{(N)} - \alpha_{n+1}^{(N+1)} b_N^{(N)} (d_{N+1}) + \alpha_{N+1}^{(N+1)} d_{N+1} \\ &= f_N^{(N)} + \alpha_{N+1}^{(N+1)} \left(d_{N+1} - b_N^{(N)} (d_{N+1}) \right) \end{aligned} \tag{4.29}$$

which also leaves us with the determination of $\alpha \in \mathbb{R}$ and $d \in \mathcal{D}$. All in all, in contrast to the minimization of $\mathcal{J}$ (see (4.21)) in the RFMP algorithm, the functional we minimize in the ROFMP algorithm in order to implement a pre-fitting technique is given by

$$\begin{aligned} &\mathcal{J}_O \left(f_N^{(N)} + \alpha d; T\neg, \lambda \left(\delta, y^\delta \right), y^\delta \right) \\ &\qquad := \left\| R^N - \alpha \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\|_{\mathbb{R}^\ell}^2 + \lambda \left(\delta, y^\delta \right) \left\| f_N^{(N)} + \alpha \left(d - b_N^{(N)} (d) \right) \right\|_{\mathcal{H}_2(\Omega)}^2 . \end{aligned} \tag{4.30}$$

That means, we seek

$$\begin{aligned} &(\alpha_{N+1}, d_{N+1}) \\ &= \underset{(\alpha, d) \in \mathbb{R} \times \mathcal{D}}{\arg\min} \left(\left\| R^N - \alpha \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\|_{\mathbb{R}^\ell}^2 + \lambda \left(\delta, y^\delta \right) \left\| f_N^{(N)} + \alpha \left(d - b_N^{(N)} (d) \right) \right\|_{\mathcal{H}_2(\Omega)}^2 \right) \end{aligned} \tag{4.31}$$

in the N-th iteration. Note that we take over the formulation with the minimum (instead of the infimum) from before. Again, we consider the coefficient α on its own:

$$\begin{aligned} 0 &= \frac{\partial}{\partial \alpha} \left(\left\| R^N - \alpha \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\|_{\mathbb{R}^\ell}^2 + \lambda \left(\delta, y^\delta \right) \left\| f_N^{(N)} + \alpha \left(d - b_N^{(N)} (d) \right) \right\|_{\mathcal{H}_2(\Omega)}^2 \right) \\ &= \frac{\partial}{\partial \alpha} \left(\left\| R^N \right\|_{\mathbb{R}^\ell}^2 - 2\alpha \left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\rangle_{\mathbb{R}^\ell} + \alpha^2 \left\| \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\|_{\mathbb{R}^\ell}^2 \right. \\ &\qquad + \lambda \left(\delta, y^\delta \right) \left\| f_N^{(N)} \right\|_{\mathcal{H}_2(\Omega)}^2 + 2\alpha\lambda \left(\delta, y^\delta \right) \left\langle f_N^{(N)}, d - b_N^{(N)} (d) \right\rangle_{\mathcal{H}_2(\Omega)} \\ &\qquad \left. + \alpha^2 \lambda \left(\delta, y^\delta \right) \left\| d - b_N^{(N)} (d) \right\|_{\mathcal{H}_2(\Omega)}^2 \right) \\ &= -2 \left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\rangle_{\mathbb{R}^\ell} + 2\alpha \left\| \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\|_{\mathbb{R}^\ell}^2 \\ &\qquad + 2\lambda \left(\delta, y^\delta \right) \left\langle f_N^{(N)}, d - b_N^{(N)} (d) \right\rangle_{\mathcal{H}_2(\Omega)} + 2\alpha\lambda \left(\delta, y^\delta \right) \left\| d - b_N^{(N)} (d) \right\|_{\mathcal{H}_2(\Omega)}^2 . \end{aligned}$$

Thus, we have the coefficient α given by

$$\alpha = \frac{\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right) \left\langle f_N^{(N)}, d - b_N^{(N)}(d) \right\rangle_{\mathcal{H}_2(\Omega)}}{\left\| \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \left\| d - b_N^{(N)}(d) \right\|_{\mathcal{H}_2(\Omega)}^2}. \tag{4.32}$$

The coefficient depends on the chosen dictionary element and, once again, it suffices to determine the latter one. Inserting (4.32) into the functional $\mathcal{J}_O$, we obtain – similarly as with the RFMP algorithm – that the minimization of $\mathcal{J}_O$ with respect to $\alpha \in \mathbb{R}$ and $d \in \mathcal{D}$ reformulates in the following way:

$$\begin{aligned}
&\left\| R^{N+1} \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \left\| f_{N+1}^{(N+1)} \right\|_{\mathcal{H}_2(\Omega)}^2 \\
&= \left\| R^N \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \left\| f_N^{(N)} \right\|_{\mathcal{H}_2(\Omega)}^2 \\
&\quad - 2\alpha \left(\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\rangle_{\mathbb{R}^\ell} + \lambda\left(\delta, y^\delta\right) \left\langle f_N^{(N)}, d - b_N^{(N)}(d) \right\rangle_{\mathcal{H}_2(\Omega)} \right) \\
&\quad + \alpha^2 \left(\left\| \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \left\| d - b_N^{(N)}(d) \right\|_{\mathcal{H}_2(\Omega)}^2 \right) \\
&= \left\| R^N \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \left\| f_N^{(N)} \right\|_{\mathcal{H}_2(\Omega)}^2 \\
&\quad - 2 \frac{\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right) \left\langle f_N^{(N)}, d - b_N^{(N)}(d) \right\rangle_{\mathcal{H}_2(\Omega)}}{\left\| \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \left\| d - b_N^{(N)}(d) \right\|_{\mathcal{H}_2(\Omega)}^2} \\
&\qquad \times \left(\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\rangle_{\mathbb{R}^\ell} + \lambda\left(\delta, y^\delta\right) \left\langle f_N^{(N)}, d - b_N^{(N)}(d) \right\rangle_{\mathcal{H}_2(\Omega)} \right) \\
&\quad + \left(\frac{\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right) \left\langle f_N^{(N)}, d - b_N^{(N)}(d) \right\rangle_{\mathcal{H}_2(\Omega)}}{\left\| \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \left\| d - b_N^{(N)}(d) \right\|_{\mathcal{H}_2(\Omega)}^2} \right)^2 \\
&\qquad \times \left(\left\| \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \left\| d - b_N^{(N)}(d) \right\|_{\mathcal{H}_2(\Omega)}^2 \right)
\end{aligned}$$

$$
\begin{aligned}
&= \left\|R^N\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|f_N^{(N)}\right\|_{\mathcal{H}_2(\Omega)}^2 \\
&\quad - 2\frac{\left(\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle f_N^{(N)}, d - b_N^{(N)}(d)\right\rangle_{\mathcal{H}_2(\Omega)}\right)^2}{\left\|\mathcal{P}_{\mathcal{V}_N^\perp} T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|d - b_N^{(N)}(d)\right\|_{\mathcal{H}_2(\Omega)}^2} \\
&\quad + \frac{\left(\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle f_N^{(N)}, d - b_N^{(N)}(d)\right\rangle_{\mathcal{H}_2(\Omega)}\right)^2}{\left\|\mathcal{P}_{\mathcal{V}_N^\perp} T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|d - b_N^{(N)}(d)\right\|_{\mathcal{H}_2(\Omega)}^2} \\
&= \left\|R^N\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|f_N^{(N)}\right\|_{\mathcal{H}_2(\Omega)}^2 \\
&\quad - \frac{\left(\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle f_N^{(N)}, d - b_N^{(N)}(d)\right\rangle_{\mathcal{H}_2(\Omega)}\right)^2}{\left\|\mathcal{P}_{\mathcal{V}_N^\perp} T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|d - b_N^{(N)}(d)\right\|_{\mathcal{H}_2(\Omega)}^2}.
\end{aligned} \tag{4.33}
$$

Thus, we can equivalently determine (the generally non-unique) d_{N+1} via the maximization

$$
d_{N+1} := \operatorname*{arg\,max}_{d\in\mathcal{D}} \frac{\left(\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle f_N^{(N)}, d - b_N^{(N)}(d)\right\rangle_{\mathcal{H}_2(\Omega)}\right)^2}{\left\|\mathcal{P}_{\mathcal{V}_N^\perp} T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|d - b_N^{(N)}(d)\right\|_{\mathcal{H}_2(\Omega)}^2} \tag{4.34}
$$

with respect to $d \in \mathcal{D}$. Note again that these derivations are independent of the size of the dictionary and we formulate (4.34) with the maximum instead of the supremum as this is usually done in the literature.

Remark 4.4.1. In this thesis, the ROFMP algorithm will always be used with a finite dictionary. Then the values (4.34) can be computed and compared for all dictionary elements in order to obtain the next basis element d_{N+1}.

Further, note that, if $\lambda\left(\delta, y^\delta\right) = 0$ and $T\neg$ is the discretization of the identity, the ROFMP algorithm coincides with the orthogonal matching pursuit.

The iterated ROFMP As we will see with the theoretical results of the ROFMP algorithm, the residual R^N will be constant after a certain number of iterations. As this probably will not be a desired approximation, an iterated version of the ROFMP algorithm was developed by Telschow (2014) as well. This algorithm is similar to the use of an iterated Tikhonov-Philipps regularization, but enforces more accuracy. In particular, after a certain number of iterations $K \in \mathbb{N}$, the ROFMP algorithm is "restarted".

For a description, we extend our previous notation. In the iterated ROFMP algorithm, we divide the iterations into parts of the size of K. Each part is called an ROFMP step. In theory, we allow the iterated ROFMP algorithm to make infinitely many ROFMP steps $k \in \mathbb{N}_0$. The set of images under $T_\neg$ of previously chosen dictionary elements as well as its orthogonal complement are then given by

$$ {}_k\mathcal{V}_N \qquad \text{and} \qquad {}_k\mathcal{V}_N^\perp, \qquad 0 \leq N \leq K,\ k \in \mathbb{N}_0. $$

The restart is done by setting all current projection coefficients to zero at the beginning of an ROFMP step k, i. e. we have

$$ {}_k\mathcal{V}_0 = \emptyset \qquad \text{and} \qquad {}_k\mathcal{V}_0^\perp = \mathbb{R}^\ell. $$

In this way, previously chosen dictionary elements $d_n,\ n = 1, ..., K$, can be chosen a second time which allows again an improvement of the approximation in the next ROFMP step. Further, we analogously have ${}_kb_N^{(N)}$ and, in particular, ${}_kb_0^{(0)} = 0$. As the projection coefficients are set to zero after the ROFMP step k_0, the coefficients ${}_{k_0}\alpha_n^{(K)}$ for $n = 1, ..., K$ will not be updated in the next ROFMP steps $k > k_0$. That means, we fix the obtained approximation at the end of every ROFMP step. Thus, for the approximation, we set

$$ {}_kf_N^{(N)} = {}_{k-1}f_K^{(K)} + \sum_{n=1}^{N} {}_k\alpha_n^{(N)}\, {}_kd_n, \qquad 1 \leq N \leq K,\ k \in \mathbb{N}. \tag{4.35} $$

This is done in order to improve the accuracy in the result of the algorithm in comparison to if we set ${}_kf_0^{(0)} = 0$. At last, for the residual, we have ${}_1R^0 = y$ and

$$ \begin{aligned} {}_kR^N &= y - T_\neg {}_kf_N^{(N)} \\ &= y - T_\neg {}_{k-1}f_K^{(K)} - \sum_{n=1}^{N} {}_k\alpha_n^{(N)} T_\neg\, {}_kd_n \\ &= {}_{k-1}R^K - \sum_{n=1}^{N} {}_k\alpha_n^{(N)} T_\neg\, {}_kd_n \\ &= {}_kR^{N-1} - {}_k\alpha_N^{(N)} \mathcal{P}_{{}_k\mathcal{V}_{N-1}^\perp} T_\neg\, {}_kd_N, \end{aligned} $$

which implies ${}_kR^0 = {}_{k-1}R^K$. Then the iterated ROFMP considers the minimization problem

$$ \begin{aligned} ({}_k\alpha_{N+1}, {}_kd_{N+1}) = \underset{(\alpha,d)\in\mathbb{R}\times\mathcal{D}}{\arg\min} \Bigg(& \left\| {}_kR^N - \alpha \mathcal{P}_{{}_k\mathcal{V}_N^\perp} T_\neg d \right\|_{\mathbb{R}^\ell}^2 \\ & + \lambda\left(\delta, y^\delta\right) \left\| {}_kf_N^{(N)} + \alpha\left(d - {}_kb_N^{(N)}(d)\right) \right\|_{\mathcal{H}_2(\Omega)}^2 \Bigg) \end{aligned} $$

or the maximization

$$ {}_kd_{N+1} \coloneqq \operatorname*{arg\,max}_{d\in\mathcal{D}} \frac{\left(\left\langle {}_kR^N, \mathcal{P}_{{}_k\mathcal{V}_N^\perp} T\neg d \right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right) \left\langle {}_kf_N^{(N)}, d - {}_kb_N^{(N)}(d) \right\rangle_{\mathcal{H}_2(\Omega)} \right)^2}{\left\| \mathcal{P}_{{}_k\mathcal{V}_N^\perp} T\neg d \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \left\| d - {}_kb_N^{(N)}(d) \right\|_{\mathcal{H}_2(\Omega)}^2}, $$

respectively. Note that, by construction, the choice of the next dictionary element is generally not unique in the iterated ROFMP as well. Further, note once more that we took over the formulations with the minimum and maximum instead of the infimum and supremum, respectively, as this is in accordance to the literature. Experiments have shown that the iterated ROFMP algorithm improves the basic ROFMP algorithm very well. A pseudo-code for the iterated ROFMP algorithm is given in Algorithm 4.

Projection coefficients in detail At this point, we discuss the terms $\beta_n^{(N)}(d)$ for $d \in \mathcal{D}$ and $n = 1, ..., N,\ N \in \mathbb{N}$, in more detail. They are not unique in general and we follow the idea from Telschow (2014). Note that the following considerations have not been stated in previous publications with respect to the ROFMP algorithm. The projection coefficients were first used in (4.5) with respect to the orthogonal matching pursuit. They occur in the (Euclidean) projection onto the span of $\mathcal{V}_N$ as given in (4.27). In the setting of the ROFMP algorithm, this means, we have

$$ \mathcal{P}_{\mathcal{V}_N} T\neg d = \sum_{n=1}^{N} \beta_n^{(N)}(d) T\neg d_n. $$

The terms $\beta_n^{(N)}(d)$ are related to the Gram-Schmidt orthonormalization scheme. However, for practical purposes, we need to obtain a recursive as well as an explicit definition of the coefficients in the iteration N. We start with a recursive definition which is useful for an efficient implementation.

Theorem 4.4.2. *The projection coefficients $\beta_n^{(N)}(d),\ n = 1, ..., N,\ N \in \mathbb{N}$, are given by*

$$ \beta_N^{(N)}(d) \coloneqq \frac{\left\langle \mathcal{P}_{\mathcal{V}_{N-1}^\perp} T\neg d, \mathcal{P}_{\mathcal{V}_{N-1}^\perp} T\neg d_N \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_{N-1}^\perp} T\neg d_N \right\|_{\mathbb{R}^\ell}^2} = \frac{\left\langle T\neg d, \mathcal{P}_{\mathcal{V}_{N-1}^\perp} T\neg d_N \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_{N-1}^\perp} T\neg d_N \right\|_{\mathbb{R}^\ell}^2} \tag{4.36} $$

$$ \beta_n^{(N)}(d) \coloneqq \beta_n^{(N-1)}(d) - \beta_N^{(N)}(d)\beta_n^{(N-1)}(d_n), \qquad n = 1, ..., N-1, \tag{4.37} $$

for any dictionary element $d \in \mathcal{D}$. Then it holds

$$ \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d = T\neg d - \sum_{n=1}^{N} \beta_n^{(N)}(d) T\neg d_n. \tag{4.38} $$

Data: $y \in \mathbb{R}^\ell$
Result: approximation ${}_kf_N^{(N)}$
initialization: $\mathcal{D},\ f_0,\ R^0 = y - T\neg f_0,\ K \in \mathbb{N}$;
$N = 0$;
while *(stopping criteria not fulfilled)* **do**

$$
\begin{aligned}
{}_kd_{N+1} &:= \arg\max_{d\in\mathcal{D}} \frac{\left(\left\langle {}_kR^N, \mathcal{P}_{{}_k\mathcal{V}_N^\perp} T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle {}_kf_N^{(N)},\, d - {}_kb_N^{(N)}(d)\right\rangle_{\mathcal{H}_2(\Omega)}\right)^2}{\left\|\mathcal{P}_{{}_k\mathcal{V}_N^\perp} T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|d - {}_kb_N^{(N)}(d)\right\|_{\mathcal{H}_2(\Omega)}^2};\\
{}_k\alpha_{N+1}^{(N+1)} &:= \frac{\left\langle {}_kR^N, \mathcal{P}_{{}_k\mathcal{V}_N^\perp} T\neg\, {}_kd_{N+1}\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle {}_kf_N^{(N)},\, {}_kd_{N+1} - {}_kb_N^{(N)}({}_kd_{N+1})\right\rangle_{\mathcal{H}_2(\Omega)}}{\left\|\mathcal{P}_{{}_k\mathcal{V}_N^\perp} T\neg\, {}_kd_{N+1}\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|{}_kd_{N+1} - {}_kb_N^{(N)}({}_kd_{N+1})\right\|_{\mathcal{H}_2(\Omega)}^2};\\
{}_kR^{N+1} &:= {}_kR^N - {}_k\alpha_{N+1}^{(N+1)} \mathcal{P}_{{}_k\mathcal{V}_N^\perp} T\neg\, {}_kd_{N+1};
\end{aligned}
$$

if $(N \geq 1)$ **then**
  ${}_k\alpha_n^{(N+1)} := {}_k\alpha_n^{(N)} - {}_k\alpha_{N+1}^{(N+1)}\, {}_k\beta_n^{(N)}\ ({}_kd_{N+1}),\ n = 1, ..., N$;
end
N be increased by 1;
if $(N == K)$ **then**
  ${}_k\mathcal{V}_N = \emptyset$;
  $N = 0$;
  k be increased by 1;
end
end
return ${}_kf_N^{(N)} = \sum_{\kappa=1}^{k-1}\sum_{n=1}^{K} {}_\kappa\alpha_n^{(K)}\, {}_\kappa d_n + \sum_{n=1}^{N} {}_k\alpha_n^{(N)}\, {}_kd_n$;

Algorithm 4: Pseudo-code for the iterated regularized orthogonal functional matching pursuit (ROFMP) algorithm.

Proof. First of all, note that the second equality in (4.36) holds due to simple orthogonality reasons. We show that (4.38) holds when using (4.36) and (4.37) via an induction over the iteration N. Let N equal 1. Then we have

$$\beta_1^{(1)}(d) = \frac{\langle T\neg d, T\neg d_1\rangle_{\mathbb{R}^\ell}}{\|T\neg d_1\|^2_{\mathbb{R}^\ell}}.$$

From Schmidt's orthogonalization scheme, we know that it holds

$$\mathcal{P}_{\mathcal{V}_1^\perp} T\neg d = T\neg d - \frac{\langle T\neg d, T\neg d_1\rangle_{\mathbb{R}^\ell}}{\|T\neg d_1\|^2_{\mathbb{R}^\ell}} T\neg d_1 = T\neg d - \beta_1^{(1)}(d) T\neg d_1.$$

Now, we consider the case $N+1$. We have to show that it holds

$$\mathcal{P}_{\mathcal{V}_{N+1}^\perp} T\neg d = T\neg d - \sum_{n=1}^{N+1} \beta_n^{(N+1)}(d) T\neg d_n$$

with

$$\beta_{N+1}^{(N+1)}(d) \coloneqq \frac{\left\langle \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d_{N+1} \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d_{N+1} \right\|^2_{\mathbb{R}^\ell}}$$

and

$$\beta_n^{(N+1)}(d) \coloneqq \beta_n^{(N)}(d) - \beta_{N+1}^{(N+1)}(d)\beta_n^{(N)}(d_{N+1}).$$

The induction hypothesis is

$$\mathcal{P}_{\mathcal{V}_N^\perp} T\neg d = T\neg d - \sum_{n=1}^{N} \beta_n^{(N)}(d) T\neg d_n.$$

Note that, by construction, we have

$$T\neg d_{N+1} \in \mathcal{V}_{N+1}$$

and, thus,

$$\mathcal{P}_{\mathcal{V}_N^\perp} T\neg d_{N+1} = T\neg d_{N+1} - \mathcal{P}_{\mathcal{V}_N} T\neg d_{N+1} \in \mathcal{V}_{N+1}$$

due to (4.5). Further, it also holds that

$$\mathcal{P}_{\mathcal{V}_{N+1}^\perp} T\neg d \in \mathcal{V}_{N+1}^\perp$$

for all $d \in \mathcal{D}$. Hence, we have

$$\mathcal{P}_{\mathcal{V}_N^\perp} T\neg d_{N+1} \perp \mathcal{P}_{\mathcal{V}_{N+1}^\perp} T\neg d$$

and, moreover, it holds

$$\mathcal{P}_{\mathcal{V}_{N+1}^{\perp}} T\neg d = \mathcal{P}_{\mathcal{V}_N^{\perp}} T\neg d - \frac{\left\langle \mathcal{P}_{\mathcal{V}_N^{\perp}} T\neg d, \mathcal{P}_{\mathcal{V}_N^{\perp}} T\neg d_{N+1} \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_N^{\perp}} T\neg d_{N+1} \right\|_{\mathbb{R}^\ell}^2} \mathcal{P}_{\mathcal{V}_N^{\perp}} T\neg d_{N+1},$$

again, due to (4.5). Thus, we obtain

$$\begin{aligned}
&\mathcal{P}_{\mathcal{V}_{N+1}^{\perp}} T\neg d \\
&\quad = \mathcal{P}_{\mathcal{V}_N^{\perp}} T\neg d - \frac{\left\langle \mathcal{P}_{\mathcal{V}_N^{\perp}} T\neg d, \mathcal{P}_{\mathcal{V}_N^{\perp}} T\neg d_{N+1} \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_N^{\perp}} T\neg d_{N+1} \right\|_{\mathbb{R}^\ell}^2} \mathcal{P}_{\mathcal{V}_N^{\perp}} T\neg d_{N+1} \\
&\quad = T\neg d - \sum_{n=1}^{N} \beta_n^{(N)}(d) T\neg d_n - \beta_{N+1}^{(N+1)}(d) \left(T\neg d_{N+1} - \sum_{n=1}^{N} \beta_n^{(N)}(d_{N+1}) T\neg d_n \right) \\
&\quad = T\neg d - \sum_{n=1}^{N} \left(\beta_n^{(N)}(d) - \beta_{N+1}^{(N+1)}(d) \beta_n^{(N)}(d_{N+1}) \right) T\neg d_n - \beta_{N+1}^{(N+1)}(d) T\neg d_{N+1} \\
&\quad = T\neg d - \sum_{n=1}^{N} \beta_n^{(N+1)}(d) T\neg d_n - \beta_{N+1}^{(N+1)}(d) T\neg d_{N+1} \\
&\quad = T\neg d - \sum_{n=1}^{N+1} \beta_n^{(N+1)}(d) T\neg d_n.
\end{aligned}$$

□

Next, we give an explicit formulation of the projection coefficients $\beta_n^{(N)}(d)$.

Theorem 4.4.3. *For any dictionary element $d \in \mathcal{D}$, the projection coefficients $\beta_n^{(N)}(d)$, $n = 1, ..., N,\ N \in \mathbb{N}$, of the N-th iteration of the ROFMP algorithm are given by*

$$\begin{aligned}
\beta_n^{(N)}(d) &= \beta_n^{(n)}(d) - \sum_{j=n+1}^{N} \beta_j^{(j)}(d) \beta_n^{(j-1)}(d_j), \qquad n = 1, ..., N, \\
&= \frac{\left\langle \mathcal{P}_{\mathcal{V}_{n-1}^{\perp}} T\neg d, \mathcal{P}_{\mathcal{V}_{n-1}^{\perp}} T\neg d_n \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_{n-1}^{\perp}} T\neg d_n \right\|_{\mathbb{R}^\ell}^2} - \sum_{j=n+1}^{N} \frac{\left\langle \mathcal{P}_{\mathcal{V}_{j-1}^{\perp}} T\neg d, \mathcal{P}_{\mathcal{V}_{j-1}^{\perp}} T\neg d_j \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_{j-1}^{\perp}} T\neg d_j \right\|_{\mathbb{R}^\ell}^2} \beta_n^{(j-1)}(d_j) \\
&= \frac{\left\langle T\neg d, \mathcal{P}_{\mathcal{V}_{n-1}^{\perp}} T\neg d_n \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_{n-1}^{\perp}} T\neg d_n \right\|_{\mathbb{R}^\ell}^2} - \sum_{j=n+1}^{N} \frac{\left\langle T\neg d, \mathcal{P}_{\mathcal{V}_{j-1}^{\perp}} T\neg d_j \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_{j-1}^{\perp}} T\neg d_j \right\|_{\mathbb{R}^\ell}^2} \beta_n^{(j-1)}(d_j)
\end{aligned}$$

Proof. Again, we consider this via induction over the iteration N. For $N = 1$, we have

$$\beta_1^{(1)}(d) = \beta_1^{(1)}(d) - 0.$$

We consider the iteration $N + 1$. Let $n \in \mathbb{N}$ with $n < N + 1$. From the previous theorem, we know that

$$\beta_n^{(N+1)}(d) = \beta_n^{(N)}(d) - \beta_{N+1}^{(N+1)}(d)\beta_n^{(N)}(d_{N+1}).$$

With the induction hypothesis

$$\beta_n^{(N)}(d) = \beta_n^{(n)}(d) - \sum_{j=n+1}^{N} \beta_j^{(j)}(d)\beta_n^{(j-1)}(d_j)$$

for all $n = 1, ..., N - 1$ with $N \in \mathbb{N}$, we immediately obtain

$$\begin{aligned}\beta_n^{(N+1)}(d) &= \beta_n^{(n)}(d) - \sum_{j=n+1}^{N} \beta_j^{(j)}(d)\beta_n^{(j-1)}(d_j) - \beta_{N+1}^{(N+1)}(d)\beta_n^{(N)}(d_{N+1}) \\ &= \beta_n^{(n)}(d) - \sum_{j=n+1}^{N+1} \beta_j^{(j)}(d)\beta_n^{(j-1)}(d_j).\end{aligned}$$

□

A summary of theoretical results For the ROFMP algorithm, there exist not as many theoretical results as for the RFMP algorithm. The few existing ones are summarized next, confer also with Chapter 8.
In the unregularized case, the residuals of the so-called orthogonal functional matching pursuit (OFMP) algorithm vanish at a certain point. In the regularized case (i. e. for the (iterated) ROFMP algorithm), the convergence of the sequence of functionals (4.30) can still be shown. However, the residuals stagnate at a certain point in the ROFMP algorithm. Thus, the iterated ROFMP algorithm is much more interesting also with respect to theoretical results. For this variant, the convergence of the approximation can be shown under certain technical assumptions. However, the actual limit is unknown up to now.

Realization in practice An implementation of the iterated ROFMP algorithm is naturally a bit more complex than an implementation of the RFMP algorithm. However, in Telschow (2014), there is an overview of an efficient implementation of the algorithm. In this implementation, certain terms (e. g. the values $T\neg d$ and $\langle d, \tilde{d}\rangle_{\mathcal{H}_2(\Omega)}$ for dictionary elements d and $\tilde{d}$) are computed before the iteration process starts. This is similar as with the RFMP algorithm. Further, this implementation has some 'preprocessing' in each iteration to compute the terms (4.32) and (4.34) more efficiently, in particular, with respect to the terms $b_n^{(N)}(d),\ d \in \mathcal{D},\ n = 1, ..., N,\ N \in \mathbb{N}$. We give an overview of an implementation in Appendix B as well.

Important to note is that the term (4.34) is not well-defined for functions from the span of previously chosen dictionary elements. Thus, such functions should be avoided when evaluating the quotient. If a finite dictionary is used, this can be done manually by checking whether $\mathcal{P}_{\mathcal{V}_N^\perp} T \neg d$ is numerically zero.

4.5. Notes on further research

A third flavour: the (regularized) weak functional matching pursuit ((R)WFMP) algorithm A second improvement of the RFMP algorithm is the RWFMP algorithm which was developed by Kontak (2018). This algorithm relies on the weak matching pursuit (Temlyakov, 2000). The general idea is also mentioned in Mallat and Zhang (1993). The problem it addresses is the runtime of the RFMP algorithm. In sensible experiments, a finite a-priori manually chosen dictionary needs to be sufficiently large to increase the probability that it contains dictionary elements that enable a good approximation. However, in each iteration of the RFMP algorithm, the quotient (4.25) needs to be evaluated for each dictionary element. This accounts for the biggest part of the runtime of the iteration process (that means, preprocessing is excluded) of the RFMP algorithm. The RWFMP algorithm aims to determine a dictionary element which only yields a value 'near' the maximum of (4.25). The description 'near' is measured with a so-called weakness parameter. First experiments show that the RWFMP significantly improves the runtime of the RFMP while maintaining a similarly low approximation error (see Kontak and Michel, 2019).

Task 1: determining the dictionary As we already mentioned at several points throughout this chapter, the dictionary obviously plays an important role with an IPMP algorithm: each of them can choose a best basis only among the dictionary. Thus, the obtained approximation can only be as good as the dictionary. In previous works, the dictionary had to be finite due to practical reasons. In Part II of this thesis, we develop learning algorithms which are able to use an infinite dictionary.

Task 2: suitable termination criteria An open question for the IPMP algorithms (i. e. the RFMP, ROFMP and the RWFMP) is what suitable termination criteria can be. In the same line of thought, the question of the size of K in the iterated ROFMP algorithm is open. With respect to the termination criteria, experiments up to now use a maximal number of iteration, a minimal size of the coefficient $|\alpha_{N+1}|$ or a minimal size of $\|R^{N+1}\|$. These criteria work well in practice, but are mostly set manually, except if we can relate the minimal allowed size of $\|R^{N+1}\|$ to a noise level. In the future it would be desirable if these criteria could be investigated as to how they influence the result. This may also lead to a unified strategy for setting the termination criteria in future work. As the IPMP algorithms solve –

in a certain sense – an optimization problem, it might also be possible to find such a most suitable termination criterion similar to the well-established termination criteria of optimization algorithms. In particular, it might be possible to determine which values that are naturally inherent in the algorithms (e. g. the Tikhonov functional, the relative data error, the $\mathcal{H}_2(\Omega)$-norm of the approximation) are most influential. Then the iteration process could be ended if this value only changes slightly anymore.

Task 3: parameter choice strategies As usual in inverse problems, the question for a suitable parameter choice rule is also an open question with respect to the IPMP algorithms. A first survey has shown that only some well-known methods work well with the R(O)FMP (see Gutting et al., 2017). However, all of these rules for determining a specific value $\lambda(\delta, y^\delta)$ demand that the algorithms are run with several test values. Dependent on the size of the experiment and the specific IPMP algorithm, this is very expensive with respect to time. Thus, it would be desirable if a parameter choice rule that automatically determines an optimal value $\lambda(\delta, y^\delta)$ specifically for the IPMP algorithms can be developed.
For example, we could aim to not only determine the next basis element d_{N+1} in an iteration N of an IPMP algorithm but also an optimal value of the regularization parameter. If we still adhere to the procedure of an IPMP algorithm, we have to define a model of the parameter choice strategy. This model should still take the noise level δ and the perturbed data into consideration. Furthermore, it may now also dependent on the dictionary elements. For the RFMP algorithm, a starting point of such an investigation may be to use

$$\lambda\left(\delta, y^\delta, d\right) = \frac{\|f_N\|_{\mathcal{H}_2(\Omega)}}{\|d\|_{\mathcal{H}_2(\Omega)}} 10^{-100\delta} \left\|y^\delta\right\|_{\mathbb{R}^\ell}.$$

This approach still gives a relative value with respect to the noise and the data. Furthermore, it relates a prospective basis element with the current approximation. If the latter one is rather smooth, it allows to choose more local dictionary elements in the next steps. If the approximation is not that smooth, it penalizes the choice of such functions. Hence, such a strategy would adjust itself along the iteration process.

Part II.

A learning approach for spherical inverse problems

5. An introduction to learning

In the first part of this thesis, we outlined the basic mathematical and geophysical results that are necessary for the task of learning a dictionary for the downward continuation of satellite data. Our novel algorithm will be presented in this second part.
We start our explanation with a wider look on machine learning. However, the author admits that, nowadays, the term 'learning' is, in general, widely used and well-understood due to the recent successes of, in particular, deep learning methods. Therefore, such an introduction will be incomplete by nature. Nonetheless, as we will develop a learning algorithm in the following chapters, it is appropriate to give a broader view on learning in this thesis as well. At least, this shall shed some light on where our algorithm is located in the literature.

5.1. A bit about machine learning

We consider different types of learning that can be found in natural as well as artificial intelligence. The explanations with respect to machine learning are based on Cucker and Smale (2002); Ghahramani (2004); Kaelbling et al. (1996); Poggio and Shelton (1999); Poggio and Smale (2003); Russell and Norvig (2010); Sutton and Barto (2018).
We start our considerations at a very general point of view. As an extension to Russell and Norvig (2010), we say:

learning occurs in the absence of skills.

If we lack the knowledge or the ability that is needed to do a task, we are forced to learn what we are missing. For example, some of the "intelligent systems" learn to brake if an obstacle appears, do a facial recognition or collect trash while maintaining enough battery power to get back to the charging station. In nature, learning is a life-saving ability of all creatures. Though it is said to be unable (or unwilling) to learn, even a cat acquires certain abilities such as hunting mice when growing up.
Dependent on the given information as well as what skills we are missing, learning takes place in different approaches. For artificial intelligence, these are usually separated into techniques of unsupervised and supervised learning. A particular case of the latter one is reinforcement learning. Naturally, they all have a (more or less complicated) mathematical foundation. However, in the absence of skills, we are mostly unable to derive a mathematical model analytically. Hence,

learning always includes – at least some kind of – training data. Next, we shortly explain the different learning approaches.
At first, let us consider the cat. A well-known idiosyncrasy of any feline is that it regularly patrols its realm to ensure that everything is in order. Thus, as a first step – before actually deciding whether there is danger or not – the cat has to compare a memorized state to the current state of its environment. This comparison allows a categorization in normal and not normal. Similarly, an intelligent system in a car has to check the frontal environment. It categorizes the road ahead either as clear or with an obstacle. In both cases, the system and the cat do not need an external feedback. However, they are only able to sort the input into categorizations or, in other words, they find patterns or relationships in the input. In particular, they do not evaluate them. The recognition of patterns is a classical task for unsupervised learning where the 'agent' (or learner) only works with single input. Mathematically, unsupervised learning is heavily indebted to statistical modelling. A class with practical relevance are algorithms of the k-means-type for unsupervised learning.
We easily see that the questions behind many real-world problem demand different kind of answers than can be provided with unsupervised learning. For instance, let us consider typical yes-no-questions: if our cat lives with its brother, it has experienced that meeting another cat provides no danger. Now picture the cat on its daily patrol in the backyard and the neighbour's cat is visiting for the first time. In contrast to its past with its brother, this one may be a hurtful encounter. The cat receives that it projected the wrong decision and, thus, has to re-evaluate (i. e. learn) meeting another one of its kind.
Also an intelligent system is able to recognize more than just patterns if this is necessary. For instance, it is able to learn whether a customer is in a specific target group of age or whether a particular person is entering a mall. Hence, if the agent trains with pairs of input and output (i. e. correct evaluations of related inputs), it is able to learn to project the output of unknown input data.
In these cases, we speak of supervised learning approaches. Such methods enable to learn from examples (i. e. by training instead of programming) how to correctly evaluate a problem. Mathematically, the input-output pairs enable to model an approximation or interpolation problem. Its solution then serves as a generalization of the training data and allows to project the output of unknown input. Naturally, the question arises how well this projection is. Usually, this is modelled via a loss or error function or a – more general – quality measure. The optimization of this measure yields the desired function. Problems that occur when computing a projection are classical ones: is the approximation problem well-posed? Can the projection be represented by only a few basis functions (i. e. is the projection sparse)? How well is the bias-variance problem balanced? The latter problem is also known as the 'trade-off between the curse of dimensionality and the blessing of the smoothness' (see e. g. Poggio and Smale, 2003, Section 4). In other words, we have to balance the number of examples to the size of the function space we assume the projection is from (also called hypothesis space in

machine learning). Hence, this problem is closely connected to over- and underfitting problems in practice.
However, if the difficulty of the task increases, a more flexible feedback appears to be a better learning setup. For our cat, this is the case when it learns how to catch a mouse. Of course, the cat receives the previously mentioned feedback of success or failure. Looking closely, it perceives much more information from its environment: was the hiding spot well chosen? Was the cat silent enough when creeping up to the mouse? In which direction did the wind blow? How (else) could the mouse get an early start on the cat's attack? Was the cat nowhere near the mouse or was it a last second flight from the cat's claws? Taking these answers into account as well, the cat is able to build a much more distinguished picture of its last attempt and, thus, naturally learns more easily to become a better hunter. Also some intelligent systems have to learn complex tasks and are better off when taking into account as much environmental information as possible. For instance, let us consider a robot that collects trash and runs on a rechargeable battery. Naturally, every now and then, it must decide whether to keep cleaning or to withdraw to the charging station. This decision may be based solely on the battery status. Moreover, it could also depend on the robot's distance to the charging station or its experiences of where it can expect to collect a huge amount of trash.
The learning approach that takes into account a distinguished feedback (also called reward) and decides for one of diverse possible actions is modelled by reinforcement learning techniques. In this particular learning situation, the tradeoff between exploration and exploitation needs to be taken into consideration as well. This tradeoff summarizes that, for a maximal long-time success, it is not safe per se to always decide for the maximal success in the short-term.

5.2. The task of dictionary learning

As stated in Section 4.5, an open task with respect to the IPMP algorithms is the choice of the dictionary. Here we aim to learn such a dictionary. This task, as easy as it is said, includes two aspects: on the one hand, we want to know which exact dictionary should be used for a particular inverse problem; on the other hand, we want to define a routine that determines this dictionary automatically. In this sense, we lack the knowledge (in theory and practice) what a "well-working" dictionary is and we lack the ability to reasonably determine one. Or in other words: we need to learn what trial functions should be inserted in a dictionary, why they should be picked and how they can be determined. Naturally, the answers to these questions are tightly connected to each other.
Before we develop our results, we give a brief summary of dictionary learning so far. This section is based on Aharon et al. (2006); Bruckstein et al. (2009); Rubinstein et al. (2010). Note that these are mostly surveys up to the state-of-the-art. Thus, the interested reader is also referred to the references therein.

First of all, the underlying problems of previous dictionary learning approaches are mostly approximation or interpolation ones just like we described them in Section 4.1. In particular, for a signal given via discrete values $y \in \mathbb{R}^\ell$, a representation in dictionary elements is sought. However, this representation should be "sparse". Here, sparsity means that only a few dictionary elements with essential coefficients are combined in this representation (or in other words, most dictionary elements have nearly vanishing coefficients in such a representation). Dictionary learning then aims to seek a set of trial functions which provides a sparse representation for at least all training data.
Implicitly, the determination of such a set includes that we need to be able to find a sparse representation of a signal. Unfortunately, this is proven to be an NP-hard challenge (see e. g. Bruckstein et al., 2009; Garey and Johnson, 2009).
Nonetheless, this approach was a fruitful research area which was also coined "SparseLand" (Bruckstein et al., 2009). Mathematically speaking, finding a representation of a signal y can be modelled as an underdetermined system of linear equations. This formulates as

$$y^\delta = D\alpha \qquad \text{with} \qquad D = \left(I \neg d_j\right)_{j=1,\dots,M} \qquad \text{and} \qquad \alpha = (\alpha_1, \dots, \alpha_M)\,,$$

where we use the same notation as in Chapter 4 and, in particular, dictionary elements $d_j \in \mathcal{D}$ for $j = 1, \dots, M = |\mathcal{D}|$. Then finding the sparsest representation is done by minimizing a sparsity measure (such as the l^0 measure) subject to the underdetermined system of linear equations:

$$\|\alpha\|_{l^0} \to \min! \qquad \text{subject to} \qquad y^\delta = D\alpha$$

with

$$\|\cdot\|_{l^0} := |\{i \mid \alpha_i \neq 0\}|.$$

However, obviously it depends also on the dictionary how sparse a solution can be. Therefore, in dictionary learning approaches, also the elements of $\mathcal{D}$ are variable. The most natural set that is able to provide sparsity is probably an overcomplete dictionary. Historically, this lead to an increase of specifically constructed trial functions in order to find the most suitable basis for a given signal. Note that we also introduced several different types of trial functions in the first part of this thesis. Such tailored basis functions usually enable an efficient implementation due to an analytic formulation of the functions. Further, they provide a certain localization and a multiresolution. However, the approach to determine basis functions via a sophisticated mathematical modelling bears its difficulties when it comes to adapting them to the signal as well as to higher dimensions.
Therefore, at the end of the 1990s, the literature saw a shift from constructed to trained dictionary elements. Though the efficiency of an analytic expression of the basis functions is lost in most cases, a trained dictionary appears to be more adaptive to the signal than a particularly modelled basis. In general, the shift to

training a dictionary from examples naturally enables a decoupling of determining the dictionary and the representation of the signal. Mathematically, this can be formulated as a doubled minimization problem

$$\arg\min_{\mathcal{D},\alpha\in\mathbb{R}^M} \left\| y^\delta - D\alpha \right\|_{\mathbb{R}^\ell}^2 \qquad \text{subject to} \qquad \|\alpha\|_{l^0} < A_0,\ A_0 \in \mathbb{N}.$$

In practice, this mostly leads to alternating algorithms: first a dictionary is determined, then a sparse representation of the signal in this dictionary is computed. In Aharon et al. (2006), several criteria for dictionary learning approaches are summarized. These criteria are:

- flexibility
  The method should decouple the dictionary design and the data coding and, in this way, allow the combination of the learning algorithm with any pursuit algorithm.

- simplicity
  It should be easy to explain the method. In particular, this is fulfilled if the method implements some natural generalization of the k-means algorithm.

- efficiency
  The method should be numerically efficient and converge fast. However, the authors already admit that alternating techniques have a limited efficiency by nature.

- well-defined objective
  There must exist a well-defined objective function that measures the success of the method mathematically. Note that this corresponds to the mentioned quality measure in machine learning.

Further, Prünte (2008) demands that also the given data as well as the dictionary structure should be included in the learning process. Note that the training itself is data-based by nature. However, also in this respect, some current techniques vary. Recent approaches consider only a partial view on the training data at a time and call this method online dictionary learning.
Moreover, in contemporary approaches, different version of "parametrized dictionaries" are developed. In our context, the most interesting approach of these tries to combine trained and analytic dictionaries. The suggestion in this approach is to find a representation $D = BA$ for an analytic dictionary B and a sparse matrix A.

6. Towards learning dictionaries

In this chapter, we begin our description of the novel learning algorithms for the IPMP algorithms. As we put it before, we have to explain which dictionary elements we insert into the learnt dictionary, why we choose exactly those and how we determine them. In the first section, we develop the idea of our learning algorithms. We base it on previous works on dictionary learning and our unique situation of dealing with spherical ill-posed inverse problems. Then we elaborate theoretical aspects that shall answer the question why we choose certain trial functions as learnt dictionary elements. The question how we learn these dictionary elements will be answered in the next chapter. Our novel algorithms automatically also answers the question which particular dictionary elements shall be chosen.

6.1. From the status-quo to the particular situation

Naturally, dictionary learning is characterized as a supervised learning task as we do not want to only find "similar" dictionaries but "better" ones. In that respect, we have to evaluate the learnt dictionaries at some point for which we need outputs for comparison.
One of the major advantages of the IPMP algorithms are their ability to combine established trial functions in order to determine a better approximation than traditional methods. Thus, we do not aim to determine the most suitable type of trial function for a signal as was done in the harmonic analysis approaches of dictionary learning: we want to work with diverse bases. Further, we also do not want to modify the given trial functions as was usually done in purely trained dictionaries.
Hence, to some extent, we pursue a learning method that combines training with analytical elements similar as some of the latest approaches mentioned previously. In particular, we aim to learn a best basis from a range of well-known analytical trial functions. That means, we pursue a set of trial functions that is most suitable for the approximation of a given signal. How suitable they are shall be measured mathematically. Further, this set shall still contain the advantages of analytical basis functions like efficiency, localization and multiresolution.
The difference in our situation, however, lies in the

- operator in use
  We have to work with operators related to spherical ill-posed inverse problems (not only the identity). Further, we aim to obtain a dictionary and, thus, an approximation consisting of the trial functions themselves and not their discretization.

- measure to be minimized
  The IPMP algorithms are equipped with an $\mathcal{H}_2(\Omega)$-penalty term. In their development, it was discovered that such (or similar) measures yield a far better result for spherical ill-posed inverse problems than measures that are purely aiming on sparsity.

- training data
  The problem is that we do not have classic training data in all the cases of spherical ill-posed inverse problems in which we are lacking a dictionary. If we consider, for example, the downward continuation of GRACE satellite data, we are able to work with such classical sets of data. The GRACE mission provided about 10 years of monthly data of the gravitational potential. The on-going GRACE-FO mission is about to continue these measurements. In this case, we could learn a dictionary "for GRACE" from several months and use it for approximating the signal measured in a different month. That means, in this case, we have classical training data. However, if we consider, for example, the gravitational potential of the EGM2008, we do not have several data but only a one-time measurement. Nonetheless, also for the approximation from these values, we only have rule-of-thumb dictionaries up to now.

Therefore, we see that we need to develop a very specific strategy which is tailored for the IPMP algorithms. Nonetheless, at some points, we will be able to build some bridges to certain aspects of (dictionary) learning as well.
We start the development of such a dictionary learning approach by considering the criteria given by Aharon et al. (2006) and re-interpret them for our case:

- flexibility
  Flexibility can and, therefore, should only be ensured in the sense that the fundamental idea should be applicable for all IPMP algorithms.

- simplicity
  As an ill-posed inverse problem poses a different challenge than a purely least-squares optimization, we refrain from the idea of the k-means algorithm, but rather propose the respective IPMP algorithm as the basic algorithm. Note that, in this way, we incorporate as much information as we have in the learning process.

- efficiency
  Of course, the learning algorithm should be efficient such that it can be used in practice. With respect to the simplicity, we propose that our learning algorithm is said to be efficient if computing the learnt dictionary and applying it in an IPMP algorithm does not take longer (in comparable CPU-time) than running only the respective IPMP algorithm with a rule-of-thumb-like chosen dictionary. If we learn a dictionary for and from one input data (like the EGM2008), we can reasonably define the efficiency of our learning algorithm in this way. If we use a set of training data (like it is possible with the monthly GRACE data), however, we have no reasonable comparison time. Therefore, we assume that, if the efficiency of the learning algorithm is acceptable for a single input, the efficiency for a set of training data is, too.

- well-defined objective
  Last but not least, we have to consider a mathematical justification of our learning algorithm as well. That means, our algorithm shall have a well-defined objective that explains why we pick a certain dictionary element. In other words, we have to formulate our goal, i. e. learn a "well-working" dictionary mathematically, and investigate its properties. Naturally, this well-defined objective needs to be well-connected with the "what" and "how" to pick as well.

On the way to develop such a well-defined objective for our learning algorithm, we first make some observations. First of all, we will only establish such a goal for learning a dictionary for the RFMP algorithm. This is due to the fact that we only have valid theoretical results for this one. For the ROFMP algorithm, unfortunately, we currently still lack proper convergence proofs. However, we assume that a well-defined learning measure should be related to such results. Thus, we will only transfer the learning idea to the orthogonal algorithm.
For the RFMP algorithm, we considered the objective

$$(\alpha_{N+1}, d_{N+1}) := \underset{(\alpha,d) \in \mathbb{R}\times\mathcal{D}}{\arg\min} \mathcal{J}\left(f_N + \alpha d; T_\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right)$$

in the N-th iteration, see (4.22). Ideally, the dictionary $\mathcal{D}$ would be (over-) complete and, thus, infinite. In this case, we write $\mathcal{D}^{\mathrm{Inf}}$ instead and consider

$$(\alpha_{N+1}, d_{N+1}) := \underset{(\alpha,d) \in \mathbb{R}\times\mathcal{D}^{\mathrm{Inf}}}{\arg\min} \mathcal{J}\left(f_N + \alpha d; T_\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right). \tag{6.1}$$

Note that, as we pointed out before, in particular in the case of an infinite dictionary, the existence of this minimum is in question and we better consider the infimum. Hence, in a perfect world, this approach would yield the solution

$$\inf_{f_\infty \in \overline{\operatorname{span}\mathcal{D}^{\mathrm{Inf}}}} \mathcal{J}\left(f_\infty; T_\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) \tag{6.2}$$

for $N \to \infty$ and

$$f_\infty = \sum_{n=1}^{\infty} \alpha_n d_n.$$

Here, a perfect world stands for the case that we can assure that the sum of greedy steps also yields the overall optimal solution. However, problems like this are of the travelling salesman type. Thus, they are known to be NP-hard (see e. g. Garey and Johnson, 2009, p. 114). That means, we cannot be sure that we obtain the optimal approximation by greedily choosing dictionary elements. Apart from that, we also have to admit that, in practice, the nearest to the optimum we can get is to compute the optimal best-N-term approximation. The smallest problem at this stage may actually be to compute the next basis element d_{N+1} from infinitely many trial functions. Nonetheless, we should be aware that this feature might be a very costly one. With these things in mind, the best we can hope to determine is

$$\inf_{f_N \in \operatorname{span} \mathcal{D}^{\mathrm{Inf}}} \mathcal{J}\left(f_N; T\urcorner, \lambda\left(\delta, y^\delta\right), y^\delta\right) \tag{6.3}$$

with

$$f_N = \sum_{n=1}^{N} \alpha_n d_n \tag{6.4}$$

for the iteration N at termination. In experiments with a finite $\mathcal{D}$, the approximation obtained from the RFMP algorithm proved to be satisfyingly close to the desired f_N. Further, the aspect of learning the dictionary is contained in this formulation as f_N is chosen from the span of infinitely many elements. Hence, we declare (6.3) as the main objective of our learning algorithm for the RFMP algorithm. Note that this idea can be easily transferred to learning a dictionary for the ROFMP algorithm as we only need to exchange the functional $\mathcal{J}$ with $\mathcal{J}_O$. Summarized, our novel learning methods shall extend the IPMP algorithms to an infinite dictionary. Hence, we call them learning inverse problem matching pursuit (LIPMP) algorithms.

Let us consider the superset of the basis elements of (6.4) because this enables us to retrieve the approximation f_N from the learnt dictionary. In other words, we know exactly which N dictionary elements should be at least in the learnt dictionary. More generally speaking, this allows us to define an upper bound $D \geq N$ for a learnt dictionary $\mathcal{D}$ where $\mathcal{D}$ should still be a subset of $\mathcal{D}^{\mathrm{Inf}}$. All in all, we can reformulate (6.3) to

$$\inf_{\substack{\mathcal{D} \subset \mathcal{D}^{\mathrm{Inf}} \\ |\mathcal{D}| \leq D}} \min_{f_N \in \operatorname{span} \mathcal{D}} \mathcal{J}\left(f_N; T\urcorner, \lambda\left(\delta, y^\delta\right), y^\delta\right). \tag{6.5}$$

Hence, we tailored a dictionary learning ansatz closely to the RFMP algorithm and reformulated the natural objective to a doubled minimization problem for a

predefined dictionary size D. Obviously, such a problem represents the task of finding a dictionary and an approximation simultaneously. As mentioned before, such a task is common in the field of dictionary learning.
The question arises how we learn a dictionary from this objective. In practice, (6.3) is exchanged by (6.1) in an iteration N. Then we have defined what we pick in each iteration and can straight-forwardly define our learnt dictionary as a superset of the basis elements of the structure book that defines (6.4).

Definition 6.1.1. *Let $\mathcal{B} := \{(\alpha_n, d_n)\}_{n=1,\dots,N}$ be the structure book, see (4.8), of an LIPMP algorithm after $N \in \mathbb{N}$ iterations with respect to perturbed data y^δ and the operator $T^\neg$ and using the regularization parameter $\lambda(\delta, y^\delta)$. For an infinite dictionary $\mathcal{D}^{\mathrm{Inf}}$, the smallest learnt dictionary $\mathcal{D}_N^*\left(f_0, T^\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) \subset \mathcal{D}^{\mathrm{Inf}}$ of the given inverse problem is defined as*

$$\mathcal{D}_N^*\left(f_0, T^\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) := \{f_0\} \cup \{d \mid \exists \alpha \in \mathbb{R} : \ (\alpha, d) \in \mathcal{B}\}.$$

In particular, this means, we obtain a sequence of smallest learnt dictionaries by the update rule

$$\mathcal{D}_{N+1}^*\left(f_0, T^\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) = \mathcal{D}_N^*\left(f_0, T^\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) \cup \{d_{N+1}\}.$$

Note that the details of how to pick these basis elements are given in Chapter 7.

6.2. Optimal, near-optimal and well-working dictionaries

As we pointed out, ideally we would like to solve (6.2). $\mathcal{D}^{\mathrm{Inf}}$ is an example of an (over-) complete dictionary in $\mathcal{H}_2(\Omega)$, this means we obtain the solution of

$$\inf_{f_\infty \in \mathcal{H}_2(\Omega)} \mathcal{J}\left(f_\infty; T^\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right), \ f_\infty = \sum_{n=1}^{\infty} \alpha_n d_n, \ d_n \in \mathcal{H}_2(\Omega), \ \alpha_n \in \mathbb{R}.$$

In the sequel, we assume that the coefficients α_n, $n \in \mathbb{N}$, are decreasingly ordered. Though the solution f_∞ is unique due to the theory of Tikhonov-Phillips regularization, we see that any (over-) complete dictionary in $\mathcal{H}_2(\Omega)$ is able to reproduce the solution. Thus, we define an optimal dictionary as follows.

Definition 6.2.1. *An optimal dictionary $\mathcal{D}$ is an (over-) complete dictionary.*

Theorem 6.2.2. *Any basis system is an optimal dictionary.*

Thus, we see that an optimal dictionary exists. However, it is anything but unique. We immediately see that any optimal dictionary must be infinite and,

thus, cannot be used in the IPMP algorithms as described in Section 4.3 and Section 4.4. Further, in general, the solution f_∞ cannot be revealed in practice because we have to terminate a respective algorithm at a certain point.
Therefore, a more practical objective is given in (6.3). Though we still have an infinite dictionary, we only seek a finite linear combination f_N as an approximation of f_∞. This is an element of a finite linear subspace of $\operatorname{span} \mathcal{D}^{\text{Inf}}$ as well as $\overline{\operatorname{span} \mathcal{D}^{\text{Inf}}}$. What effect does it have on the dictionary? We saw that solving (6.3) is equivalent to solving (6.5). This shows that the approximation f_N of the solution f_∞ can be obtained by a finite dictionary $\mathcal{D}$ as well. Hence, we propose to consider nearly-optimal dictionaries.

Definition 6.2.3. *Let f_∞ be the solution of the regularized normal equation with respect to the regularization parameter $\lambda(\delta, y^\delta)$ for perturbed data y^δ and the operator $T^\neg$. Further, let*

$$f_\infty = \sum_{n=1}^{\infty} \alpha_n d_n \tag{6.6}$$

be a representation of f_∞ in a complete dictionary $\mathcal{D}^{\text{Inf}}$ with decreasingly ordered values $|\alpha_n|$, $n \in \mathbb{N}$. A nearly-optimal N-dictionary N-$\mathcal{D}\left(T^\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right)$ is a finite dictionary such that it contains $d_1, ..., d_N$.

Obviously, for a given optimal dictionary, at least one nearly optimal dictionary exists.

Theorem 6.2.4. *The smallest nearly-optimal N-dictionary contains only the dictionary elements $d_1, ..., d_N$ from the representation* (6.6) *of f_∞.*

However, because the representation of f_N in dictionary elements is not unique, a nearly-optimal N-dictionary is also not unique.
Even though learning a nearly-optimal N-dictionary by assuming we are able to choose from infinitely many dictionary elements is a bit more practical, also this objective lacks a fundamental aspect. Determining the solution f_∞ and, thus, the respective approximation f_N is NP-hard (see e. g. Bruckstein et al., 2009; Garey and Johnson, 2009).
In practice, we can only take iteratively greedy steps instead and hope for the best. Therefore, we settle for a weaker characterization.

Definition 6.2.5. *Let f_∞ be the solution of the regularized normal equation with respect to the regularization parameter $\lambda(\delta, y^\delta)$ for perturbed data y^δ and the operator $T^\neg$. Further, let*

$$f_\infty = \sum_{n=1}^{\infty} \alpha_n d_n \tag{6.7}$$

be a representation of f_∞ in a complete dictionary $\mathcal{D}^{\text{Inf}}$ with decreasingly ordered values $|\alpha_n|$, $n \in \mathbb{N}$. At last, let $\left(\mathcal{D}_N\left(T^\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right)\right)_{N \in \mathbb{N}_0}$ be a sequence of finite dictionaries for which it holds:

(SWW1) For all $N \in \mathbb{N}_0$*, we have* $\mathcal{D}_N\left(T_\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) \subseteq \mathcal{D}_{N+1}\left(T_\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right)$.
(SWW2) For all d_i *used in* (6.7)*, we have* $d_i \in \bigcup_{n=0}^{\infty} \mathcal{D}_n\left(T_\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right)$.
Then $\left(\mathcal{D}_N\left(T_\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right)\right)_{N \in \mathbb{N}_0}$ *is called a sequence of well-working dictionaries.*

By construction, a sequence of well-working dictionaries exists: if each $\mathcal{D}_N$ is a nearly-optimal N-dictionary N-$\mathcal{D}$, then the sequence is well-working. However, the order of the sequence elements is far from unique and, thus, the sequence itself is neither. Nonetheless, we assume that for some N the dictionary $\mathcal{D}_N(T_\neg, \lambda\left(\delta, y^\delta\right), y^\delta)$ of a sequence of well-working dictionaries will be able to reconstruct a good approximation of f_∞. Therefore, learning a dictionary that is an element of a sequence of well-working dictionaries shall be the well-defined objective for our dictionary learning algorithm.

6.3. An outlook

The previous formal characterizations of desired dictionaries are a novelty. However, note that we started our considerations with the solution of the Tikhonov functional. The question arises whether this functional is the best quality measure for our task. In general, we can say it is a good starting point because the (L)RFMP algorithm converges towards its solution (confer Chapter 8). However, the author recognizes that, also in the light of Prünte (2008), an improved quality measure could probably be developed. In the future research, we could think about combining the Tikhonov functional with a total variation TV dependent on the residual and, thus, on the data. This could be done either as an additional summand

$$\inf_{f_\infty \in \overline{\operatorname{span} \mathcal{D}^{\mathrm{Inf}}}} \left[\mathcal{J}\left(f_\infty; T_\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) - \mu TV\left(y^\delta - T_\neg f_\infty\right)\right] \qquad \text{for } \mu \in \mathbb{R}^+,$$

considering it in the parameter choice rule

$$\inf_{f_\infty \in \overline{\operatorname{span} \mathcal{D}^{\mathrm{Inf}}}} \mathcal{J}\left(f_\infty; T_\neg, \lambda\left(\delta, y^\delta, TV\left(y^\delta - T_\neg f_\infty\right)\right), y^\delta\right)$$

or for a dual approach such as

$$\inf_{f_\infty \in \overline{\operatorname{span} \mathcal{D}^{\mathrm{Inf}}}} \mathcal{J}\left(f_\infty; T_\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) \quad \text{subject to} \quad TV\left(y^\delta - T_\neg f_\infty\right) \geq \Delta$$

for a given threshold $\Delta > 0$. Note that the first suggestion would probably lead to an elastic-net approach (see e. g. Daubechies et al., 2004; De Mol et al., 2009; Zou and Hastie, 2005) in the learning algorithms and, as we will see, the IPMP algorithms.

7. A learning algorithm

In the last chapters, we considered the theoretical aspects of learning a dictionary: we formulated the task of learning a dictionary and developed a theoretical criterion for a desirable dictionary. With these considerations in mind, we now look again at the matching pursuits from Section 4.3 and Section 4.4. For both the RFMP and ROFMP algorithm, we introduce a learning variant in this chapter. These **l**earning **i**nverse **p**roblem **m**atching **p**ursuit (LIPMP) algorithms have not only an approximation but also a learnt dictionary as a result. In particular, in this chapter, we develop the **l**earning **r**egularized **f**unctional **m**atching **p**ursuit (LRFMP) and the **l**earning **r**egularized **o**rthogonal **f**unctional **m**atching **p**ursuit (LROFMP) algorithm. We first introduce their main idea and routine. Then we consider certain optimization problems in detail: how are they modelled? What terms do they depend on in practice? What additional features are useful to guide the learning process? At last, we summarize the LIPMP algorithms in pseudocodes.

7.1. Idea and main structure

In Chapter 6, we describe that the objective of any previously introduced matching pursuit for inverse problems can be generalized with respect to an unknown finite best dictionary. Further, we showed how this is related to a doubled minimization problem which is a well-known starting point in the field of dictionary learning. However, the task usually handed to an IPMP algorithm is not suitable to be combined with classical dictionary learning methods as we pointed out in the introduction to this thesis (Chapter 1). Thus, the novel approach we develop next follows its own approach. Maybe in contrast to what could be expected, for learning an IPMP dictionary, we do not concentrate on the dictionary itself but on the inverse problem at hand.
We aim to solve the inverse problem simultaneously with the determination of a dictionary. One reason is that, in our setting, it is mostly too expensive with respect to time and/or storage to alternate between steps in the dictionary learning process and solving an inverse problem. Further, naturally, we cannot expect to learn a good dictionary independent of, at least, some information about the problem it should be used for.
We consider the probably most practical formulations of the dictionary learning

task from Section 6.1 again in more detail: for the LRFMP algorithm, we have

$$(\alpha_{N+1}, d_{N+1}) := \underset{(\alpha,d)\in\mathbb{R}\times\mathcal{D}^{\mathrm{Inf}}}{\arg\min} \left(\mathcal{J}\left(f_N + \alpha d; T_{\urcorner}, \lambda\left(\delta, y^{\delta}\right), y^{\delta}\right)\right) \tag{7.1}$$

and, for the LROFMP algorithm, we look at

$$\begin{aligned}\left(\alpha^{(N+1)}, d_{N+1}\right) := \underset{(\alpha,d)\in\mathbb{R}^{N+1}\times\mathcal{D}^{\mathrm{Inf}}}{\arg\min} \left(\mathcal{J}_O\left(f_N^{(N+1)} + \alpha_{N+1} d; T_{\urcorner}, \lambda\left(\delta, y^{\delta}\right), y^{\delta}\right)\right), \\ \alpha = (\alpha_1, ..., \alpha_{N+1}) \in \mathbb{R}^{N+1},\end{aligned} \tag{7.2}$$

for an infinite dictionary

$$\mathcal{D}^{\mathrm{Inf}} := [\widehat{N}]_{\mathrm{SH}} + \left[\mathbb{B}^4\right]_{\mathrm{SL}} + \left[\mathring{\mathbb{B}}\right]_{\mathrm{APK}} + \left[\mathring{\mathbb{B}}\right]_{\mathrm{APW}} \tag{7.3}$$

with $\widehat{N} \subset \mathcal{N}$ and using Definition 4.2.2. Note that we maintain the formulation with the minimum instead of using the infimum as we did in Sections 4.3 and 4.4. Further, note that these trial functions are in particular suitable for spherical inverse problems. If we compare (7.1) to (4.22) and (7.2) to (4.31), we see that the IPMP algorithms are built to solve these kinds of minimization problems in order to solve the related inverse problems – except that they are used with an a-priori chosen finite dictionary. Hence, we concentrate on modelling these minimization problems for an infinite dictionary. If we incorporate such a model into the respective IPMP algorithm, we obtain an algorithm which solves an inverse problem and is (nearly) independent of an a-priori dictionary choice. That means, we have an advanced standalone matching pursuit for inverse problems.
From the way the underlying IPMP algorithm works, the obtained algorithm naturally produces a learnt dictionary as well: in each IPMP iteration, a trial function is chosen from a finite set of dictionary elements and is added to the structure book (confer Definition 4.1.5, (4.8)). Furthermore, after termination, the approximation is a decomposition in this best basis. If we are able to choose each of these basis elements from an infinite set of dictionary elements, the structure book defines a finite best subset of these infinitely many ones – at least, for the experiment setting from which it was obtained. Hence, the basis elements in the structure book define our learnt dictionary. Then, in future runs of an IPMP algorithm, it is sufficient to use the learnt dictionary of the respective LIPMP algorithm as a dictionary.
In summary, if we are able to model an IPMP algorithm with an infinite dictionary, we automatize the selection of a best basis among infinitely many possibilities and, in this way, learn a dictionary. We make notes of these aspects.

Remark 7.1.1. Each LIPMP algorithm follows the same routine as its respective IPMP algorithm. We also say that an IPMP algorithm underlies the respective LIPMP algorithm. However, in each iteration $N \in \mathbb{N}_0$, we choose the next basis element d_{N+1} from the infinitely many dictionary elements of $\mathcal{D}^{\mathrm{Inf}}$.

A further reason for this approach is that, by construction, the learnt dictionary is tailored specifically for the underlying IPMP algorithm, for a given inverse problem and a particular Tikhonov regularization strategy. In this way, we incorporate as much information in the learning process as we have. This usually supports a better learning result and was also suggested in previous works on dictionary learning as well (see e. g. Prünte, 2008).
Next, we concentrate on (7.1) and (7.2). We have seen in Section 4.3 and Section 4.4 that these minimization problems are equivalent to the following maximization problems (confer (4.25) and (4.34), respectively). Let $N \in \mathbb{N}$ be the current iteration. For the LRFMP algorithm, we consider

$$d_{N+1} := \operatorname*{arg\,max}_{d \in \mathcal{D}^{\mathrm{Inf}}} \frac{\left(\left\langle R^N, T^\neg d \right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right) \left\langle f_N, d \right\rangle_{\mathcal{H}_2(\Omega)} \right)^2}{\left\| T^\neg d \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \left\| d \right\|_{\mathcal{H}_2(\Omega)}^2} \tag{7.4}$$

and, for the LROFMP algorithm, we have

$$d_{N+1} := \operatorname*{arg\,max}_{d \in \mathcal{D}^{\mathrm{Inf}}} \frac{\left(\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T^\neg d \right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right) \left\langle f_N^{(N)}, d - b_N^{(N)}(d) \right\rangle_{\mathcal{H}_2(\Omega)} \right)^2}{\left\| \mathcal{P}_{\mathcal{V}_N^\perp} T^\neg d \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \left\| d - b_N^{(N)}(d) \right\|_{\mathcal{H}_2(\Omega)}^2}. \tag{7.5}$$

In accordance to the use of the minimum, we formulate these definitions with the maximum. Note that, in the following development of the LROFMP algorithm, we will use the notation from the non-iterated ROFMP algorithm for better readability. Further, note that the choice of d_{N+1} is still not unique in general. Hence, we make the following definition.

Definition 7.1.2. *For the Sobolev space $\mathcal{H}_2(\Omega)$, we define the objective function of the LRFMP algorithm in the N-th iteration as*

$$\mathrm{RFMP}(d; N) := \frac{\left(\left\langle R^N, T^\neg d \right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right) \left\langle f_N, d \right\rangle_{\mathcal{H}_2(\Omega)} \right)^2}{\left\| T^\neg d \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right) \left\| d \right\|_{\mathcal{H}_2(\Omega)}^2}, \tag{7.6}$$

where $d \in \mathcal{D}^{\mathrm{Inf}}$, R^N is the current residual, f_N the current approximation (for both see Definition 4.1.5 using $T^\neg$ instead of $I^\neg$), $T^\neg$ depends on the inverse problem (see Remark 4.2.4) and $\lambda(\delta, y^\delta)$ is given by the regularization strategy. In the sequel, we use the abbreviations

$$a_1(d) := \left\langle R^N, T^\neg d \right\rangle_{\mathbb{R}^\ell}, \qquad a_2(d) := \left\langle f_N, d \right\rangle_{\mathcal{H}_2(\Omega)},$$
$$b_1(d) := \left\| T^\neg d \right\|_{\mathbb{R}^\ell}^2, \qquad \text{and} \qquad b_2(d) := \left\| d \right\|_{\mathcal{H}_2(\Omega)}^2,$$

such that

$$\mathrm{RFMP}(d;N) := \frac{\left(a_1(d) - \lambda\left(\delta, y^\delta\right) a_2(d)\right)^2}{b_1(d) + \lambda\left(\delta, y^\delta\right) b_2(d)}. \tag{7.7}$$

Theorem 7.1.3. *The minimization of $\mathcal{J}$, see* (4.22)*, in the N-th step of the LRFMP algorithm with respect to a dictionary element $d \in \mathcal{D}^{\mathrm{Inf}}$ and a real coefficient α as seen in* (7.1) *is equivalent to the maximization of* $\mathrm{RFMP}(\cdot;N)$ *with respect to d and*

$$\alpha := \frac{a_1(d^*) - \lambda\left(\delta, y^\delta\right) a_2(d^*)}{b_1(d^*) + \lambda\left(\delta, y^\delta\right) b_2(d^*)}$$

for the optimal d^*.

Proof. This was already shown in Section 4.3 □

Analogously, we obtain for the LROFMP algorithm the following result.

Definition 7.1.4. *For the Sobolev space $\mathcal{H}_2(\Omega)$, we define the objective function of the LROFMP algorithm in the N-th iteration as*

$$\mathrm{ROFMP}(d;N) := \frac{\left(\left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle f_N^{(N)}, d - b_N^{(N)}(d)\right\rangle_{\mathcal{H}_2(\Omega)}\right)^2}{\left\|\mathcal{P}_{\mathcal{V}_N^\perp} T\neg d\right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\|d - b_N^{(N)}(d)\right\|_{\mathcal{H}_2(\Omega)}^2} \tag{7.8}$$

where $d \in \mathcal{D}^{\mathrm{Inf}}$, R^N is the current residual, $f_N^{(N)}$ the current approximation (for both see Definition 4.1.6 using $T\neg$ instead of $I\neg$), $\mathcal{P}_{\mathcal{V}_N^\perp}$ is the orthogonal projection onto $\mathcal{V}_N^\perp$ (confer (4.27) *for $\mathcal{V}_N$), $b_N^{(N)}$ is given by*

$$b_N^{(N)}(d) := \sum_{n=1}^{N} \beta_n^{(N)}(d) d_n$$

with the projection coefficients $\beta_n^{(N)}(d)$, $n = 1, \ldots, N$, as given in Theorem 4.4.2 or Theorem 4.4.3, respectively, $T\neg$ depends on the inverse problem (see Remark 4.2.4) and $\lambda(\delta, y^\delta)$ is given by the regularization strategy.

Note that, also here, the projection coefficients are generally not unique. Basic functional analysis yields the following, more practical formulation of the objective function of the LROFMP algorithm.

Lemma 7.1.5. *For the objective function of the LROFMP algorithm, it holds*

$$\begin{aligned}&\mathrm{ROFMP}(d;N)\\&=\frac{\left(\left\langle R^N,\mathcal{P}_{\mathcal{V}_N^\perp}T\neg d\right\rangle_{\mathbb{R}^\ell}+\lambda\left(\delta,y^\delta\right)\left(\left\langle f_N^{(N)},b_N^{(N)}(d)\right\rangle_{\mathcal{H}_2(\Omega)}-\left\langle f_N^{(N)},d\right\rangle_{\mathcal{H}_2(\Omega)}\right)\right)^2}{\left\|\mathcal{P}_{\mathcal{V}_N^\perp}T\neg d\right\|_{\mathbb{R}^\ell}^2+\lambda\left(\delta,y^\delta\right)\left(\|d\|_{\mathcal{H}_2(\Omega)}^2-2\left\langle d,b_N^{(N)}(d)\right\rangle_{\mathcal{H}_2(\Omega)}+\left\|b_N^{(N)}(d)\right\|_{\mathcal{H}_2(\Omega)}^2\right)}\end{aligned}$$

with the same notation as in Definition 7.1.4. Analogously to Definition 7.1.2, we abbreviate

$$\begin{aligned}a_3(d)&:=\left\langle R^N,\mathcal{P}_{\mathcal{V}_N^\perp}T\neg d\right\rangle_{\mathbb{R}^\ell},&a_4(d)&:=\left\langle f_N^{(N)},b_N^{(N)}(d)\right\rangle_{\mathcal{H}_2(\Omega)},\\b_3(d)&:=\left\|\mathcal{P}_{\mathcal{V}_N^\perp}T\neg d\right\|_{\mathbb{R}^\ell}^2,&b_4(d)&:=\left\langle d,b_N^{(N)}(d)\right\rangle_{\mathcal{H}_2(\Omega)},\\b_5(d)&:=\left\|b_N^{(N)}(d)\right\|_{\mathcal{H}_2(\Omega)}^2.\end{aligned}$$

Then we obtain

$$\mathrm{ROFMP}(d;N):=\frac{\left(a_3(d)+\lambda\left(\delta,y^\delta\right)\left(a_4(d)-a_2(d)\right)\right)^2}{b_3(d)+\lambda\left(\delta,y^\delta\right)\left(b_2(d)-2b_4(d)+b_5(d)\right)}\tag{7.9}$$

with $a_2(d)$ and $b_2(d)$ as in Definition 7.1.2.

Proof. Use the linearity of the inner product and the norm, respectively, for the terms

$$\left\langle f_N^{(N)},d-b_N^{(N)}(d)\right\rangle_{\mathcal{H}_2(\Omega)}\qquad\text{and}\qquad\left\|d-b_N^{(N)}(d)\right\|_{\mathcal{H}_2(\Omega)}^2.$$

□

Theorem 7.1.6. *The minimization of $\mathcal{J}_O$, see* (4.31)*, in the N-th step of the LROFMP algorithm with respect to a dictionary element $d\in\mathcal{D}^{\mathrm{Inf}}$ and a real coefficient α as seen in* (7.2) *is equivalent to the maximization of* $\mathrm{ROFMP}(\cdot;N)$ *with respect to d and with*

$$\alpha=\frac{a_3(d^*)+\lambda\left(\delta,y^\delta\right)\left(a_4(d^*)-a_2(d^*)\right)}{b_3(d^*)+\lambda\left(\delta,y^\delta\right)\left(b_2(d^*)-2b_4(d^*)+b_5(d^*)\right)}$$

for the optimal d^*.

Proof. This has already been shown in Section 4.4. □

Now we consider the maximization of these objective functions. Though we do not a-priori know a best basis for a given problem or which trial functions should be contained in an optimal finite dictionary, the infinite dictionary $\mathcal{D}^{\mathrm{Inf}}$ form (7.3) is not even similarly unknown to us. We outline how the maximizations can be done for the particular trial function classes of $\mathcal{D}^{\mathrm{Inf}}$, i. e. for spherical harmonics,

Slepian functions as well as Abel–Poisson low and band pass filters. As these functions are very different, the maximization problems (7.4) and (7.5), respectively, cannot be treated uniformly any further for the classes. Hence, we divide the maximizations into smaller ones.

Remark 7.1.7. The maximization of either $\mathrm{RFMP}(d;N)$ or $\mathrm{ROFMP}(d;N)$ with respect to $d \in \mathcal{D}^{\mathrm{Inf}}$ is done by maximizing the respective objective function with respect to $d \in [\cdot]_\bullet \subseteq \mathcal{D}^{\mathrm{Inf}}$ for $\bullet \in \{\mathrm{SH}, \mathrm{APK}, \mathrm{APW}, \mathrm{SL}\}$ separately. In particular, we consider

$$\mathrm{RFMP}(d;N) \to \max! \qquad \text{subject to} \qquad d \in [\cdot]_\bullet \tag{7.10}$$

for the LRFMP algorithm and

$$\mathrm{ROFMP}(d;N) \to \max! \qquad \text{subject to} \qquad d \in [\cdot]_\bullet \tag{7.11}$$

for the LROFMP algorithm.

Naturally, one of the solutions of the smaller maximization problems (7.10) and (7.11), respectively, for each trial function class also solves the unified maximization problems (7.4) and (7.5), respectively. Therefore, we call any of the solutions of (7.10) or (7.11) a candidate of the objective function of the respective LIPMP algorithm. As the set of candidates is finite, we can evaluate the objective functions $\mathrm{RFMP}(\cdot;N)$ and $\mathrm{ROFMP}(\cdot;N)$, respectively, for all of the candidates, compare these values and choose the candidate with the maximal value as the next basis element d_{N+1}. That means, the set of candidates can be treated analogously to a finite dictionary in an IPMP algorithm. In this way, we can choose a basis element in the current iteration N from infinitely many dictionary elements. After that, the LIPMP algorithm follows the structure of the underlying IPMP algorithm to finish the iteration N. This includes computing the related coefficient and necessary updates (confer Section 4.3 and Section 4.4, respectively).

Remark 7.1.8. The maximization of either $\mathrm{RFMP}(d;N)$ or $\mathrm{ROFMP}(d;N)$ with respect to each trial function class under consideration yields a finite set of candidates from the infinite dictionary $\mathcal{D}^{\mathrm{Inf}}$. The next chosen basis element d_{N+1} is the candidate that has the highest value when inserted into the respective objective function.

We note two properties that are included in this approach.

(a) When computing the candidates, the algorithm considers all trial function classes the user allows. In the language of machine learning, this is a phase where the agent (i. e. the decision maker for d_{N+1}) "explores" its opportunities. Then the candidate is chosen that yields the highest value in the respective objective function. That means, the agent decides for the possible action which yields the most success or – in other words – "exploits" its opportunities. Hence, this learning approach takes care of balancing exploration and exploitation in each iteration.

(b) By exploring all trial function classes and autonomously deciding for a candidate, the LIPMP algorithm simultaneously decides which trial function classes are indeed useful for a decomposition in a best basis. Hence, in an analysis of the structure book of the LIPMP algorithm, we see a hint that a class is not necessary for a decomposition if the number of chosen functions from this class is negligible in comparison to other classes. Further, the user does not need to know beforehand which trial function classes are indeed the most suitable ones (except for a reduction of runtime).

All in all, the LIPMP algorithm follows the general structure as given in Figure 7.1. We start in the red circle with a similar initialization as in the underlying IPMP algorithm. Then we step into the iteration process. First, we consider the optimization problems for the different trial function classes. This yields a set of candidates from which we choose the best candidate as the next basis element d_{N+1}. We compute some updates (dependent on the respective IPMP algorithm; for details see Telschow (2014) and Appendix B). If at least one termination criterion is fulfilled, we stop the LIPMP algorithm and give the basis elements of the structure book as our learnt dictionary as well as the obtained approximation as our results. If no termination criterion is fulfilled yet, we step into the next iteration. Note that the termination criteria we use in practice are inherited from the IPMP algorithms.
Next, we explain how to solve the optimization problem for each trial function class in an iteration of the LIPMP algorithms.

7.2. Optimization problems in detail

In this section, we look at the optimization problems for spherical harmonics, Slepian functions and Abel-Poisson low and band pass filters in more detail. We derive as much as we can from the optimization problems for an arbitrary operator $T_\neg$. As an example for practical purposes, we consider the discretized upward continuation operator $\mathcal{T}_\neg$ (confer Theorem 2.4.7 and Remark 4.2.4) where a particular one is needed.

7.2.1. Formulation of parametrized optimization problems

In (7.10) and (7.11), respectively, we formulated optimization problems which need to be modelled for practical purposes. In general, the idea is to parametrize each trial function class via their particular characteristics. What remains can be solved with established numerical algorithms for optimization. Though it would be interesting to also have some theoretical results with respect to the optimization problems, e. g. whether each problem is convex, the complicated objective functions $\mathrm{RFMP}(\cdot; N)$ and $\mathrm{ROFMP}(\cdot; N)$, respectively, forbid any further investigation in this direction.

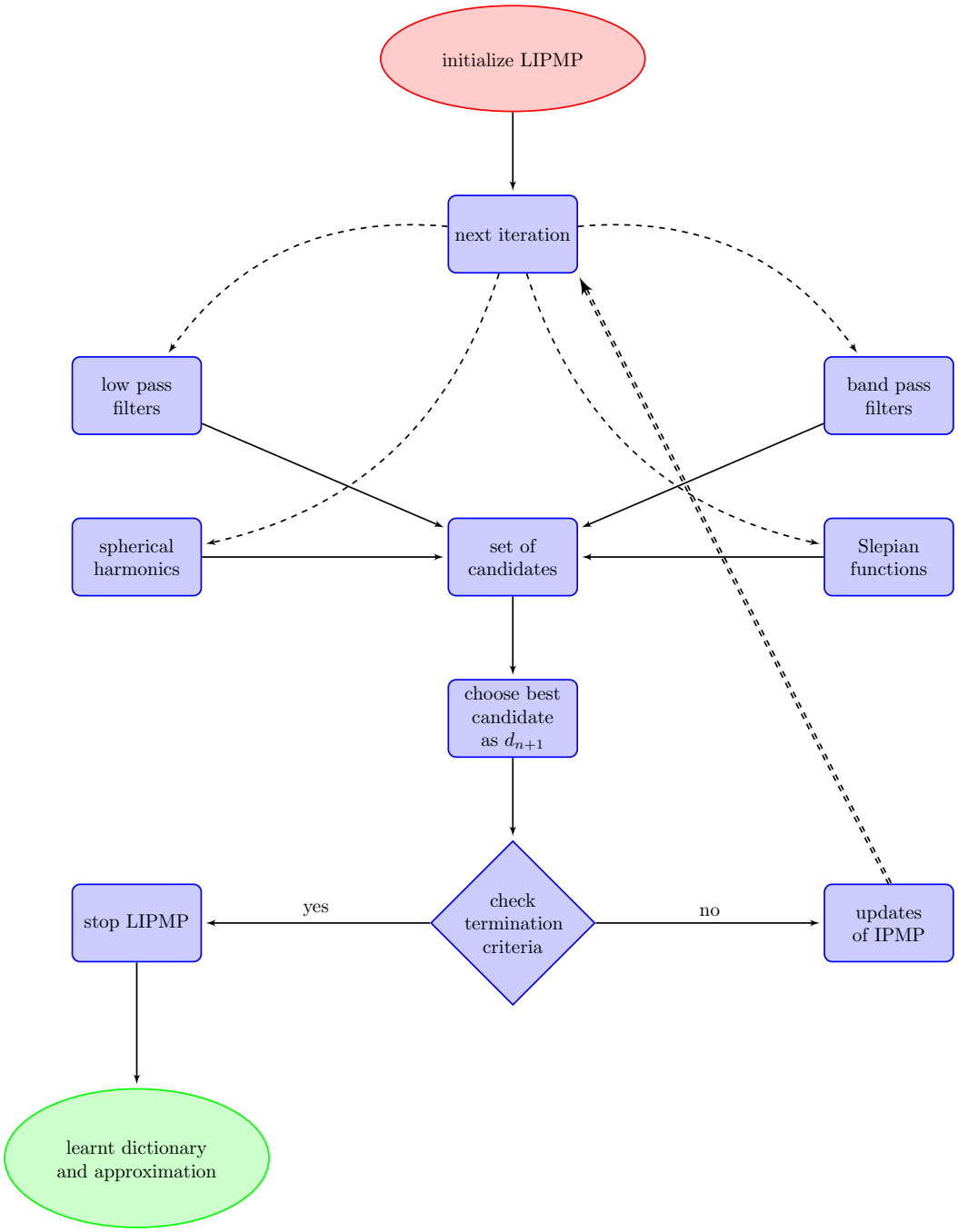


Figure 7.1.: Schematic representation of the basic idea of the LIPMP algorithms. The different types of lines are only used for an improved visualization. The algorithm starts in the red circle.

Spherical harmonics The spherical harmonics (see Definition 2.2.3) are distinguished by their degree $n \in \mathbb{N}_0$ and order $j = -n, ..., n$. The computation of high degrees (and arbitrary orders) is expensive in practice. Furthermore, in the line of thought of a matching pursuit, it is not sensible to allow the use of high degrees in an approximation: high degrees have a small amplitude and, thus, approximate local structures. As they are global functions, overall, this might have a bad influence on the approximation. In the sense of the matching pursuits, they are not the right vocabulary for local structures which is the reason why we also consider low and band pass filters in $\mathcal{D}^{\text{Inf}}$. Hence, it is sensible to consider only spherical harmonics up to a fixed, relatively low degree in $\mathcal{D}^{\text{Inf}}$. That means, we consider a proper subset $\widehat{N} \subset \mathcal{N}$. However, this degree is unknown to us. Thus, in our learning algorithms, we aim to learn the maximal degree $\nu_0 \in \mathbb{N}_0$ of the spherical harmonics.
We proceed as follows: in each iteration $N \in \mathbb{N}_0$, we consider all spherical harmonics up to a certain degree $\nu \in \mathbb{N}$ where we suppose that $\nu_0 < \nu < \infty$ holds, i. e.

$$\widehat{N} = \{(n, j) \mid n = 0, .., \nu,\ j = -n, .., n\}$$

This is a finite number of polynomials. Thus, for each $Y_{n,j} \in \mathcal{D}^{\text{Inf}}$ up to degree ν, we can compute $\text{RFMP}(Y_{n,j}; N)$ and $\text{ROFMP}(Y_{n,j}; N)$, respectively. In particular, we have

$$\begin{aligned} \text{RFMP}(d; N) \to \max! \qquad \text{subject to} \qquad & d = Y_{n,j}, \\ & n \in \mathbb{N}_0\ n \leq \nu,\ j \in \mathbb{Z},\ -n \leq j \leq n \end{aligned}$$

and

$$\begin{aligned} \text{ROFMP}(d; N) \to \max! \qquad \text{subject to} \qquad & d = Y_{n,j}, \\ & n \in \mathbb{N}_0,\ n \leq \nu,\ j \in \mathbb{Z},\ -n \leq j \leq n \end{aligned}$$

respectively, and can determine the maximum by comparing the values among all spherical harmonics in $\mathcal{D}^{\text{Inf}}$. As we suppose that $\nu_0 < \nu < \infty$, the spherical harmonics $Y_{\nu,j}$, $j = -\nu, ..., \nu$, will not be chosen as basis elements. Thus, we define the learnt maximal degree ν_0 as

$$\begin{aligned} \nu_0 := \max\{n \mid \exists j = -n, ..., n\ \exists i = 1, ..., N_0\ \exists \alpha_i \in \mathbb{R} : \\ Y_{n,j} = d_i \text{ with } (\alpha_i, d_i) \in \{(\alpha_n, d_n)\}_{n=1,...,N_0}\} \end{aligned} \tag{7.12}$$

for a maximal iteration N_0 and the structure book $\{(\alpha_n, d_n)\}_{n=1,...,N_0}$. That means, the learnt ν_0 is the maximal degree of chosen spherical harmonics. Furthermore, the learnt spherical harmonics are exactly the spherical harmonics from the structure book.
Note that ν_0 is obtained after the LIPMP algorithm is terminated. In each iteration, the search for the most suitable spherical harmonic up to degree ν is executed in the same way as in the (non-learning) IPMP algorithms. Thus, for an

efficient implementation of the LIPMP algorithm with respect to spherical harmonics, it is sensible to perform a preprocessing in the same manner as with the underlying IPMP algorithm. Hence, we run the LIPMP algorithm with a finite starting dictionary which contains at least all spherical harmonics up to degree ν. However, of course, the starting dictionary should only have a formal or efficiency reason and not have (a major) influence on the learnt dictionary in order to have a truly automatized selection of basis elements from infinitely many possibilities. For spherical harmonics, this means, it should hold that $\nu_0 < \nu$.

Slepian functions A set of Slepian functions of band-limit $L \in \mathbb{N}$ is characterized by their localization region $R \subseteq \Omega$. In the case that R is a spherical cap, the Slepian functions can be parametrized by the size of the localization region $c \in [-1, 1]$ and its centre $A(\alpha, \beta, \gamma)\varepsilon^3$ (see Example 3.1.5 and Example 3.1.10), where $A \in \mathrm{SO}(3)$ describes a rotation matrix dependent on the Euler angles $\alpha,\ \gamma \in [0, 2\pi[$ and $\beta \in [0, \pi]$ (see Definition 3.1.6). However, if we look closely at Definition 3.1.7, we see that we can also use $\alpha, \gamma \in \mathbb{R}$ as the Wigner rotation matrices are 2π-periodic in these arguments. Due to certain symmetry relations (see e. g. Edmonds, 1996, Section 4.2), the matrices are in general not π-periodic in β which means that this constraint must not be violated. Thus, for Slepian functions, we consider the optimization problem

$$\begin{aligned} &\mathrm{RFMP}\left(g^{(k,L)}\left(\left(c, A(\alpha,\beta,\gamma)\varepsilon^3\right), \cdot\right); N\right) \to \max! \\ &\quad \text{subject to } k = 1, ..., (L+1)^2,\ c \in [-1, 1],\ \beta \in [0, \pi]\ \ (\text{and } \alpha,\ \gamma \in [0, 2\pi[) \end{aligned} \tag{7.13}$$

and

$$\begin{aligned} &\mathrm{ROFMP}\left(g^{(k,L)}\left(\left(c, A(\alpha,\beta,\gamma)\varepsilon^3\right), \cdot\right); N\right) \to \max! \\ &\quad \text{subject to } k = 1, ..., (L+1)^2,\ c \in [-1, 1],\ \beta \in [0, \pi]\ \ (\text{and } \alpha,\ \gamma \in [0, 2\pi[) \end{aligned} \tag{7.14}$$

respectively. With respect to c, α, β and γ, this is a continuous optimization problem with a non-linear objective function and bound constraints. Thus, to solve it and, by this, determine an optimized localization region R^* of a Slepian function in the current iteration, we can use well-known routines like the DIRECT and / or the SLSQP algorithm (see Appendix A.4). With this region R^*, the particular k-th Slepian function $g^{(k,L)}(R^*, \cdot)$ can be obtained by comparing the values $\mathrm{RFMP}(g^{(i,L)}(R^*, \cdot); N)$ and $\mathrm{ROFMP}(g^{(i,L)}(R^*, \cdot); N)$, respectively, for all $i = 1, ..., (L+1)^2$ as the band-limit $L < \infty$ is finite.
In general, numerical optimization routines may terminate more easily with a good solution if the optimization process starts at a point not too far away from the solution. Hence, we suggest to insert a few sets of Slepian functions of band-limit L (i. e. all Slepian functions related to a few localization regions) in the finite starting dictionary.

Abel–Poisson low and band pass filters The Abel–Poisson low and band pass filters are both based on Abel–Poisson kernels $K(x,\cdot)$ for $x \in \mathring{\mathbb{B}}$ (see Examples 3.3.4, 3.3.17 and 3.4.9). The argument $x \in \mathring{\mathbb{B}}$ defines the scale of the localization (via $|x|$) and the centre of the localization (via $x/|x|$) and, thus, characterizes each of the filters. Due to $x \in \mathring{\mathbb{B}}$, we obtain the following continuous optimization problem with a non-linear objective function and a non-linear constraint:

$$\mathrm{RFMP}(K(x,\cdot);N) \to \max! \qquad \text{subject to} \qquad \|x\|_{\mathbb{R}^3}^2 < 1, \tag{7.15}$$

$$\mathrm{ROFMP}(K(x,\cdot);N) \to \max! \qquad \text{subject to} \qquad \|x\|_{\mathbb{R}^3}^2 < 1, \tag{7.16}$$

respectively, for the Abel–Poisson low pass filters; and

$$\mathrm{RFMP}(W(x,\cdot);N) \to \max! \qquad \text{subject to} \qquad \|x\|_{\mathbb{R}^3}^2 < 1, \tag{7.17}$$

$$\mathrm{ROFMP}(W(x,\cdot);N) \to \max! \qquad \text{subject to} \qquad \|x\|_{\mathbb{R}^3}^2 < 1, \tag{7.18}$$

respectively, for the Abel–Poisson band pass filters. These problems can be solved with established methods like the DIRECT and / or the SLSQP algorithm (see Appendix A.4). With the same justification as for the Slepian functions, we suggest to insert also a few Abel–Poisson low and band pass filters into the starting dictionary.
Note that we have to take into account that a dictionary must not include the nullspace of the operator $\mathcal{T}_\neg$ because otherwise the functions $\mathrm{RFMP}(\cdot;N)$ (Definition 7.1.2 and $\mathrm{ROFMP}(\cdot;N)$ Definition 7.1.4) are not well-defined. Any nullspace of a linear operator includes the zero function. If the regularization parameter satisfies $\lambda(\delta, y^\delta) > 0$, the zero function is also the only problematic trial function. Note that the Abel–Poisson band pass filter for $x = 0$ is constantly 0. That means, actually, we would have an additional constraint in this case: $0 < \|x\|_{\mathbb{R}^3}^2$. However, in our experiments, neither does the algorithm choose this function as the next best basis element nor does it cause trouble for the used optimization software if we neglect this constraint. In the light of keeping the number of constraints low, we therefore do without this aspect in this thesis. Nonetheless, a prospective user should bear this situation in mind if problems occur in future tests. At last, note that, due to Theorem 2.4.7, the nullspace of the downward continuation operator contains only the zero function. This might be different if this learning approach is transferred to other operators of spherical inverse problems.

7.2.2. Regarding gradient-based optimization

Some of the (previously mentioned) optimization routines are gradient-based methods. It is well-known that numerical optimization algorithms are more efficient if they are gradient-based as they need less iterations than derivative-free ones. Thus, we want to use these methods in the LIPMP algorithms (as well). Therefore, we consider the partial derivatives of the objective functions $\mathrm{RFMP}(d;N)$ and $\mathrm{ROFMP}(d;N)$, respectively, for an Abel–Poisson low and band

pass filter $d(z)$, $z \in \mathbb{R}^3$, as well as for a Slepian function $d(z) \coloneqq d(z(c,\ \alpha,\ \beta,\ \gamma))$, $z \in \mathbb{R}^4$, at this point. For readability, we abbreviate

$$\begin{aligned} a_i &\coloneqq a_i(d(z)),\ i = 1, ..., 4, \\ b_j &\coloneqq b_j(d(z)),\ j = 1, ..., 5, \end{aligned}$$

and

$$\lambda \coloneqq \lambda\left(\delta, y^\delta\right)$$

from Definition 7.1.2 and Lemma 7.1.5 as well as

$$\delta_z \coloneqq \begin{cases} 3, & d(z) = K(z, \cdot) \text{ or } d = W(z, \cdot) \\ 4, & d(z(c,\ \alpha,\ \beta,\ \gamma)) = g^{(k,L)}\left(\left(c, A(\alpha, \beta, \gamma)\varepsilon^3\right), \cdot\right) \end{cases} \tag{7.19}$$

for the next considerations. Then, with the common rules for differentiation and for $z_j,\ j = 1, ..., \delta_z$, we have

$$\begin{aligned} &\frac{\partial}{\partial z_j}\mathrm{RFMP}(d(z); N) \qquad (7.20) \\ &\quad = \frac{2\left(a_1 - \lambda a_2\right)\left(\frac{\partial}{\partial z_j}a_1 - \lambda\frac{\partial}{\partial z_j}a_2\right)\left(b_1 + \lambda b_2\right) - \left(a_1 - \lambda a_2\right)^2\left(\frac{\partial}{\partial z_j}b_1 + \lambda\frac{\partial}{\partial z_j}b_2\right)}{\left(b_1 + \lambda b_2\right)^2} \\ &\frac{\partial}{\partial z_j}\mathrm{ROFMP}(d(z); N) \\ &\quad = \frac{2\left(a_3 + \lambda\left(a_4 - a_2\right)\right)\left(\frac{\partial}{\partial z_j}a_3 - \lambda\left(\frac{\partial}{\partial z_j}a_4 - \frac{\partial}{\partial z_j}a_2\right)\right)\left(b_3 + \lambda(b_2 - 2b_4 + b_5)\right)}{\left(b_3 + \lambda\left(b_2 - 2b_4 + b_5\right)\right)^2} \\ &\qquad - \frac{\left(a_3 + \lambda\left(a_4 - a_2\right)\right)^2\left(\frac{\partial}{\partial z_j}b_3 + \lambda\left(\frac{\partial}{\partial z_j}b_2 - 2\frac{\partial}{\partial z_j}b_4 + \frac{\partial}{\partial z_j}b_5\right)\right)}{\left(b_3 + \lambda\left(b_2 - 2b_4 + b_5\right)\right)^2}. \qquad (7.21) \end{aligned}$$

Note that we parametrized the Abel–Poisson low and band pass filters via a ball, i. e. via a traditional spherical setting. However, we consider the Cartesian derivatives at this point in order to avoid singularities at the poles that are otherwise inherited from a spherical parametrization.
For practical purposes, we consider the terms $a_i(d(z))$, $i = 1, ..., 4$, and $b_j(d(z))$, $j = 1, ..., 5$, in detail. Further, for the derivatives of $\mathrm{RFMP}(\cdot; N)$ and $\mathrm{ROFMP}(\cdot; N)$ as seen in (7.20) and (7.21), respectively, we also discuss

$$\begin{aligned} \frac{\partial}{\partial z_j}a_1(d(z)) &= \frac{\partial}{\partial z_j}\left\langle R^N, T\neg d\right\rangle_{\mathbb{R}^\ell}, \\ \frac{\partial}{\partial z_j}a_2(d(z)) &= \frac{\partial}{\partial z_j}\left\langle f_N, d\right\rangle_{\mathcal{H}_2(\Omega)} = \frac{\partial}{\partial z_j}\left\langle f_N^{(N)}, d\right\rangle_{\mathcal{H}_2(\Omega)}, \end{aligned}$$

$$
\begin{aligned}
\frac{\partial}{\partial z_j} a_3(d(z)) &= \frac{\partial}{\partial z_j} \left\langle R^N, \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\rangle_{\mathbb{R}^\ell}, \\
\frac{\partial}{\partial z_j} a_4(d(z)) &= \frac{\partial}{\partial z_j} \left\langle f_N^{(N)}, b_N^{(N)}(d(z)) \right\rangle_{\mathcal{H}_2(\Omega)}, \\
\frac{\partial}{\partial z_j} b_1(d(z)) &= \frac{\partial}{\partial z_j} \left\| T\neg d \right\|_{\mathbb{R}^\ell}^2, \\
\frac{\partial}{\partial z_j} b_2(d(z)) &= \frac{\partial}{\partial z_j} \|d\|_{\mathcal{H}_2(\Omega)}^2 = \frac{\partial}{\partial z_j} \langle d, d \rangle_{\mathcal{H}_2(\Omega)}, \\
\frac{\partial}{\partial z_j} b_3(d(z)) &= \frac{\partial}{\partial z_j} \left\| \mathcal{P}_{\mathcal{V}_N^\perp} T\neg d \right\|_{\mathbb{R}^\ell}^2, \\
\frac{\partial}{\partial z_j} b_4(d(z)) &= \frac{\partial}{\partial z_j} \left\langle b_N^{(N)}(d(z)), d \right\rangle_{\mathcal{H}_2(\Omega)} \\
\frac{\partial}{\partial z_j} b_5(d(z)) &= \frac{\partial}{\partial z_j} \left\| b_N^{(N)}(d(z)) \right\|_{\mathcal{H}_2(\Omega)}^2 = \frac{\partial}{\partial z_j} \left\langle b_N^{(N)}(d(z)), b_N^{(N)}(d(z)) \right\rangle_{\mathcal{H}_2(\Omega)}
\end{aligned}
$$

for the case that $d(z)$ is either a Slepian function or a Abel–Poisson low or band pass filter. Before we consider these terms, we clarify a general aspect of the differentiation of Slepian functions.

Differentiating a Slepian function with respect to the localization region Due to Definition 3.1.3, we see that the partial derivative $\partial/\partial z_m$, $z_m \in \{c,\ \alpha,\ \beta,\ \gamma\}$, acts only on the Fourier coefficients $g_{n,j}^{(k,L)}(c, A(\alpha,\beta,\gamma)\varepsilon^3)$ of a Slepian function $g^{(k,L)}((c, A(\alpha,\beta,\gamma)\varepsilon^3), \cdot)$. This is also the case for the upward continued function, see (3.18), and, thus, builds the basis for all further considerations. Hence, we have to formulate

$$
\frac{\partial}{\partial z_m} g_{n,j}^{(k,L)} \left(c, A\left(\alpha,\beta,\gamma\right)\varepsilon^3 \right) \tag{7.22}
$$

for practical purposes. That means, we have to compute the derivative of an eigenvector of the Slepian problem with respect to the spherical cap it is localized in. In Theorem 3.1.9, we have seen that the complex coefficients $\widetilde{g}_{n,j}^{(k,L)}$ of a Slepian function localized in an arbitrary spherical cap can be represented via Wigner rotation matrices and the complex coefficients with respect to the spherical cap with centre ε^3. We have the relation

$$
\widetilde{g}_{n,j}^{(k,L)} \left(c, A(\alpha,\beta,\gamma)\varepsilon^3 \right) = \sum_{i=-n}^{n} D_{j,i}^n(\alpha,\beta,\gamma) \widetilde{g}_{n,i}^{(k,L)} \left(c, \varepsilon^3 \right).
$$

Note that the exchange from real coefficients to complex ones (see Remark 2.2.9) is linear and, hence, we only have to consider the derivative of the complex coef-

ficients in the sequel. In particular, we have

$$\frac{\partial}{\partial c}\widetilde{g}_{n,j}^{(k,L)}\left(c, A(\alpha,\beta,\gamma)\varepsilon^3\right) = \sum_{i=-n}^{n} D_{j,i}^{n}(\alpha,\beta,\gamma)\frac{\partial}{\partial c}\widetilde{g}_{n,i}^{(k,L)}\left(c,\varepsilon^3\right) \tag{7.23}$$

and

$$\frac{\partial}{\partial z_m}\widetilde{g}_{n,j}^{(k,L)}\left(c, A(\alpha,\beta,\gamma)\varepsilon^3\right) = \sum_{i=-n}^{n}\left(\frac{\partial}{\partial z_m}D_{j,i}^{n}(\alpha,\beta,\gamma)\right)\widetilde{g}_{n,i}^{(k,L)}\left(c,\varepsilon^3\right) \tag{7.24}$$

for $z_m \in \{\alpha,\ \beta,\ \gamma\}$. Thus, note that, in practice, we only need to compute the derivatives of the Wigner rotation matrices with respect to the Euler angles and the derivative of the real Slepian coefficients with respect to the size of the spherical cap. Next, we consider these derivatives in detail. We first consider the derivatives with respect to the Euler angles. With the definition of the Wigner rotation matrices (Definition 3.1.7), we obtain the following recursion formulae.

$$\frac{\partial}{\partial z_m}D^0(\alpha,\beta,\gamma) = 0, \tag{7.25}$$

$$\frac{\partial}{\partial\alpha}D^1(\alpha,\beta,\gamma) = \begin{pmatrix} \frac{1+\cos(\beta)}{2}\mathrm{i}\mathrm{e}^{\mathrm{i}(\alpha+\gamma)} & \frac{\sin(\beta)}{\sqrt{2}}\mathrm{i}\mathrm{e}^{\mathrm{i}\alpha} & \frac{1-\cos(\beta)}{2}\mathrm{i}\mathrm{e}^{\mathrm{i}(\alpha-\gamma)} \\ 0 & 0 & 0 \\ -\frac{1-\cos(\beta)}{2}\mathrm{i}\mathrm{e}^{\mathrm{i}(\gamma-\alpha)} & \frac{\sin(\beta)}{\sqrt{2}}\mathrm{i}\mathrm{e}^{-\mathrm{i}\alpha} & -\frac{1+\cos(\beta)}{2}\mathrm{i}\mathrm{e}^{-\mathrm{i}(\alpha+\gamma)} \end{pmatrix}, \tag{7.26}$$

$$\frac{\partial}{\partial\beta}D^1(\alpha,\beta,\gamma) = \begin{pmatrix} -\frac{\sin(\beta)}{2}\mathrm{e}^{\mathrm{i}(\alpha+\gamma)} & \frac{\cos(\beta)}{\sqrt{2}}\mathrm{e}^{\mathrm{i}\alpha} & \frac{\sin(\beta)}{2}\mathrm{e}^{\mathrm{i}(\alpha-\gamma)} \\ -\frac{\cos(\beta)}{\sqrt{2}}\mathrm{e}^{\mathrm{i}\gamma} & -\sin(\beta) & \frac{\cos(\beta)}{\sqrt{2}}\mathrm{e}^{-\mathrm{i}\gamma} \\ \frac{\sin(\beta)}{2}\mathrm{e}^{\mathrm{i}(\gamma-\alpha)} & -\frac{\cos(\beta)}{\sqrt{2}}\mathrm{e}^{-\mathrm{i}\alpha} & -\frac{\sin(\beta)}{2}\mathrm{e}^{-\mathrm{i}(\alpha+\gamma)} \end{pmatrix}, \tag{7.27}$$

$$\frac{\partial}{\partial\gamma}D^1(\alpha,\beta,\gamma) = \begin{pmatrix} \frac{1+\cos(\beta)}{2}\mathrm{i}\mathrm{e}^{\mathrm{i}(\alpha+\gamma)} & 0 & -\frac{1-\cos(\beta)}{2}\mathrm{i}\mathrm{e}^{\mathrm{i}(\alpha-\gamma)} \\ -\frac{\sin(\beta)}{\sqrt{2}}\mathrm{i}\mathrm{e}^{\mathrm{i}\gamma} & 0 & -\frac{\sin(\beta)}{\sqrt{2}}\mathrm{i}\mathrm{e}^{-\mathrm{i}\gamma} \\ \frac{1-\cos(\beta)}{2}\mathrm{i}\mathrm{e}^{\mathrm{i}(\gamma-\alpha)} & 0 & -\frac{1+\cos(\beta)}{2}\mathrm{i}\mathrm{e}^{-\mathrm{i}(\alpha+\gamma)} \end{pmatrix}, \tag{7.28}$$

$$\begin{aligned}\frac{\partial}{\partial z_m} D^n_{k,j} = a^n_{k,j} &\left(\left(\frac{\partial}{\partial z_m} D^1_{0,0} \right) D^{n-1}_{k,j} + D^1_{0,0} \frac{\partial}{\partial z_m} D^{n-1}_{k,j} \right) \\ &+ b^n_{k,j} \left(\left(\frac{\partial}{\partial z_m} D^1_{1,0} \right) D^{n-1}_{k-1,j} + D^1_{1,0} \frac{\partial}{\partial z_m} D^{n-1}_{k-1,j} \right) \\ &+ b^n_{-k,j} \left(\left(\frac{\partial}{\partial z_m} D^1_{-1,0} \right) D^{n-1}_{k+1,j} + D^1_{-1,0} \frac{\partial}{\partial z_m} D^{n-1}_{k+1,j} \right), \\ &\qquad -n+1 \leq j \leq n-1, \end{aligned} \tag{7.29}$$

$$\begin{aligned}\frac{\partial}{\partial z_m} D^n_{k,j} = c^n_{k,-j} &\left(\left(\frac{\partial}{\partial z_m} D^1_{0,-1} \right) D^{n-1}_{k,j+1} + D^1_{0,-1} \frac{\partial}{\partial z_m} D^{n-1}_{k,j+1} \right) \\ &+ d^n_{k,-j} \left(\left(\frac{\partial}{\partial z_m} D^1_{1,-1} \right) D^{n-1}_{k-1,j+1} + D^1_{1,-1} \frac{\partial}{\partial z_m} D^{n-1}_{k-1,j+1} \right) \\ &+ d^n_{-k,-j} \left(\left(\frac{\partial}{\partial z_m} D^1_{-1,-1} \right) D^{n-1}_{k+1,j+1} + D^1_{-1,-1} \frac{\partial}{\partial z_m} D^{n-1}_{k+1,j+1} \right), \\ &\qquad -n \leq j \leq n-2, \end{aligned} \tag{7.30}$$

$$\begin{aligned}\frac{\partial}{\partial z_m} D^n_{k,j} = c^n_{k,j} &\left(\left(\frac{\partial}{\partial z_m} D^1_{0,1} \right) D^{n-1}_{k,j-1} + D^1_{0,1} \frac{\partial}{\partial z_m} D^{n-1}_{k,j-1} \right) \\ &+ d^n_{k,j} \left(\left(\frac{\partial}{\partial z_m} D^1_{1,1} \right) D^{n-1}_{k-1,j-1} + D^1_{1,1} \frac{\partial}{\partial z_m} D^{n-1}_{k-1,j-1} \right) \\ &+ d^n_{-k,j} \left(\left(\frac{\partial}{\partial z_m} D^1_{-1,1} \right) D^{n-1}_{k+1,j-1} + D^1_{-1,1} \frac{\partial}{\partial z_m} D^{n-1}_{k+1,j-1} \right), \\ &\qquad -n+2 \leq j \leq n. \end{aligned} \tag{7.31}$$

Now, we look at the derivative with respect to the size of the cap $c \in [-1,1]$. In Theorem 3.1.13, we stated an explicit formulation for the coefficients of a Slepian function. There, for a band-limit $L \in \mathbb{N}$ and an order $j \in \mathbb{Z}_0$, $-L \leq j \leq L$, we first compute the values $b_{i,i+1}$, $i = 1, ..., L-|j|$, and afterwards the values of the vector v_i, $i = L-|j|+1, ..., 1$. In the same routine, we obtain the derivatives. First, we state the derivatives of the entries of the commuting matrix, see Theorem 3.1.11. We have

$$\frac{\partial}{\partial c} M^j_{(m,k),(n,j)}(c) = \begin{cases} -n(n+1), & (m,k) = (n,j) \\ 0, & else. \end{cases} \tag{7.32}$$

Before we can state the derivatives of $b_{i,i+1}$ and v_i, respectively, we have to look at the derivative with respect to c of an eigenvalue $\rho(c)$. We know (see (2.6)) that an eigenvalue solves, in our case, the equation

$$F(c, \rho(c)) := \det\left(M^j(c) - \rho(c) I \right) = 0.$$

We consider the total differential of F and use the chain rule for this:

$$0 = \frac{\mathrm{d}}{\mathrm{d}c} F(c, \rho(c)) = \frac{\partial}{\partial c} F(c, \rho(c)) + \frac{\partial}{\partial \rho} F(c, \rho(c)) \rho'(c).$$

Therefore, we obtain

$$\rho'(c) = -\frac{\frac{\partial}{\partial c}F\left(c, \rho(c)\right)}{\frac{\partial}{\partial \rho}F\left(c, \rho(c)\right)}$$

if $\frac{\partial}{\partial \rho}F\left(c, \rho(c)\right) \neq 0$. However, we know, for example from Seibert (2018), that, for the matrix M^j with $j = -L, ..., L$, each eigenspace has dimension 1. Thus, it holds $\frac{\partial}{\partial \rho}F\left(c, \rho(c)\right) \neq 0$ for all eigenvalues $\rho(c)$ of M^j. For $\rho'(c)$, we consider the nominator and denominator separately and start with the former one:

$$\begin{aligned}
&\frac{\partial}{\partial c}F\left(c, \rho(c)\right) = \frac{\partial}{\partial c}\det\Big(M^j(c) - \rho(c)I\Big) \\
&= \sum_{i=1}^{L-|j|+1}\sum_{m=1}^{L-|j|+1}(-1)^{i+m}\left(\frac{\partial}{\partial c}\Big(M^j(c)_{(i,|j|),(m,|j|)} - \rho(c)\delta_{i,m}\Big)\right) \\
&\qquad\qquad \times \det\Big(M^j(c) - \rho(c)I\Big)^{i,m} \\
&= \sum_{i=1}^{L-|j|+1}\sum_{m=1}^{L-|j|+1}(-1)^{i+m}\left(-i(i+1)\delta_{i,m}\right)\det\Big(M^j(c) - \rho(c)I\Big)^{i,m} \\
&= -\sum_{i=1}^{L-|j|+1} i(i+1)\det\Big(M^j(c) - \rho(c)I\Big)^{i,i}
\end{aligned}$$

where we used (2.7) and (7.32). For the denominator, we obtain in a similar fashion the following result:

$$\begin{aligned}
&\frac{\partial}{\partial \rho}F\left(c, \rho(c)\right) = \frac{\partial}{\partial \rho}\det\Big(M^j(c) - \rho(c)I\Big) \\
&= \sum_{i=1}^{L-|j|+1}\sum_{m=1}^{L-|j|+1}(-1)^{i+m}\left(\frac{\partial}{\partial \rho}\Big(M^j(c)_{(i,|j|),(m,|j|)} - \rho(c)\delta_{i,m}\Big)\right) \\
&\qquad\qquad \times \det\Big(M^j(c) - \rho(c)I\Big)^{i,m} \\
&= \sum_{i=1}^{L-|j|+1}\sum_{m=1}^{L-|j|+1}(-1)^{i+m}\left(-\delta_{i,m}\right)\det\Big(M^j(c) - \rho(c)I\Big)^{i,m} \\
&= -\sum_{i=1}^{L-|j|+1}\det\Big(M^j(c) - \rho(c)I\Big)^{i,i}.
\end{aligned}$$

Hence, we see that

$$\rho'(c) = -\frac{\sum_{i=1}^{L-|j|+1} i(i+1)\det\left(M^j(c) - \rho(c)I\right)^{i,i}}{\sum_{i=1}^{L-|j|+1}\det\left(M^j(c) - \rho(c)I\right)^{i,i}},$$

where we obtain $\det\left(M^j(c)-\rho(c)I\right)_{i,i}$, $i=1,...,L-|j|+1$, in the (de-)nominator with the use of the three term recursion (2.8).
Using the result, we have for the derivatives of the values $b_{i,i+1}(c)$ (confer Theorem 3.1.13):

$$\begin{aligned}
\frac{\partial}{\partial c}b_{1,2}(c) &= \frac{\partial}{\partial c}\frac{\left(M^j(c)\right)_{(|j|,|j|),(|j|+1,|j|)}}{\left(M^j(c)\right)_{(|j|,|j|),(|j|,|j|)}-\rho(c)} \\
&= \frac{\left(\frac{\partial}{\partial c}\left(M^j(c)\right)_{(|j|,|j|),(|j|+1,|j|)}\right)\left(\left(M^j(c)\right)_{(|j|,|j|),(|j|,|j|)}-\rho(c)\right)}{\left(\left(M^j(c)\right)_{(|j|,|j|),(|j|,|j|)}-\rho(c)\right)^2} \\
&\quad - \frac{\left(M^j(c)\right)_{(|j|,|j|),(|j|+1,|j|)}\left(\left(\frac{\partial}{\partial c}\left(M^j(c)\right)_{(|j|,|j|),(|j|,|j|)}\right)-\rho'(c)\right)}{\left(\left(M^j(c)\right)_{(|j|,|j|),(|j|,|j|)}-\rho(c)\right)^2} \\
&= -\frac{\left(M^j(c)\right)_{(|j|,|j|),(|j|+1,|j|)}\left(-|j|(|j|+1)-\rho'(c)\right)}{\left(\left(M^j(c)\right)_{(|j|,|j|),(|j|,|j|)}-\rho(c)\right)^2} \\
&= \frac{\left(M^j(c)\right)_{(|j|,|j|),(|j|+1,|j|)}\left(|j|(|j|+1)+\rho'(c)\right)}{\left(\left(M^j(c)\right)_{(|j|,|j|),(|j|,|j|)}-\rho(c)\right)^2} \qquad (7.33)
\end{aligned}$$

and

$$\begin{aligned}
&\frac{\partial}{\partial c}b_{i,i+1}(c) \\
&= \frac{\partial}{\partial c}\frac{\left(M^j(c)\right)_{(i,|j|),(i+1,|j|)}}{\left(M^j(c)\right)_{(i,|j|),(i,|j|)}-\rho(c)-\left(M^j(c)\right)_{(i,|j|),(i-1,|j|)}b_{i-1,i}(c)} \\
&= \frac{\left[\frac{\partial}{\partial c}\left(M^j(c)\right)_{(i,|j|),(i+1,|j|)}\right]}{\left[\left(M^j(c)\right)_{(i,|j|),(i,|j|)}-\rho(c)-\left(M^j(c)\right)_{(i,|j|),(i-1,|j|)}b_{i-1,i}(c)\right]^2} \\
&\qquad \times\left[\left(M^j(c)\right)_{(i,|j|),(i,|j|)}-\rho(c)-\left(M^j(c)\right)_{(i,|j|),(i-1,|j|)}b_{i-1,i}(c)\right] \\
&\quad - \frac{\left(M^j(c)\right)_{(i,|j|),(i+1,|j|)}}{\left[\left(M^j(c)\right)_{(i,|j|),(i,|j|)}-\rho(c)-\left(M^j(c)\right)_{(i,|j|),(i-1,|j|)}b_{i-1,i}(c)\right]^2} \\
&\qquad \times\left[\frac{\partial}{\partial c}\left(M^j(c)\right)_{(i,|j|),(i,|j|)}-\rho'(c)-\frac{\partial}{\partial c}\left(\left(M^j(c)\right)_{(i,|j|),(i-1,|j|)}b_{i-1,i}(c)\right)\right]
\end{aligned}$$

$$= -\frac{\left(M^j(c)\right)_{(i,|j|),(i+1,|j|)}\left[-i(i+1)-\rho'(c)-\left(M^j(c)\right)_{(i,|j|),(i-1,|j|)}\frac{\partial}{\partial c}b_{i-1,i}(c)\right]}{\left[\left(M^j(c)\right)_{(i,|j|),(i,|j|)}-\rho(c)-\left(M^j(c)\right)_{(i,|j|),(i-1,|j|)}b_{i-1,i}(c)\right]^2}$$

$$= \frac{\left(M^j(c)\right)_{(i,|j|),(i+1,|j|)}\left[i(i+1)+\rho'(c)+\left(M^j(c)\right)_{(i,|j|),(i-1,|j|)}\frac{\partial}{\partial c}b_{i-1,i}(c)\right]}{\left[\left(M^j(c)\right)_{(i,|j|),(i,|j|)}-\rho(c)-\left(M^j(c)\right)_{(i,|j|),(i-1,|j|)}b_{i-1,i}(c)\right]^2}. \quad (7.34)$$

This gives us the derivative of each eigenvector coefficient as

$$\frac{\partial}{\partial c}v_{L-|j|+1}(c) = 0 \quad (7.35)$$

and, for $i = 1, ..., L-|j|$,

$$\begin{aligned}\frac{\partial}{\partial c}v_i(c) &= -\frac{\partial}{\partial c}\left(b_{i,i+1}(c)v_{i+1}(c)\right)\\ &= -\left(\frac{\partial}{\partial c}b_{i,i+1}(c)\right)v_{i+1}(c) - b_{i,i+1}(c)\left(\frac{\partial}{\partial c}v_{i+1}(c)\right). \quad (7.36)\end{aligned}$$

Next, we discuss the terms $a_i(d(z))$, $i = 1, ..., 4$, and $b_j(d(z))$, $j = 1, ..., 5$, from Definition 7.1.2 and Lemma 7.1.5 as well as their derivatives as given in the beginning of this subsection. Note that, for previously chosen basis elements, we neglect the argument where possible and simply write $d_n := d_n(z_n)$.

7.2.3. Inner products dependent on the operator

First, we consider the terms $a_1(d(z))$, $a_3(d(z))$, $b_1(d(z))$ and $b_3(d(z))$ for $z \in \mathbb{R}^{\delta_z}$ (confer (7.19)) with respect to Slepian functions as well as Abel–Poisson low and band pass filters. We recall that, using (4.4) as well as Theorem 4.4.3 for $\mathcal{P}_{\mathcal{V}_N^\perp}T\neg d(z)$, we have

$$\mathcal{P}_{\mathcal{V}_N^\perp}T\neg d(z) = T\neg d(z) - \mathcal{P}_{\mathcal{V}_N}T\neg d(z) = T\neg d(z) - \sum_{n=1}^{N}\beta_n^{(N)}(d(z))T\neg d_n. \quad (7.37)$$

For the derivative with respect to z_j, $j = 1, ..., \delta_z$, we obtain

$$\frac{\partial}{\partial z_j}\mathcal{P}_{\mathcal{V}_N^\perp}T\neg d(z) = \frac{\partial}{\partial z_j}T\neg d(z) - \sum_{n=1}^{N}\left(\frac{\partial}{\partial z_j}\beta_n^{(N)}(d(z))\right)T\neg d_n. \quad (7.38)$$

Hence, we consider the derivative of the projection coefficients $\beta_n^{(N)}(d(z))$, $n = 1, ..., N$, of the current iteration N. It holds

$$\frac{\partial}{\partial z_j}\beta_N^{(N)}(d(z)) = \frac{\left\langle \frac{\partial}{\partial z_j} T\neg d(z), \mathcal{P}_{\mathcal{V}_{N-1}^{\perp}} T\neg d_N \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_{N-1}^{\perp}} T\neg d_N \right\|_{\mathbb{R}^\ell}^2} \tag{7.39}$$

$$\begin{aligned}
&\frac{\partial}{\partial z_j}\beta_n^{(N)}(d(z)) \\
&\quad = \frac{\left\langle \frac{\partial}{\partial z_j} T\neg d(z), \mathcal{P}_{\mathcal{V}_{n-1}^{\perp}} T\neg d_n \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_{n-1}^{\perp}} T\neg d_n \right\|_{\mathbb{R}^\ell}^2} - \sum_{j=n+1}^{N} \frac{\left\langle \frac{\partial}{\partial z_j} T\neg d(z), \mathcal{P}_{\mathcal{V}_{j-1}^{\perp}} T\neg d_j \right\rangle_{\mathbb{R}^\ell}}{\left\| \mathcal{P}_{\mathcal{V}_{j-1}^{\perp}} T\neg d_j \right\|_{\mathbb{R}^\ell}^2} \beta_n^{(j-1)}(d_n)
\end{aligned} \tag{7.40}$$

for $n = 1, ..., N-1$. Thus, the derivatives of the projection coefficients depend only on the derivatives of $T\neg d$. Moreover, we now see that all terms $a_1(d(z))$, $a_3(d(z))$, $b_1(d(z))$ and $b_3(d(z))$ depend only on $T\neg$ and cannot be discussed any further for an arbitrary operator.
For the upward continuation operator $\mathcal{T}$, we gave the upward continued values of our dictionary elements in (2.27), (3.16), (3.17) and (3.18).
At this point, however, we consider an explicit form of $\mathcal{T}K(x, \cdot)$ and $\mathcal{T}W(x, \cdot)$ (with $x = z$) which is useful for practical purposes. For any $\eta \in \Omega$, we obtain for the Abel–Poisson low pass filter $K(x, \cdot)$

$$\begin{aligned}
\mathcal{T}K(x, \cdot)(\sigma\eta) = \frac{1}{\sigma} K\left(\frac{x}{\sigma}, \eta\right) &= \frac{1}{\sigma} \frac{1 - \left|\frac{x}{\sigma}\right|^2}{4\pi \left[1 + \left|\frac{x}{\sigma}\right|^2 - 2\frac{x}{\sigma} \cdot \eta\right]^{3/2}} \\
&= \frac{\sigma^{-2}\left(\sigma^2 - |x|^2\right)}{4\pi\sigma \left[\sigma^{-2}\left(\sigma^2 + |x|^2 - 2\sigma x \cdot \eta\right)\right]^{3/2}} \\
&= \frac{\sigma^{-2}\left(\sigma^2 - |x|^2\right)}{4\pi\sigma\sigma^{-3} \left[\sigma^2 + |x|^2 - 2\sigma x \cdot \eta\right]^{3/2}} \\
&= \frac{\sigma^2 - |x|^2}{4\pi \left[\sigma^2 + |x|^2 - 2\sigma x \cdot \eta\right]^{3/2}}.
\end{aligned} \tag{7.41}$$

Note that this is in accordance with the definition of the Abel–Poisson kernel on a ball with radius σ (see e. g. Freeden et al., 1998, p. 86). Hence, for the Abel–

Poisson band pass filter $W(x,\cdot)$, we obtain

$$\begin{aligned}\mathcal{T}W(x,\cdot)(\sigma\eta) &= \frac{1}{\sigma}K\left(\frac{x}{\sigma},\eta\right) - \frac{1}{\sigma}K\left(\frac{|x|x}{\sigma},\eta\right)\\ &= \frac{\sigma^2-|x|^2}{4\pi\left[\sigma^2+|x|^2-2\sigma x\cdot\eta\right]^{3/2}} - \frac{\sigma^2-|x|^4}{4\pi\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^{3/2}}.\end{aligned} \tag{7.42}$$

As we have seen, for the gradients of the terms $a_1(d(z))$, $a_3(d(z))$, $b_1(d(z))$ and $b_3(d(z))$, we are interested in particular in $\partial/\partial x_j \mathcal{T}d(x)$ for an Abel–Poisson low and band pass filters. In the sequel, we use

$$|x|^2 = \sum_{j=1}^{3} x_j^2. \tag{7.43}$$

Then, for the derivative of $\mathcal{T}K(x,\cdot)$, we obtain

$$\begin{aligned}\frac{\partial}{\partial x_j}\mathcal{T}K(x,\cdot)(\sigma\eta) &= \frac{\partial}{\partial x_j}\,\frac{\sigma^2-|x|^2}{4\pi\left[\sigma^2+|x|^2-2\sigma x\cdot\eta\right]^{3/2}}\\ &= \frac{1}{4\pi}\left(\frac{\left(\frac{\partial}{\partial x_j}\left(\sigma^2-\sum_{k=1}^{3}x_k^2\right)\right)\left[\sigma^2+|x|^2-2\sigma x\cdot\eta\right]^{3/2}}{\left[\sigma^2+|x|^2-2\sigma x\cdot\eta\right]^{3}}\right.\\ &\qquad\left.-\frac{\left(\sigma^2-|x|^2\right)\frac{\partial}{\partial x_j}\left[\sigma^2+\sum_{k=1}^{3}x_k^2-2\sigma\sum_{k=1}^{3}x_k\eta_k\right]^{3/2}}{\left[\sigma^2+|x|^2-2\sigma x\cdot\eta\right]^{3}}\right)\\ &= \frac{1}{4\pi}\left(-\frac{2x_j}{\left[\sigma^2+|x|^2-2\sigma x\cdot\eta\right]^{3/2}}\right.\\ &\qquad\left.-\frac{\left(\sigma^2-|x|^2\right)\frac{3}{2}\left[\sigma^2+|x|^2-2\sigma x\cdot\eta\right]^{1/2}\left(2x_j-2\sigma\eta_j\right)}{\left[\sigma^2+|x|^2-2\sigma x\cdot\eta\right]^{3}}\right)\\ &= \frac{1}{4\pi}\left(-\frac{2x_j}{\left[\sigma^2+|x|^2-2\sigma x\cdot\eta\right]^{3/2}} - \frac{3\left(\sigma^2-|x|^2\right)\left(x_j-\sigma\eta_j\right)}{\left[\sigma^2+|x|^2-2\sigma x\cdot\eta\right]^{5/2}}\right)\end{aligned} \tag{7.44}$$

for $\eta\in\Omega$ and $\sigma>1$. With this result, we obtain for the derivative of $\mathcal{T}W(x,\cdot)$ with respect to x_j that it holds

$$\frac{\partial}{\partial x_j}\mathcal{T}W(x,\cdot)(\sigma\eta) = \frac{\partial}{\partial x_j}\mathcal{T}K(x,\cdot)(\sigma\eta) - \frac{\partial}{\partial x_j}\mathcal{T}K(|x|x,\cdot)(\sigma\eta) \tag{7.45}$$

$$= \frac{1}{4\pi}\left(-\frac{2x_j}{\left[\sigma^2+|x|^2-2\sigma x\cdot\eta\right]^{3/2}} - \frac{3\left(\sigma^2-|x|^2\right)\left(x_j-\sigma\eta_j\right)}{\left[\sigma^2+|x|^2-2\sigma x\cdot\eta\right]^{5/2}} \right.$$
$$+ \frac{4|x|^2x_j}{\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^{3/2}}$$
$$\left. + \frac{3\left(\sigma^2-|x|^4\right)\left(2|x|^2x_j-\sigma|x|^{-1}x_jx\cdot\eta-\sigma|x|\eta_j\right)}{4\pi\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^{5/2}} \right). \tag{7.46}$$

For the first term of the right-hand side of (7.45), we have already obtained (7.44). For the second term, we obtain with (7.42) and (7.43) analogously

$$\frac{\partial}{\partial x_j}\mathcal{T}K(|x|x,\cdot)(\sigma\eta)$$
$$= \frac{\partial}{\partial x_j}\frac{\sigma^2-|x|^4}{4\pi\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^{3/2}}$$
$$= \frac{1}{4\pi}\left(\frac{\left(\frac{\partial}{\partial x_j}\left(\sigma^2-\left(\sum_{k=1}^3 x_k^2\right)^2\right)\right)\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^{3/2}}{\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^3} \right.$$
$$\left. - \frac{\left(\sigma^2-|x|^4\right)\frac{\partial}{\partial x_j}\left[\sigma^2+\left(\sum_{k=1}^3 x_k^2\right)^2-2\sigma\left(\sum_{k=1}^3 x_k^2\right)^{1/2}\left(\sum_{k=1}^3 x_k\eta_k\right)\right]^{3/2}}{\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^3} \right)$$
$$= \frac{1}{4\pi}\left(\frac{-4\left(\sum_{k=1}^3 x_k^2\right)x_j}{\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^{3/2}} - \frac{\left(\sigma^2-|x|^4\right)\frac{3}{2}\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^{1/2}}{\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^3} \right.$$
$$\left. \times\left[4\left[\sum_{k=1}^3 x_k^2\right]x_j - 2\sigma\frac{1}{2}\left[\sum_{k=1}^3 x_k^2\right]^{-1/2}(2x_j)\left[\sum_{k=1}^3 x_k\eta_k\right] - 2\sigma\left[\sum_{k=1}^3 x_k^2\right]^{1/2}\eta_j\right]\right)$$
$$= \frac{1}{4\pi}\left(\frac{-4|x|^2x_j}{\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^{3/2}} \right.$$
$$\left. - \frac{\frac{3}{2}\left(\sigma^2-|x|^4\right)\left(4|x|^2x_j-2\sigma\frac{1}{2}|x|^{-1}(2x_j)x\cdot\eta-2\sigma|x|\eta_j\right)}{\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^{5/2}} \right)$$

$$= \frac{1}{4\pi}\left(\frac{-4|x|^2 x_j}{\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^{3/2}} - \frac{3\left(\sigma^2-|x|^4\right)\left(2|x|^2x_j-\sigma|x|^{-1}x_j x\cdot\eta-\sigma|x|\eta_j\right)}{\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta\right]^{5/2}}\right)$$

for $\eta \in \Omega$ and $\sigma > 1$. Taking all these things into consideration, we now give the terms $a_1(d(z))$, $a_3(d(z))$, $b_1(d(z))$ and $b_3(d(z))$ for $z \in \mathbb{R}^{\delta_z}$ (confer (7.19)). Note that we use the discretized operator $\mathcal{T}_\urcorner$ in these terms. That means, we evaluate $\mathcal{T}d$ at certain gridpoints $\sigma\eta^{(i)}$, $\sigma > 1$, $\eta^{(i)} \in \Omega$, $i = 1, ..., \ell$. Correspondingly, we use $e^{(i)} = (\delta_{j,i})_{j=1,...,\ell}$ for the i-th canonical Cartesian basis vector. Further, note that, for each of the terms discussed next, we give references to all previous equations needed for computation. We obtain the following formulations of the terms in question:

$$a_1(K(x,\cdot)) = \sum_{i=1}^{\ell} R_i^N \mathcal{T}_\urcorner^i K(x,\cdot) = \sum_{i=1}^{\ell} \frac{R_i^N\left(\sigma^2-|x|^2\right)}{4\pi\left[\sigma^2+|x|^2-2\sigma x\cdot\eta^{(i)}\right]^{3/2}} \tag{7.47}$$

using (7.41),

$$\begin{aligned}\frac{\partial}{\partial x_j}a_1(K(x,\cdot)) &= \sum_{i=1}^{\ell} R_i^N \frac{\partial}{\partial x_j}\mathcal{T}_\urcorner^i K(x,\cdot)\\ &= \sum_{i=1}^{\ell}\frac{R_i^N}{4\pi}\left(-\frac{2x_j}{\left[\sigma^2+|x|^2-2\sigma x\cdot\eta^{(i)}\right]^{3/2}} - \frac{3\left(\sigma^2-|x|^2\right)\left(x_j-\sigma\eta_j^{(i)}\right)}{\left[\sigma^2+|x|^2-2\sigma x\cdot\eta^{(i)}\right]^{5/2}}\right)\end{aligned} \tag{7.48}$$

using (7.44),

$$\begin{aligned}a_3(K(x,\cdot)) &= \sum_{i=1}^{\ell} R_i^N\left(e^{(i)}\cdot\mathcal{P}_{\mathcal{V}_N^\perp}\mathcal{T}_\urcorner K(x,\cdot)\right)\\ &= \sum_{i=1}^{\ell} R_i^N\left(\mathcal{T}_\urcorner^i K(x,\cdot) - \sum_{n=1}^{N}\beta_n^{(N)}(K(x,\cdot))\mathcal{T}_\urcorner^i d_n\right)\end{aligned} \tag{7.49}$$

using (2.27), (3.18), Theorem 4.4.3, (7.37), (7.41) and (7.42),

$$\begin{aligned}\frac{\partial}{\partial x_j}a_3(K(x,\cdot)) &= \sum_{i=1}^{\ell} R_i^N\frac{\partial}{\partial x_j}\left(e^{(i)}\cdot\mathcal{P}_{\mathcal{V}_N^\perp}\mathcal{T}_\urcorner K(x,\cdot)\right)\\ &= \sum_{i=1}^{\ell} R_i^N\left(\frac{\partial}{\partial x_j}\mathcal{T}_\urcorner^i K(x,\cdot) - \sum_{n=1}^{N}\left(\frac{\partial}{\partial x_j}\beta_n^{(N)}(K(x,\cdot))\right)\mathcal{T}_\urcorner^i d_n\right)\end{aligned} \tag{7.50}$$

using (2.27), (3.18), (7.38), (7.39), (7.40), (7.41), (7.42) and (7.44),

$$b_1(K(x,\cdot)) = \sum_{i=1}^{\ell} \left(\mathcal{T}_{\neg}^{i} K(x,\cdot)\right)^2 = \sum_{i=1}^{\ell} \frac{\left(\sigma^2 - |x|^2\right)^2}{16\pi^2 \left[\sigma^2 + |x|^2 - 2\sigma x \cdot \eta^{(i)}\right]^3} \tag{7.51}$$

using (7.41),

$$\frac{\partial}{\partial x_j} b_1(K(x,\cdot)) = \sum_{i=1}^{\ell} \frac{\partial}{\partial x_j} \left(\mathcal{T}_{\neg}^{i} K(x,\cdot)\right)^2 \quad = \sum_{i=1}^{\ell} 2 \left(\mathcal{T}_{\neg}^{i} K(x,\cdot)\right) \frac{\partial}{\partial x_j} \left(\mathcal{T}_{\neg}^{i} K(x,\cdot)\right) \tag{7.52}$$

using (7.41) and (7.44),

$$\begin{aligned} b_3(K(x,\cdot)) &= \sum_{i=1}^{\ell} \left(e^{(i)} \cdot \mathcal{P}_{\mathcal{V}_N^{\perp}} \mathcal{T}_{\neg} K(x,\cdot)\right)^2 \\ &= \sum_{i=1}^{\ell} \left(\mathcal{T}_{\neg}^{i} K(x,\cdot) - \sum_{n=1}^{N} \beta_n^{(N)} \left(K(x,\cdot)\right) \mathcal{T}_{\neg}^{i} d_n\right)^2 \end{aligned} \tag{7.53}$$

using (2.27), (3.18), Theorem 4.4.3, (7.37), (7.41), (7.42),

$$\begin{aligned} \frac{\partial}{\partial x_j} b_3\left(K(x,\cdot)\right) &= \sum_{i=1}^{\ell} \frac{\partial}{\partial x_j} \left(e^{(i)} \cdot \mathcal{P}_{\mathcal{V}_N^{\perp}} \mathcal{T}_{\neg} K(x,\cdot)\right)^2 \\ &= \sum_{i=1}^{\ell} 2 \left(e^{(i)} \cdot \mathcal{P}_{\mathcal{V}_N^{\perp}} \mathcal{T}_{\neg} K(x,\cdot)\right) \left(\frac{\partial}{\partial x_j} \mathcal{T}_{\neg}^{i} K(x,\cdot) - \sum_{n=1}^{N} \left(\frac{\partial}{\partial x_j} \beta_n^{(N)} \left(K(x,\cdot)\right)\right) \mathcal{T}_{\neg}^{i} d_n\right) \end{aligned} \tag{7.54}$$

using (2.27), (3.18), Theorem 4.4.3, (7.37), (7.38), (7.39), (7.40), (7.41), (7.42) and (7.44),

$$\begin{aligned} a_1(W(x,\cdot)) &= \sum_{i=1}^{\ell} R_i^N \mathcal{T}_{\neg}^{i} W(x,\cdot) \\ &= \sum_{i=1}^{\ell} \frac{R_i^N}{4\pi} \left(\frac{\sigma^2 - |x|^2}{\left[\sigma^2 + |x|^2 - 2\sigma x \cdot \eta^{(i)}\right]^{3/2}} - \frac{\sigma^2 - |x|^4}{\left[\sigma^2 + |x|^4 - 2\sigma |x| x \cdot \eta^{(i)}\right]^{3/2}}\right) \end{aligned} \tag{7.55}$$

using (7.42),

$$\begin{aligned}\frac{\partial}{\partial x_j}a_1(W(x,\cdot)) &= \sum_{i=1}^{\ell} R_i^N \frac{\partial}{\partial x_j}\mathcal{T}_{\neg}^i W(x,\cdot)\\ &= \sum_{i=1}^{\ell}\frac{R_i^N}{4\pi}\left(-\frac{2x_j}{\left[\sigma^2+|x|^2-2\sigma x\cdot\eta^{(i)}\right]^{3/2}}-\frac{3\left(\sigma^2-|x|^2\right)\left(x_j-\sigma\eta_j^{(i)}\right)}{\left[\sigma^2+|x|^2-2\sigma x\cdot\eta^{(i)}\right]^{5/2}}\right.\\ &\qquad+\frac{4|x|^2x_j}{\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta^{(i)}\right]^{3/2}}\\ &\qquad\left.+\frac{3\left(\sigma^2-|x|^4\right)\left(2|x|^2x_j-\sigma|x|^{-1}x_jx\cdot\eta^{(i)}-\sigma|x|\eta_j^{(i)}\right)}{4\pi\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta^{(i)}\right]^{5/2}}\right)\end{aligned}\tag{7.56}$$

using (7.46),

$$\begin{aligned}a_3(W(x,\cdot)) &= \sum_{i=1}^{\ell} R_i^N\left(e^{(i)}\cdot\mathcal{P}_{\mathcal{V}_N^{\perp}}\mathcal{T}_{\neg}W(x,\cdot)\right)\\ &= \sum_{i=1}^{\ell} R_i^N\left(\mathcal{T}_{\neg}^iW(x,\cdot)-\sum_{n=1}^{N}\beta_n^{(N)}(W(x,\cdot))\mathcal{T}_{\neg}^id_n\right)\end{aligned}\tag{7.57}$$

using (2.27), (3.18), Theorem 4.4.3, (7.37), (7.41) and (7.42),

$$\begin{aligned}\frac{\partial}{\partial x_j}a_3(W(x,\cdot)) &= \sum_{i=1}^{\ell} R_i^N\frac{\partial}{\partial x_j}\left(e^{(i)}\cdot\mathcal{P}_{\mathcal{V}_N^{\perp}}\mathcal{T}_{\neg}W(x,\cdot)\right)\\ &= \sum_{i=1}^{\ell} R_i^N\left(\frac{\partial}{\partial x_j}\mathcal{T}_{\neg}^iW(x,\cdot)-\sum_{n=1}^{N}\left(\frac{\partial}{\partial x_j}\beta_n^{(N)}(W(x,\cdot))\right)\mathcal{T}_{\neg}^id_n\right)\end{aligned}\tag{7.58}$$

using (2.27), (3.18), (7.38), (7.39), (7.40), (7.41), (7.42) and (7.46),

$$\begin{aligned}b_1(W(x,\cdot)) &= \sum_{i=1}^{\ell}\left(\mathcal{T}_{\neg}^iW(x,\cdot)\right)^2\\ &= \sum_{i=1}^{\ell}\left(\frac{\sigma^2-|x|^2}{4\pi\left[\sigma^2+|x|^2-2\sigma x\cdot\eta^{(i)}\right]^{3/2}}-\frac{\sigma^2-|x|^4}{4\pi\left[\sigma^2+|x|^4-2\sigma|x|x\cdot\eta^{(i)}\right]^{3/2}}\right)^2\end{aligned}\tag{7.59}$$

using (7.42),

$$\frac{\partial}{\partial x_j}b_1(W(x,\cdot)) = \sum_{i=1}^{\ell}\frac{\partial}{\partial x_j}\left(\mathcal{T}_{\neg}^iW(x,\cdot)\right)^2 = \sum_{i=1}^{\ell}2\left(\mathcal{T}_{\neg}^iW(x,\cdot)\right)\frac{\partial}{\partial x_j}\left(\mathcal{T}_{\neg}^iW(x,\cdot)\right)\tag{7.60}$$

using (7.42) and (7.46),

$$b_3(W(x,\cdot)) = \sum_{i=1}^{\ell} \left(e^{(i)} \cdot \mathcal{P}_{\mathcal{V}_N^{\perp}} \mathcal{T}_{\neg} W(x,\cdot) \right)^2 = \sum_{i=1}^{\ell} \left(\mathcal{T}_{\neg}^i W(x,\cdot) - \sum_{n=1}^{N} \beta_n^{(N)} \left(W(x,\cdot) \right) \mathcal{T}_{\neg}^i d_n \right)^2 \tag{7.61}$$

using (2.27), (3.18), Theorem 4.4.3, (7.37), (7.41), (7.42),

$$\frac{\partial}{\partial x_j} b_3 \left(W(x,\cdot) \right) = \sum_{i=1}^{\ell} \frac{\partial}{\partial x_j} \left(e^{(i)} \cdot \mathcal{P}_{\mathcal{V}_N^{\perp}} \mathcal{T}_{\neg} W(x,\cdot) \right)^2 = \sum_{i=1}^{\ell} 2 \left(e^{(i)} \cdot \mathcal{P}_{\mathcal{V}_N^{\perp}} \mathcal{T}_{\neg} W(x,\cdot) \right) \left(\frac{\partial}{\partial x_j} \mathcal{T}_{\neg}^i W(x,\cdot) - \sum_{n=1}^{N} \left(\frac{\partial}{\partial x_j} \beta_n^{(N)} \left(W(x,\cdot) \right) \right) \mathcal{T}_{\neg}^i d_n \right) \tag{7.62}$$

using (2.27), (3.18), Theorem 4.4.3, (7.37), (7.38), (7.39), (7.40), (7.41), (7.42) and (7.46),

$$a_1 \left(g^{(k,L)} \left(\left(c, A\varepsilon^3 \right), \cdot \right) \right) = \sum_{i=1}^{\ell} R_i^N \mathcal{T}_{\neg}^i g^{(k,L)} \left(\left(c, A\varepsilon^3 \right), \cdot \right) = \sum_{i=1}^{\ell} \sum_{n=0}^{L} \sum_{j=-n}^{n} R_i^N g_{n,j}^{(k,L)} \left(c, A\varepsilon^3 \right) \sigma^{-n-1} Y_{n,j} \left(\eta^{(i)} \right) \tag{7.63}$$

using (3.18),

$$\frac{\partial}{\partial z_m} a_1 \left(g^{(k,L)} \left(\left(c, A\varepsilon^3 \right), \cdot \right) \right) = \sum_{i=1}^{\ell} R_i^N \frac{\partial}{\partial z_m} \mathcal{T}_{\neg}^i g^{(k,L)} \left(\left(c, A\varepsilon^3 \right), \cdot \right) = \sum_{i=1}^{\ell} \sum_{n=0}^{L} \sum_{j=-n}^{n} R_i^N \left(\frac{\partial}{\partial z_m} g_{n,j}^{(k,L)} \left(c, A\varepsilon^3 \right) \right) \sigma^{-n-1} Y_{n,j} \left(\eta^{(i)} \right) \tag{7.64}$$

using (7.24), (7.35) and (7.36),

$$a_3 \left(g^{(k,L)} \left(\left(c, A\varepsilon^3 \right), \cdot \right) \right) = \sum_{i=1}^{\ell} R_i^N \left(e^{(i)} \cdot \mathcal{P}_{\mathcal{V}_N^{\perp}} \mathcal{T}_{\neg} g^{(k,L)} \left(\left(c, A\varepsilon^3 \right), \cdot \right) \right) = \sum_{i=1}^{\ell} R_i^N \left(\mathcal{T}_{\neg}^i g^{(k,L)} \left(\left(c, A\varepsilon^3 \right), \cdot \right) - \sum_{n=1}^{N} \beta_n^{(N)} \left(g^{(k,L)} \left(\left(c, A\varepsilon^3 \right), \cdot \right) \right) \mathcal{T}_{\neg}^i d_n \right) \tag{7.65}$$

using (2.27), (3.18), Theorem 4.4.3, (7.37), (7.41) and (7.42),

$$\begin{aligned}\frac{\partial}{\partial z_m} a_3\left(g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right) &= \sum_{i=1}^{\ell} R_i^N \frac{\partial}{\partial z_m}\left(e^{(i)} \cdot \mathcal{P}_{\mathcal{V}_N^{\perp}} \mathcal{T}_{\neg} g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right) \\ &= \sum_{i=1}^{\ell} R_i^N \left(\frac{\partial}{\partial z_m} \mathcal{T}_{\neg}^i g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right. \\ &\qquad \left. - \sum_{n=1}^{N}\left(\frac{\partial}{\partial z_m} \beta_n^{(N)}\left(g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right)\right) \mathcal{T}_{\neg}^i d_n\right) \end{aligned} \tag{7.66}$$

using (2.27), (3.18), (7.24), (7.35), (7.36), (7.38), (7.39), (7.40), (7.41) and (7.42),

$$\begin{aligned} b_1\left(g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right) &= \sum_{i=1}^{\ell}\left(\mathcal{T}_{\neg}^i g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right)^2 \\ &= \sum_{i=1}^{\ell}\left(\sum_{n=0}^{L}\sum_{j=-n}^{n} g_{n,j}^{(k,L)}\left(c, A\varepsilon^3\right) \sigma^{-n-1} Y_{n,j}\left(\eta^{(i)}\right)\right)^2 \end{aligned} \tag{7.67}$$

using (3.18),

$$\begin{aligned}\frac{\partial}{\partial z_m} b_1\left(g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right) &= \sum_{i=1}^{\ell} \frac{\partial}{\partial z_m}\left(\mathcal{T}_{\neg}^i g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right)^2 \\ &= \sum_{i=1}^{\ell} 2\left(\mathcal{T}_{\neg}^i g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right) \\ &\qquad \times\left(\sum_{n=0}^{L}\sum_{j=-n}^{n}\left(\frac{\partial}{\partial z_m} g_{n,j}^{(k,L)}\left(c, A\varepsilon^3\right)\right) \sigma^{-n-1} Y_{n,j}\left(\eta^{(i)}\right)\right) \end{aligned} \tag{7.68}$$

using (3.18), (7.24), (7.35) and (7.36),

$$\begin{aligned} b_3\left(g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right) &= \sum_{i=1}^{\ell}\left(e^{(i)} \cdot \mathcal{P}_{\mathcal{V}_N^{\perp}} \mathcal{T}_{\neg} g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right)^2 \\ &= \sum_{i=1}^{\ell}\left(\mathcal{T}_{\neg}^i g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right) - \sum_{n=1}^{N} \beta_n^{(N)}\left(g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right) \mathcal{T}_{\neg}^i d_n\right)^2 \end{aligned} \tag{7.69}$$

using (2.27), (3.18), Theorem 4.4.3, (7.37), (7.41), (7.42) and, at last,

$$\begin{aligned}
&\frac{\partial}{\partial z_m} b_3\left(g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right) \\
&= \sum_{i=1}^{\ell} \frac{\partial}{\partial z_m}\left(e^{(i)} \cdot \mathcal{P}_{\mathcal{V}_N^{\perp}} \mathcal{T}_{\neg} g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right)^2 \\
&= \sum_{i=1}^{\ell} 2\left(e^{(i)} \cdot \mathcal{P}_{\mathcal{V}_N^{\perp}} \mathcal{T}_{\neg} g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right) \\
&\quad \times \left(\frac{\partial}{\partial z_m} \mathcal{T}_{\neg}^{i} g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right) - \sum_{n=1}^{N}\left(\frac{\partial}{\partial z_m} \beta_n^{(N)}\left(g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right)\right) \mathcal{T}_{\neg}^{i} d_n\right)
\end{aligned} \tag{7.70}$$

using (2.27), (3.18), Theorem 4.4.3, (7.24), (7.35), (7.36), (7.37), (7.38), (7.39), (7.40), (7.41) and (7.42).

7.2.4. Inner products of linear combinations of dictionary elements

Next, we consider the terms $a_2(d(z))$, $a_4(d(z))$, $b_4(d(z))$ and $b_5(d(z))$ from Definition 7.1.2 and Lemma 7.1.5 and for $z \in \mathbb{R}^{\delta_z}$ (confer (7.19)) and their derivatives. We see that the terms $a_2(d(z))$ and $b_4(d(z))$ have a similar structure and, thus, can be considered at once. Similarly, we can generalize the terms $a_4(d(z))$ and $b_5(d(z))$. For both generalizations, we define a general linear combination of dictionary elements by

$$c_N \coloneqq \sum_{n=1}^{N} \gamma_n(d(z)) d_n.$$

Hence, in particular, we have

$$\gamma_n(d(z)) \equiv \alpha_n$$

if $c_N = f_N = f_N^{(N)}$ as well as

$$\gamma_n(d(z)) = \beta_n^{(N)}(d(z))$$

if $c_N = b_N^{(N)}$. Then the terms $a_2(d(z))$ and $b_4(d(z))$ are particular cases of

$$\langle c_N, d\rangle_{\mathcal{H}_2(\Omega)}.$$

Similarly, we obtain the generalization

$$\langle c_N, \widetilde{c}_N\rangle_{\mathcal{H}_2(\Omega)} \tag{7.71}$$

with c_N not necessarily equal to $\widetilde{c}_N$ for the terms $a_4(d(z))$ and $b_5(d(z))$. We use the linearity of the inner product and obtain

$$\langle c_N, d(z)\rangle_{\mathcal{H}_2(\Omega)} = \sum_{n=1}^{N} \gamma_n(d(z))\langle d_n, d(z)\rangle_{\mathcal{H}_2(\Omega)} \tag{7.72}$$

as well as

$$\langle c_N, \widetilde{c}_N\rangle_{\mathcal{H}_2(\Omega)} = \sum_{n=1}^{N}\sum_{\widetilde{n}=1}^{N} \gamma_n(d(z))\widetilde{\gamma}_{\widetilde{n}}(d(z))\langle d_n, d_{\widetilde{n}}\rangle_{\mathcal{H}_2(\Omega)}. \tag{7.73}$$

These term can be computed via the inner products given in (3.19) to (3.28) in Section 3.5 as well as Theorem 4.4.3 if $c_N = b_n^{(N)}(d(z))$. Note that for an efficient implementation, one can utilize the Clenshaw algorithm (Theorem A.2.1 and Algorithm 8) for the truncated series as well as Algorithm 9 for the associated Legendre functions. For (7.72), we have explicitly the formulas (7.74) to (7.76) where we abbreviate $A = A(\alpha, \beta, \gamma)$.

$$\langle c_N, K(x,\cdot)\rangle_{\mathcal{H}_2(\Omega)} = \sum_{n=1}^{N} \gamma_n (K(x,\cdot)) \times \begin{cases} \left(m+\frac{1}{2}\right)^4 |x|^m Y_{m,k}\left(\frac{x}{|x|}\right), & d_n = Y_{m,k} \\ \sum_{m=0}^{\infty}\left(m+\frac{1}{2}\right)^4 (|x||x'|)^m \frac{2m+1}{4\pi} P_m\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right), & d_n = K(x',\cdot) \\ \sum_{m=0}^{\infty}\left(m+\frac{1}{2}\right)^4 \left(|x'|^m - |x'|^{2m}\right)|x|^m \frac{2m+1}{4\pi} P_m\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right), & d_n = W(x',\cdot) \\ \sum_{l=0}^{L}\sum_{i=-l}^{l}\left(l+\frac{1}{2}\right)^4 g_{l,i}^{(k,L)}(c, A\varepsilon^3)\,|x|^l Y_{l,i}\left(\frac{x}{|x|}\right), & d_n = g^{(k,L)}\left((c, A\varepsilon^3),\cdot\right) \end{cases} \tag{7.74}$$

$$\langle c_N, W(x,\cdot)\rangle_{\mathcal{H}_2(\Omega)} = \sum_{n=1}^{N} \gamma_n (W(x,\cdot)) \times \begin{cases} \left(m+\frac{1}{2}\right)^4 \left(|x|^m - |x|^{2m}\right) Y_{m,k}\left(\frac{x}{|x|}\right), & d_n = Y_{m,k} \\ \sum_{m=0}^{\infty}\left(m+\frac{1}{2}\right)^4 \left(|x|^m - |x|^{2m}\right)|x'|^m \frac{2m+1}{4\pi} P_m\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right), & d_n = K(x',\cdot) \\ \sum_{m=0}^{\infty}\left(m+\frac{1}{2}\right)^4 \left(|x'|^m - |x'|^{2m}\right)\left(|x|^m - |x|^{2m}\right)\frac{2m+1}{4\pi} P_m\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right), & d_n = W(x',\cdot) \\ \sum_{l=0}^{L}\sum_{i=-l}^{l}\left(l+\frac{1}{2}\right)^4 g_{l,i}^{(k,L)}(c, A\varepsilon^3)\left(|x|^l - |x|^{2l}\right) Y_{l,i}\left(\frac{x}{|x|}\right), & d_n = g^{(k,L)}\left((c, A\varepsilon^3),\cdot\right) \end{cases} \tag{7.75}$$

$$\left\langle c_N, g^{(o,L)}\left(c, A\varepsilon^3, \cdot\right)\right\rangle_{\mathcal{H}_2(\Omega)} = \sum_{n=1}^{N} \gamma_n g^{(o,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)$$
$$\times \begin{cases} \left(m+\frac{1}{2}\right)^2 g_{m,k}^{(o,L)}\left(c, A\varepsilon^3\right), & d_n = Y_{m,k},\ m \le L \\ 0, & d_n = Y_{m,k},\ m > L \\ \sum_{l=0}^{L}\sum_{i=-l}^{l}\left(l+\frac{1}{2}\right)^4 g_{l,i}^{(o,L)}\left(c, A\varepsilon^3\right)|x'|^l Y_{l,i}\left(\frac{x'}{|x'|}\right), & d_n = K(x', \cdot) \\ \sum_{l=0}^{L}\sum_{i=-l}^{l}\left(l+\frac{1}{2}\right)^4 g_{l,i}^{(o,L)}\left(c, A\varepsilon^3\right)\left(|x'|^l - |x'|^{2l}\right) Y_{l,i}\left(\frac{x'}{|x'|}\right), & d_n = W(x', \cdot) \\ \sum_{l=0}^{L}\sum_{i=-l}^{l}\left(l+\frac{1}{2}\right)^4 g_{l,i}^{(o,L)}\left(c, A\varepsilon^3\right) g_{l,i}^{(k,L)}\left(c', A'\varepsilon^3\right), & d_n = g^{(k,L)}\left(\left(c', A'\varepsilon^3\right), \cdot\right) \end{cases} \tag{7.76}$$

As the dependence on $d(z)$ in the term (7.71) is only given in the coefficients $\gamma(d(z))$, we abstain from repeating all possible combinations of inner products and once again refer to (3.19) to (3.28).
Next, we consider the partial derivatives of (7.72) and (7.73):

$$\begin{aligned}
&\frac{\partial}{\partial z_j}\langle c_N, d(z)\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \sum_{n=1}^{N} \frac{\partial}{\partial z_j}\left(\gamma_n(d(z))\langle d_n, d(z)\rangle_{\mathcal{H}_2(\Omega)}\right) \\
&\quad = \sum_{n=1}^{N}\left(\left(\frac{\partial}{\partial z_j}\gamma_n(d(z))\right)\langle d_n, d(z)\rangle_{\mathcal{H}_2(\Omega)} + \gamma_n(d(z))\frac{\partial}{\partial z_j}\langle d_n, d(z)\rangle_{\mathcal{H}_2(\Omega)}\right)
\end{aligned} \tag{7.77}$$

as well as

$$\begin{aligned}
&\frac{\partial}{\partial z_j}\langle c_N, \widetilde{c}_N\rangle_{\mathcal{H}_2(\Omega)} \\
&= \sum_{n=1}^{N}\sum_{\widetilde{n}=1}^{N}\left(\frac{\partial}{\partial z_j}\left(\gamma_n(d(z))\widetilde{\gamma}_{\widetilde{n}}(d(z))\right)\right)\langle d_n, d_{\widetilde{n}}\rangle_{\mathcal{H}_2(\Omega)} \\
&= \sum_{n=1}^{N}\sum_{\widetilde{n}=1}^{N}\left(\left(\frac{\partial}{\partial z_j}\gamma_n(d(z))\right)\widetilde{\gamma}_{\widetilde{n}}(d(z)) + \gamma_n(d(z))\frac{\partial}{\partial z_j}\widetilde{\gamma}_{\widetilde{n}}(d(z))\right)\langle d_n, d_{\widetilde{n}}\rangle_{\mathcal{H}_2(\Omega)}.
\end{aligned} \tag{7.78}$$

Hence, we see that the derivatives (7.77) and (7.78) only depend on

$$\begin{aligned}
\frac{\partial}{\partial z_j}\gamma_n(d(z)) &= \left\{\begin{array}{cc} \frac{\partial}{\partial z_j}\alpha_n, & c_N = f_N = f_N^{(N)} \\ \frac{\partial}{\partial z_j}\beta_n^{(N)}(d(z)), & c_N = b_N^{(N)} \end{array}\right\} \\
&= \left\{\begin{array}{cc} 0, & c_N = f_N = f_N^{(N)} \\ \frac{\partial}{\partial z_j}\beta_n^{(N)}(d(z)), & c_N = b_N^{(N)} \end{array}\right\}
\end{aligned} \tag{7.79}$$

and

$$\frac{\partial}{\partial z_j}\langle d_n, d(z)\rangle_{\mathcal{H}_2(\Omega)}. \tag{7.80}$$

The derivatives of the projection coefficients $\beta_n^{(N)}(d(z))$, $n = 1, ..., N$, have already been discussed in (7.39) and (7.40). With these formulas as well as (3.19) to (3.28), the derivative of (7.73) is clear.
Thus, it remains to discuss the derivative of (7.72) in detail. In particular, we have to discuss the derivative of the $\mathcal{H}_2(\Omega)$-inner product, see (7.80). We first consider these terms for a Slepian functions $d(z(c, \alpha, \beta, \gamma)) = g^{(k,L)}((c, A(\alpha, \beta, \gamma)\varepsilon^3), \cdot)$. In

(3.22), (3.25), (3.27) and (3.28), we see that, independent of d_n, the inner product is always a finite sum and the derivative acts again only on the Fourier coefficients of the Slepian function. Thus, with (7.23) and (7.24), we obtain the respective derivatives.
If $d(x)$ (again, with $x = z \in \mathbb{R}^3$) is an Abel–Poisson low or band pass filter, we have to divide the next considerations with respect to the type of d_n. If d_n is a spherical harmonic or a Slepian function, we see in (3.20), (3.21), (3.25) and (3.27) that the inner product is again a finite sum and the derivative acts on terms of the form

$$\frac{\partial}{\partial x_j}|x|^{mn}Y_{n,j}\left(\frac{x}{|x|}\right) \tag{7.81}$$

for $m,\ n \in \mathbb{N}$. We consider the whole gradient at once. This will make the next investigations easier. That means, instead of (7.81), we consider

$$\nabla\left(|x|^{mn}Y_{n,j}\left(\frac{x}{|x|}\right)\right).$$

With the use of (2.5), we obtain

$$\begin{aligned}
&\nabla_x\left(|x|^{mn}Y_{n,j}\left(\frac{x}{|x|}\right)\right) = \nabla_{x(r,\varphi,t)}\left(r^{mn}Y_{n,j}\left(\frac{x(r,\varphi,t)}{r}\right)\right)\\
&= \left(\varepsilon^r\frac{\partial}{\partial r} + \frac{1}{r}\left(\varepsilon^\varphi\frac{1}{\sqrt{1-t^2}}\frac{\partial}{\partial\varphi} + \varepsilon^t\sqrt{1-t^2}\frac{\partial}{\partial t}\right)\right)\left(r^{mn}Y_{n,j}\left(\frac{x(r,\varphi,t)}{r}\right)\right)\\
&= \varepsilon^r\left(\frac{\partial}{\partial r}\left(r^{mn}Y_{n,j}\left(\frac{x(r,\varphi,t)}{r}\right)\right)\right) + \frac{1}{r}\varepsilon^\varphi\frac{1}{\sqrt{1-t^2}}\left(\frac{\partial}{\partial\varphi}\left(r^{mn}Y_{n,j}\left(\frac{x(r,\varphi,t)}{r}\right)\right)\right)\\
&\quad + \frac{1}{r}\varepsilon^t\sqrt{1-t^2}\left(\frac{\partial}{\partial t}\left(r^{mn}Y_{n,j}\left(\frac{x(r,\varphi,t)}{r}\right)\right)\right)\\
&= \varepsilon^r\left(\frac{\partial}{\partial r}r^{mn}\right)Y_{n,j}\left(\frac{x(r,\varphi,t)}{r}\right) + r^{mn-1}\varepsilon^\varphi\frac{1}{\sqrt{1-t^2}}\frac{\partial}{\partial\varphi}Y_{n,j}\left(\frac{x(r,\varphi,t)}{r}\right)\\
&\quad + r^{mn-1}\varepsilon^t\sqrt{1-t^2}\frac{\partial}{\partial t}Y_{n,j}\left(\frac{x(r,\varphi,t)}{r}\right)\\
&= r^{mn-1}\left(\varepsilon^r mnY_{n,j}\left(\frac{x(r,\varphi,t)}{r}\right) + \varepsilon^\varphi\frac{j}{\sqrt{1-t^2}}Y_{n,-j}\left(\frac{x(r,\varphi,t)}{r}\right)\right.\\
&\qquad \left. + \varepsilon^t\sqrt{1-t^2}\sqrt{\frac{2n+1}{4\pi}\frac{(n-|j|)!}{(n+|j|)!}}P'_{n,|j|}(t)\begin{cases}\sqrt{2}\cos(j\varphi), & j<0\\ 1, & j=0\\ \sqrt{2}\sin(j\varphi), & j>0\end{cases}\right)
\end{aligned} \tag{7.82}$$

where the latter equation holds due to (2.17). We have to take a closer look at (7.82) regarding possible singularities in $t = \pm 1$ as well as $r = 0$. We begin with the case $t = \pm 1$. In that respect, the term (7.82) contains two probably problematic

terms:

$$\frac{j}{\sqrt{1-t^2}} Y_{n,-j}\left(\frac{x(r,\varphi,t)}{r}\right) \tag{7.83}$$

and

$$\sqrt{1-t^2}\sqrt{\frac{2n+1}{4\pi}\frac{(n-|j|)!}{(n+|j|)!}} P'_{n,|j|}(t). \tag{7.84}$$

We consider (7.83) first. If $j = 0$, this is a removable singularity with a zero value. In the case $j \neq 0$, using the definition of the fully normalized spherical harmonics (see (2.14)), the term reformulates to

$$\begin{aligned}
&\frac{j}{\sqrt{1-t^2}} Y_{n,-j}\left(\frac{x(r,\varphi,t)}{r}\right) \\
&= \frac{j}{\sqrt{1-t^2}}\sqrt{\frac{2n+1}{4\pi}\frac{(n-|j|)!}{(n+|j|)!}} P_{n,|j|}(t) \begin{cases} \sqrt{2}\cos(j\varphi), & j<0 \\ 1, & j=0 \\ \sqrt{2}\sin(j\varphi), & j>0 \end{cases} \\
&= \frac{j}{\sqrt{1-t^2}}\sqrt{\frac{2n+1}{4\pi}\frac{(n-|j|)!}{(n+|j|)!}} \left(1-t^2\right)^{|j|/2} \left(\frac{d^{|j|}}{dt^{|j|}} P_n(t)\right) \begin{cases} \sqrt{2}\cos(j\varphi), & j<0 \\ 1, & j=0 \\ \sqrt{2}\sin(j\varphi), & j>0 \end{cases} \\
&= j\sqrt{\frac{2n+1}{4\pi}\frac{(n-|j|)!}{(n+|j|)!}} \left(1-t^2\right)^{(|j|-1)/2} \left(\frac{d^{|j|}}{dt^{|j|}} P_n(t)\right) \begin{cases} \sqrt{2}\cos(j\varphi), & j<0 \\ 1, & j=0 \\ \sqrt{2}\sin(j\varphi), & j>0 \end{cases}.
\end{aligned} \tag{7.85}$$

Obviously, a problem exists only if

$$\frac{|j|-1}{2} < 0$$

which is equivalent to $j = 0$. As this was excluded in this case, the term (7.83) has no singularity and can be used in (7.82). Note that an efficient implementation of (7.83) is based, for instance, on Appendix A, Algorithm 14. For the second term (7.84), we have

$$\begin{aligned}
&\sqrt{1-t^2} p_{n,j} P'_{n,|j|}(t) \\
&= \sqrt{1-t^2} p_{n,j} \frac{d}{dt} P_{n,|j|}(t) \\
&= \sqrt{1-t^2} p_{n,j} \frac{d}{dt}\left(\left(1-t^2\right)^{|j|/2} \frac{d^{|j|}}{dt^{|j|}} P_n(t)\right) \\
&= \sqrt{1-t^2} p_{n,j}\left(\frac{|j|}{2}\left(1-t^2\right)^{(|j|/2)-1}(-2t)\frac{d^{|j|}}{dt^{|j|}} P_n(t) + \left(1-t^2\right)^{|j|/2} \frac{d^{|j|+1}}{dt^{|j|+1}} P_n(t)\right)
\end{aligned}$$

$$= p_{n,j}\left(-|j|t\left(1-t^2\right)^{(|j|-1)/2}\frac{d^{|j|}}{dt^{|j|}}P_n(t)+\left(1-t^2\right)^{(|j|+1)/2}\frac{d^{|j|+1}}{dt^{|j|+1}}P_n(t)\right) \quad (7.86)$$

with the abbreviation

$$p_{n,j} := \sqrt{\frac{2n+1}{4\pi}\frac{(n-|j|)!}{(n+|j|)!}}. \quad (7.87)$$

Again, the only possibly problematic term is

$$|j|\left(1-t^2\right)^{(|j|-1)/2}.$$

However, for $j = 0$, we have again a removable singularity. If $j > 0$, the exponents are non-negative. Thus, also (7.84) contains no singularity. An efficient implementation of (7.84) can be based, for instance, on Appendix A, Algorithm 15. Next, we have to discuss (7.82) for $r = 0$. First, we note that, for $mn > 1$, the term vanishes due to the factor r^{mn-1} in front and because the polynomial spherical harmonics are bounded. Further, we see that, in this case, the values of φ and t in 0 can, thus, be arbitrarily chosen. Left to discuss are $n = 0$ with $m \in \mathbb{N}$ as well as $n = 1$ with $m = 1$. In the former case, we also have $j = 0$. Hence, obviously, the first two summands in the brackets vanish. The third summand vanishes because $P'_{0,0}(t) = P'_0(t) = 0$, i. e. the derivative of the Legendre polynomial of degree 0 vanishes. Thus, we have a removable singularity for $x = 0$ and $n = 0$. Again, φ and t are arbitrary in this case. At last, we consider (7.82) for $x = 0$, $n = 1$ and $m = 1$. We see that the factor r^{mn-1} in front is constant 1. Hence, again, due to the boundedness of the spherical harmonics, the term is well-defined in this case as well. In particular, we see that the sum in the brackets rearranges to

$$\begin{aligned}\varepsilon^r Y_{1,j}\left(\frac{x(r,\varphi,t)}{r}\right)+\varepsilon^\varphi\frac{j}{\sqrt{1-t^2}}Y_{1,-j}\left(\frac{x(r,\varphi,t)}{r}\right)\\ +\varepsilon^t\sqrt{1-t^2}\sqrt{\frac{3}{4\pi}\frac{(1-|j|)!}{(1+|j|)!}}P'_{1,|j|}(t)\begin{cases}\sqrt{2}\cos(j\varphi), & j<0\\ 1, & j=0\\ \sqrt{2}\sin(j\varphi), & j>0\end{cases}\end{aligned}$$

for $|j| \leq 1$. With (2.2), (2.3), (2.4) (all with $t = \cos(\theta)$), (2.13), (2.14),(7.87) (note $p_{n,j} = p_{n,-j}$) and (A.1) (in particular, $P_1(t) = t$, $P'_1(t) = 1$, $P''_1(t) = 0$), we obtain

for $|j| = 1$

$$
\begin{aligned}
&\begin{pmatrix} \sqrt{1-t^2}\cos(\varphi) \\ \sqrt{1-t^2}\sin(\varphi) \\ t \end{pmatrix} p_{1,\pm1}\sqrt{1-t^2}P_1'(t) \begin{cases} \sqrt{2}\cos(\varphi), & j=-1 \\ \sqrt{2}\sin(\varphi), & j=1 \end{cases} \\
&\quad \pm \begin{pmatrix} -\sin(\varphi) \\ \cos(\varphi) \\ 0 \end{pmatrix} \frac{1}{\sqrt{1-t^2}} p_{1,\mp1}\sqrt{1-t^2}P_1'(t) \begin{cases} \sqrt{2}\sin(\varphi), & j=-1 \\ \sqrt{2}\cos(\varphi), & j=1 \end{cases} \\
&\quad + \begin{pmatrix} -t\cos(\varphi) \\ -t\sin(\varphi) \\ \sqrt{1-t^2} \end{pmatrix} \sqrt{1-t^2} p_{1,\pm1} \frac{\mathrm{d}}{\mathrm{d}t}\left(\sqrt{1-t^2}P_1'(t)\right) \begin{cases} \sqrt{2}\cos(\varphi), & j=-1 \\ \sqrt{2}\sin(\varphi), & j=1 \end{cases} \\
&= \begin{pmatrix} (1-t^2)\cos(\varphi) \\ (1-t^2)\sin(\varphi) \\ t\sqrt{1-t^2} \end{pmatrix} p_{1,\pm1} \begin{cases} \sqrt{2}\cos(\varphi), & j=-1 \\ \sqrt{2}\sin(\varphi), & j=1 \end{cases} \\
&\quad \pm \begin{pmatrix} -\sin(\varphi) \\ \cos(\varphi) \\ 0 \end{pmatrix} p_{1,\mp1} \begin{cases} \sqrt{2}\sin(\varphi), & j=-1 \\ \sqrt{2}\cos(\varphi), & j=1 \end{cases} \\
&\quad + \begin{pmatrix} -t\cos(\varphi) \\ -t\sin(\varphi) \\ \sqrt{1-t^2} \end{pmatrix} \sqrt{1-t^2} p_{1,\pm1} \left(-\frac{t}{\sqrt{1-t^2}}\right) \begin{cases} \sqrt{2}\cos(\varphi), & j=-1 \\ \sqrt{2}\sin(\varphi), & j=1 \end{cases} \\
&= \begin{pmatrix} (1-t^2)\cos(\varphi) \\ (1-t^2)\sin(\varphi) \\ t\sqrt{1-t^2} \end{pmatrix} p_{1,\pm1} \begin{cases} \sqrt{2}\cos(\varphi), & j=-1 \\ \sqrt{2}\sin(\varphi), & j=1 \end{cases} \\
&\quad \pm \begin{pmatrix} -\sin(\varphi) \\ \cos(\varphi) \\ 0 \end{pmatrix} p_{1,\pm1} \begin{cases} \sqrt{2}\sin(\varphi), & j=-1 \\ \sqrt{2}\cos(\varphi), & j=1 \end{cases} \\
&\quad - \begin{pmatrix} -t^2\cos(\varphi) \\ -t^2\sin(\varphi) \\ t\sqrt{1-t^2} \end{pmatrix} p_{1,\pm1} \begin{cases} \sqrt{2}\cos(\varphi), & j=-1 \\ \sqrt{2}\sin(\varphi), & j=1. \end{cases}
\end{aligned}
$$

That means, for $j = -1$, we have

$$
\begin{aligned}
&\sqrt{2}p_{1,-1} \begin{pmatrix} (1-t^2)\cos^2(\varphi) + \sin^2(\varphi) + t^2\cos^2(\varphi) \\ (1-t^2)\sin(\varphi)\cos(\varphi) - \cos(\varphi)\sin(\varphi) + t^2\sin(\varphi)\cos(\varphi) \\ t\sqrt{1-t^2}\cos(\varphi) - t\sqrt{1-t^2}\cos(\varphi) \end{pmatrix} \\
&\qquad = \sqrt{2}p_{1,-1}(1,0,0)^{\mathrm{T}}.
\end{aligned}
$$

Analogously, for $j = 1$, we obtain

$$\begin{aligned}\sqrt{2}p_{1,1} &\begin{pmatrix}(1-t^2)\cos(\varphi)\sin(\varphi) - \sin(\varphi)\cos(\varphi) + t^2\cos(\varphi)\sin(\varphi)\\ (1-t^2)\sin^2(\varphi) + \cos^2(\varphi) + t^2\sin^2(\varphi)\\ t\sqrt{1-t^2}\sin(\varphi) - t\sqrt{1-t^2}\sin(\varphi)\end{pmatrix}\\ &= \sqrt{2}p_{1,1}(0,1,0)^{\mathrm{T}}.\end{aligned}$$

At last, for $j = 0$, it similarly holds

$$\begin{aligned}&\begin{pmatrix}\sqrt{1-t^2}\cos(\varphi)\\ \sqrt{1-t^2}\sin(\varphi)\\ t\end{pmatrix} p_{1,0}P_1(t) + \begin{pmatrix}-t\cos(\varphi)\\ -t\sin(\varphi)\\ \sqrt{1-t^2}\end{pmatrix}\sqrt{1-t^2}p_{1,0}P_1'(t)\\ &= p_{1,0}\left(\begin{pmatrix}\sqrt{1-t^2}t\cos(\varphi)\\ \sqrt{1-t^2}t\sin(\varphi)\\ t^2\end{pmatrix} + \begin{pmatrix}-\sqrt{1-t^2}t\cos(\varphi)\\ -\sqrt{1-t^2}t\sin(\varphi)\\ 1-t^2\end{pmatrix}\right) = p_{1,0}\begin{pmatrix}0\\0\\1\end{pmatrix}.\end{aligned}$$

Thus, we see that also for $n = 1$ with $m = 1$ the choice of the particular φ and t is arbitrary if $x = 0$. All in all, we see that (7.82) is well-defined for $t = \pm 1$ as well as $x = 0$ and, in the latter case, the values are independent of a specific choice of φ and t.
It remains to consider (7.80) for the use in (7.72) if $d(x)$ and d_n is an Abel–Poisson low or band pass filter. In (3.23), (3.24) and (3.26), we see that such an $\mathcal{H}_2(\Omega)$-inner product is a series. Thus, first of all, we have to take into consideration the exchange of the limits. Using the Weierstrass M-test, we need to consider for $n \in \mathbb{N}_0$:

$$\begin{aligned}&\left|\nabla_x\left(A_n^2\left(|x|\,|x'|\right)^n\frac{2n+1}{4\pi}P_n\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right)\right)\right|\\ &\quad= \left|A_n^2\,|x'|^n\sum_{j=-n}^{n}Y_{n,j}\left(\frac{x'}{|x'|}\right)\nabla_x\left(|x|^nY_{n,j}\left(\frac{x}{|x|}\right)\right)\right|\\ &\quad= \left|A_n^2\,|x'|^n\sum_{j=-n}^{n}Y_{n,j}\left(\frac{x'}{|x'|}\right)\sqrt{n(2n+1)}|x|^{n-1}\widetilde{y}_{n,j}^{(2)}\left(\frac{x}{|x|}\right)\right|\\ &\quad\le A_n^2\sqrt{n(2n+1)}\,|x'|^n\sum_{j=-n}^{n}\left|Y_{n,j}\left(\frac{x'}{|x'|}\right)\right||x|^{n-1}\left|\widetilde{y}_{n,j}^{(2)}\left(\frac{x}{|x|}\right)\right|\\ &\quad\le A_n^2\sqrt{n(2n+1)}\,|x'|^n\sum_{j=-n}^{n}\sqrt{\frac{2n+1}{4\pi}}\sqrt{\frac{2n+1}{4\pi}}\\ &\quad\le A_n^2\sqrt{n(2n+1)}\frac{(2n+1)^2}{4\pi}\,|x'|^n\\ &\quad= O\left(n^5\right)|x'|^n,\end{aligned}$$

using $|x| < 1$, the gradient (note that $|x| = r$)

$$\begin{aligned}\nabla |x|^{mn} &= \left(\varepsilon^r \frac{\partial}{\partial r} + \frac{1}{r}\left(\varepsilon^\varphi \frac{1}{\sqrt{1-t^2}}\frac{\partial}{\partial \varphi} + \varepsilon^t \sqrt{1-t^2}\frac{\partial}{\partial t}\right)\right)|x|^{mn} \\ &= \varepsilon^r \frac{\partial}{\partial r}|x|^{mn} = mn\varepsilon^r |x|^{mn-1} = mnx|x|^{mn-2} \end{aligned} \tag{7.88}$$

for $m,\ n \in \mathbb{N}$, and the estimates

$$\left|Y_{m,k}\left(\frac{x}{|x|}\right)\right|^2 \leq \sum_{j=-n}^{n}\left(Y_{n,j}\left(\frac{x}{|x|}\right)\right)^2 = \frac{2n+1}{4\pi}$$

as well as

$$\left|\widetilde{y}^{(2)}_{m,k}\left(\frac{x}{|x|}\right)\right|^2 \leq \sum_{j=-n}^{n}\left|\widetilde{y}^{(2)}_{n,j}\left(\frac{x}{|x|}\right)\right|^2 = \frac{2n+1}{4\pi}$$

(see e. g. Freeden et al., 1998; Michel, 2013, 2020) where $\widetilde{y}^{(2)}_{m,k}$, $m,\ k \in \mathbb{N}_0$, $k = -m, ..., m$, are the Edmonds vector spherical harmonics of type 2 (see e. g. Edmonds, 1996; Michel, 2020). Note that, for these estimates we utilize Theorem 2.2.4 as well as $|P_n(t)| \leq 1$ for all $t \in [-1, 1]$ (see e. g. Freeden and Schreiner, 2009, p. 88). Note that (7.88) has a removable singularity for $x = 0$ as well as $n = 0$ or $n = 1$ and $m = 1$ and is singularity-free if $mn > 1$.
If two Abel–Poisson low pass filters are under investigation, this gives us the necessary upper bound as $|x'| < 1$ and fixed. In the other cases, we obtain similarly

$$\begin{aligned}&\left|\nabla_x\left(A_n^2\left(|x'|^n - |x'|^{2n}\right)|x|^n \frac{2n+1}{4\pi} P_n\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right)\right)\right| \\ &\quad \leq O\left(n^5\right)\left(|x'|^n - |x'|^{2n}\right),\end{aligned}$$

$$\begin{aligned}&\left|\nabla_x\left(A_n^2\left(|x|^n - |x|^{2n}\right)|x'|^n \frac{2n+1}{4\pi} P_n\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right)\right)\right| \\ &\quad = \left|A_n^2 |x'|^n \sum_{j=-n}^{n} Y_{n,j}\left(\frac{x'}{|x'|}\right)\nabla_x\left(\left(|x|^n - |x|^{2n}\right)Y_{n,j}\left(\frac{x}{|x|}\right)\right)\right| \\ &\quad = \Bigg|A_n^2 |x'|^n \Bigg(\sum_{j=-n}^{n} Y_{n,j}\left(\frac{x'}{|x'|}\right)\sqrt{n(2n+1)}|x|^{n-1}\widetilde{y}^{(2)}_{n,j}\left(\frac{x}{|x|}\right) \\ &\qquad\qquad - \sum_{j=-n}^{n} Y_{n,j}\left(\frac{x'}{|x'|}\right)\sqrt{n(2n+1)}|x|^{2n-1}\widetilde{y}^{(2)}_{n,j}\left(\frac{x}{|x|}\right) \\ &\qquad\qquad - \sum_{j=-n}^{n} Y_{n,j}\left(\frac{x'}{|x'|}\right) nx|x|^{2n-2} Y_{n,j}\left(\frac{x}{|x|}\right)\Bigg)\Bigg|\end{aligned}$$

$$
\begin{aligned}
&\leq A_n^2 \left|x'\right|^n \left(\sum_{j=-n}^{n} \left| Y_{n,j}\left(\frac{x'}{|x'|}\right)\right| \sqrt{n(2n+1)}|x|^{n-1} \left|\widetilde{y}_{n,j}^{(2)}\left(\frac{x}{|x|}\right)\right| \right. \\
&\qquad + \sum_{j=-n}^{n} \left| Y_{n,j}\left(\frac{x'}{|x'|}\right)\right| \sqrt{n(2n+1)}|x|^{2n-1} \left|\widetilde{y}_{n,j}^{(2)}\left(\frac{x}{|x|}\right)\right| \\
&\qquad \left. + \sum_{j=-n}^{n} \left| Y_{n,j}\left(\frac{x'}{|x'|}\right)\right| n|x|^{2n-1} \left| Y_{n,j}\left(\frac{x}{|x|}\right)\right| \right) \\
&\leq A_n^2 \left|x'\right|^n \left(\frac{(2n+1)^2}{4\pi}\sqrt{n(2n+1)} + \frac{(2n+1)^2}{4\pi}\sqrt{n(2n+1)} + \frac{(2n+1)^2}{4\pi} n \right) \\
&= A_n^2 \left|x'\right|^n \frac{(2n+1)^2}{4\pi}\left(2\sqrt{n(2n+1)} + n\right) = O\left(n^5\right)\left|x'\right|^n,
\end{aligned}
$$

$$
\begin{aligned}
&\left| \nabla_x \left(A_n^2 \left(\left|x'\right|^n - \left|x'\right|^{2n} \right) \left(|x|^n - |x|^{2n} \right) \frac{2n+1}{4\pi} P_n\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right)\right)\right| \\
&\quad \leq O\left(n^5\right)\left(\left|x'\right|^n - \left|x'\right|^{2n}\right).
\end{aligned}
$$

Hence, also in the cases of one Abel–Poisson low pass filter and one Abel–Poisson band pass filter as well as two Abel–Poisson band pass filters, the geometric series is a majorant series and we can exchange the limits of differentiation with the series. Note that these considerations include the case $\|d(x)\|_{\mathcal{H}_2(\Omega)}$ as used in $b_2(d(z))$ (see Section 7.2.5).
We see that the differentiation of the inner products (3.23), (3.24) and (3.26) acts only on terms of the form

$$
\frac{\partial}{\partial x_j}\left(|x|^{mn} P_n\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{x}{|x|}\right)\right),\ m\in\mathbb{N}.
$$

Again, we consider

$$
\nabla\left(|x|^{mn} P_n\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{x}{|x|}\right)\right)
$$

instead. This formulation yields

$$
\begin{aligned}
\nabla_x \left(|x|^{mn} P_n\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{x}{|x|}\right)\right) &= \nabla_{x(r,\varphi,t)}\left(r^{mn} P_n\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\varepsilon^r\right)\right) \\
&= \left(\varepsilon^r\frac{\partial}{\partial r} + \frac{1}{r}\left(\varepsilon^\varphi \frac{1}{\sqrt{1-t^2}}\frac{\partial}{\partial\varphi} + \varepsilon^t\sqrt{1-t^2}\frac{\partial}{\partial t}\right)\right)\left(r^{mn} P_n\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\varepsilon^r\right)\right)
\end{aligned}
$$

$$
\begin{aligned}
&= \varepsilon^r \left(\frac{\partial}{\partial r} r^{mn}\right) P_n\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\varepsilon^r\right) \\
&\quad + \frac{1}{r}\left(\varepsilon^\varphi \frac{1}{\sqrt{1-t^2}}\frac{\partial}{\partial\varphi} + \varepsilon^t\sqrt{1-t^2}\frac{\partial}{\partial t}\right)\left(r^{mn} P_n\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\varepsilon^r\right)\right) \\
&= \varepsilon^r mn r^{mn-1} P_n\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\varepsilon^r\right) \\
&\quad + r^{mn-1}\left(\varepsilon^\varphi \frac{1}{\sqrt{1-t^2}} P_n'\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\varepsilon^r\right)\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{\partial}{\partial\varphi}\varepsilon^r\right)\right. \\
&\qquad \left. + \varepsilon^t\sqrt{1-t^2} P_n'\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\varepsilon^r\right)\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{\partial}{\partial t}\varepsilon^r\right)\right) \\
&= \varepsilon^r mn |x|^{mn-1} P_n\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{x}{|x|}\right) \\
&\quad + |x|^{mn-1}\left(\varepsilon^\varphi \frac{1}{\sqrt{1-t^2}} P_n'\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{x}{|x|}\right)\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\sqrt{1-t^2}\varepsilon^\varphi\right)\right. \\
&\qquad \left. + \varepsilon^t\sqrt{1-t^2} P_n'\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{x}{|x|}\right)\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{1}{\sqrt{1-t^2}}\varepsilon^t\right)\right) \\
&= |x|^{mn-1}\left(mn\varepsilon^r P_n\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{x}{|x|}\right)\right. \\
&\quad \left. + \varepsilon^\varphi P_n'\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{x}{|x|}\right)\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\varepsilon^\varphi\right) + \varepsilon^t P_n'\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{x}{|x|}\right)\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\varepsilon^t\right)\right)
\end{aligned}
\tag{7.89}
$$

For the Legendre polynomials and their derivatives in (7.89) , an efficient implementation is given, for instance, in Appendix A, (A.1) and (A.2).
We have to discuss the well-definedness of (7.89) for $x = 0$ or $x^{(i)} = 0$, i. e. $|x| = 0$ or $|x^{(i)}| = 0$. First of all, we note that the term originates from (3.23), (3.24) and (3.26) and, hence, we discuss the well-definedness of (7.89) in the form

$$
\begin{aligned}
&\left|x^{(i)}\right|^{m'n} |x|^{mn-1}\left(mn\varepsilon^r P_n\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{x}{|x|}\right)\right. \\
&\qquad + \varepsilon^\varphi P_n'\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{x}{|x|}\right)\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\varepsilon^\varphi\right) \\
&\qquad \left. + \varepsilon^t P_n'\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\frac{x}{|x|}\right)\left(\frac{x^{(i)}}{\left|x^{(i)}\right|}\cdot\varepsilon^t\right)\right)
\end{aligned}
\tag{7.90}
$$

for n, m and $m' \in \mathbb{N}$. In particular, for $n = 0$, we compute the derivative of a

constant function. Thus, if $x = 0$ or $x^{(i)} = 0$, we have a removable singularity in this case. We consider $n \geq 1$ in the sequel and note that, for $n \in \mathbb{N}$ the Legendre polynomials can be written as

$$P_n\left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \frac{x}{|x|}\right) = \sum_{\nu=0}^{n} c_\nu \left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \frac{x}{|x|}\right)^\nu$$

and, thus, their derivatives as

$$P_n'\left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \frac{x}{|x|}\right) = \sum_{\nu=1}^{n} \nu c_\nu \left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \frac{x}{|x|}\right)^{\nu-1}.$$

Then we have for (7.90)

$$\begin{aligned}
&\left|x^{(i)}\right|^{m'n} |x|^{mn-1} \left(mn\varepsilon^r \sum_{\nu=0}^{n} c_\nu \left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \frac{x}{|x|}\right)^\nu \right. \\
&\qquad + \varepsilon^\varphi \left(\sum_{\nu=1}^{n} \nu c_\nu \left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \frac{x}{|x|}\right)^{\nu-1}\right) \left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \varepsilon^\varphi\right) \\
&\qquad \left. + \varepsilon^t \left(\sum_{\nu=1}^{n} \nu c_\nu \left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \frac{x}{|x|}\right)^{\nu-1}\right) \left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \varepsilon^t\right) \right) \\
&= mn\varepsilon^r \sum_{\nu=0}^{n} c_\nu \left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \frac{x}{|x|}\right)^\nu \left|x^{(i)}\right|^{m'n} |x|^{mn-1} \\
&\quad + \varepsilon^\varphi \left(x^{(i)} \cdot \varepsilon^\varphi\right) \sum_{\nu=1}^{n} \nu c_\nu \left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \frac{x}{|x|}\right)^{\nu-1} \left|x^{(i)}\right|^{m'n-1} |x|^{mn-1} \\
&\quad + \varepsilon^t \left(x^{(i)} \cdot \varepsilon^t\right) \sum_{\nu=1}^{n} \nu c_\nu \left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \frac{x}{|x|}\right)^{\nu-1} \left|x^{(i)}\right|^{m'n-1} |x|^{mn-1} \\
&= mn\varepsilon^r \sum_{\nu=0}^{n} c_\nu \left(x^{(i)} \cdot x\right)^\nu \left|x^{(i)}\right|^{m'n-\nu} |x|^{mn-1-\nu} \\
&\quad + \varepsilon^\varphi \left(x^{(i)} \cdot \varepsilon^\varphi\right) \sum_{\nu=1}^{n} \nu c_\nu \left(x^{(i)} \cdot x\right)^{\nu-1} \left|x^{(i)}\right|^{m'n-\nu} |x|^{mn-\nu} \\
&\quad + \varepsilon^t \left(x^{(i)} \cdot \varepsilon^t\right) \sum_{\nu=1}^{n} \nu c_\nu \left(x^{(i)} \cdot x\right)^{\nu-1} \left|x^{(i)}\right|^{m'n-\nu} |x|^{mn-\nu}
\end{aligned}$$

$$
\begin{aligned}
&= mn\varepsilon^r \sum_{\nu=0}^{n} c_\nu \left(\cos\left(\sphericalangle\left(x^{(i)}, x\right)\right)\right)^\nu \left|x^{(i)}\right|^{m'n} |x|^{mn-1} \\
&\quad + \varepsilon^\varphi \left(x^{(i)} \cdot \varepsilon^\varphi\right) \sum_{\nu=1}^{n} \nu c_\nu \left(x^{(i)} \cdot x\right)^{\nu-1} \left|x^{(i)}\right|^{m'n-\nu} |x|^{mn-\nu} \\
&\quad + \varepsilon^t \left(x^{(i)} \cdot \varepsilon^t\right) \sum_{\nu=1}^{n} \nu c_\nu \left(x^{(i)} \cdot x\right)^{\nu-1} \left|x^{(i)}\right|^{m'n-\nu} |x|^{mn-\nu}.
\end{aligned}
$$

We immediately notice that, if $n > 1$, all terms are bounded for $x = 0$ or $x^{(i)} = 0$ and all $m,\ m' \in \mathbb{N}$. In particular, because $mn - 1 > 0$ and $m'n - 1 > 0$, the gradient vanishes for $x = 0$ or $x^{(i)} = 0$. At last, we discuss the case $n = 1$ for $x = 0$ or $x^{(i)} = 0$ and all $m,\ m' \in \mathbb{N}$. For $x^{(i)} = 0$, the term also vanishes for all $n,\ m,\ m' \in \mathbb{N}$ and all $x \in \mathring{\mathbb{B}}$. For $x^{(i)} \neq 0$, the term vanishes in the case of $x = 0$ if $m > 1$. Thus, we remain with the case $x^{(i)} \neq 0$, $x = 0$ and $n = m = 1$ and $m' \in \mathbb{N}$. For these values of n and m, we consider again the formulation (7.90) for $x \neq 0$ and $x^{(i)} \neq 0$:

$$
\left|x^{(i)}\right|^{m'} \left(\varepsilon^r \left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \varepsilon^r\right) + \varepsilon^\varphi \left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \varepsilon^\varphi\right) + \varepsilon^t \left(\frac{x^{(i)}}{\left|x^{(i)}\right|} \cdot \varepsilon^t\right)\right) \tag{7.91}
$$

with $P_1(t) = t$ and $P_1'(t) = 1$ for $t \in [-1, 1]$ and $x/|x| = \varepsilon^r$. Obviously, the sum in the brackets is a decomposition of $x^{(i)}/|x^{(i)}|$ in the local orthonormal basis $\varepsilon^r,\ \varepsilon^\varphi,\ \varepsilon^t$. Thus, (7.91) equals

$$
\left|x^{(i)}\right|^{m'} \frac{x^{(i)}}{\left|x^{(i)}\right|} = \left|x^{(i)}\right|^{m'-1} x^{(i)}.
$$

That means, if $n = m = 1$, the gradient is independent of x and, hence, well-defined for $x = 0$. Summarized, we have seen that (7.89) in the formulation (7.90) (and, thus, in its use in the gradients of (3.23), (3.24) and (3.26)) is well-defined for $x = 0$ or $x^{(i)} = 0$. Furthermore, it is independent of a specific choice of φ and t for x as well as $x^{(i)}$.
All in all, for the derivatives of (7.72), we obtain the formulas (7.92) to (7.94) which can be computed via (3.19) to (3.28), (7.79), (7.82) and (7.89). Note that we again abbreviate $A = A(\alpha, \beta, \gamma)$.

$$\nabla\langle c_N, K(x,\cdot)\rangle_{\mathcal{H}_2(\Omega)} = \sum_{n=1}^{N} \left(\nabla\gamma_n(K(x,\cdot))\right)\langle d_n, K(x,\cdot)\rangle_{\mathcal{H}_2(\Omega)} + \sum_{n=1}^{N}\gamma_n(K(x,\cdot))$$

$$\times \begin{cases} \left(m+\frac{1}{2}\right)^4 \nabla\left(|x|^m Y_{m,k}\left(\frac{x}{|x|}\right)\right), & d_n = Y_{m,k} \\ \sum_{m=0}^{\infty}\left(m+\frac{1}{2}\right)^4 \frac{2m+1}{4\pi}\left(|x'|\right)^m \nabla\left(|x|^m P_m\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right)\right), & d_n = K(x',\cdot) \\ \sum_{m=0}^{\infty}\left(m+\frac{1}{2}\right)^4 \frac{2n+1}{4\pi}\left(|x'|^m - |x'|^{2m}\right)\nabla\left(|x|^m P_m\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right)\right), & d_n = W(x',\cdot) \\ \sum_{l=0}^{L}\sum_{i=-l}^{l}\left(l+\frac{1}{2}\right)^4 g_{l,i}^{(k,L)}\left(c, A\varepsilon^3\right)\nabla\left(|x|^l Y_{l,i}\left(\frac{x}{|x|}\right)\right), & d_n = g^{(k,L)}\left(\left(c, A\varepsilon^3\right),\cdot\right) \end{cases} \tag{7.92}$$

$$\nabla\langle c_N, W(x,\cdot)\rangle_{\mathcal{H}_2(\Omega)} = \sum_{n=1}^{N} \left(\nabla\gamma_n(W(x,\cdot))\right)\langle d_n, W(x,\cdot)\rangle_{\mathcal{H}_2(\Omega)} + \sum_{n=1}^{N}\gamma_n(W(x,\cdot))$$

$$\times \begin{cases} \left(m+\frac{1}{2}\right)^4 \nabla\left(\left(|x|^m - |x|^{2m}\right) Y_{m,k}\left(\frac{x}{|x|}\right)\right), & d_n = Y_{m,k} \\ \sum_{m=0}^{\infty}\left(m+\frac{1}{2}\right)^4 \frac{2m+1}{4\pi}|x'|^m \nabla\left(\left(|x|^m - |x|^{2m}\right) P_m\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right)\right), & d_n = K(x',\cdot) \\ \sum_{m=0}^{\infty}\left(m+\frac{1}{2}\right)^4 \frac{2m+1}{4\pi}\left(|x'|^m - |x'|^{2m}\right)\nabla\left(\left(|x|^m - |x|^{2m}\right) P_m\left(\frac{x}{|x|}\cdot\frac{x'}{|x'|}\right)\right), & d_n = W(x',\cdot) \\ \sum_{l=0}^{L}\sum_{i=-l}^{l}\left(l+\frac{1}{2}\right)^4 g_{l,i}^{(k,L)}\left(c, A\varepsilon^3\right)\nabla\left(\left(|x|^l - |x|^{2l}\right) Y_{l,i}\left(\frac{x}{|x|}\right)\right), & d_n = g^{(k,L)}\left(\left(c, A\varepsilon^3\right),\cdot\right) \end{cases} \tag{7.93}$$

$$
\begin{aligned}
&\nabla \left\langle c_N, g^{(o,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right\rangle_{\mathcal{H}_2(\Omega)} \\
&= \sum_{n=1}^{N} \left(\nabla \gamma_n \left(g^{(o,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right)\right) \left\langle d_n, g^{(o,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right\rangle_{\mathcal{H}_2(\Omega)} + \sum_{n=1}^{N} \gamma_n \left(g^{(o,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right) \\
&\quad \times \begin{cases} \left(m+\frac{1}{2}\right)^2 \frac{\partial}{\partial z_j} g^{(o,L)}_{m,k}(c, A\varepsilon^3), & d_n = Y_{m,k},\ m \le L \\ 0, & d_n = Y_{m,k},\ m > L \\ \sum_{l=0}^{L} \sum_{i=-l}^{l} \left(l+\frac{1}{2}\right)^4 |x'|^l Y_{l,i}\left(\frac{x'}{|x'|}\right) \frac{\partial}{\partial z_j} g^{(o,L)}_{l,i}(c, A\varepsilon^3), & d_n = K(x', \cdot) \\ \sum_{l=0}^{L} \sum_{i=-l}^{l} \left(l+\frac{1}{2}\right)^4 \left(|x'|^l - |x'|^{2l}\right) Y_{l,i}\left(\frac{x'}{|x'|}\right) \frac{\partial}{\partial z_j} g^{(o,L)}_{l,i}(c, A\varepsilon^3), & d_n = W(x', \cdot) \\ \sum_{l=0}^{L} \sum_{i=-l}^{l} \left(l+\frac{1}{2}\right)^4 g^{(k,L)}_{l,i}(c', A'\varepsilon^3) \frac{\partial}{\partial z_j} g^{(o,L)}_{l,i}(c, A\varepsilon^3), & d_n = g^{(k,L)}\left(\left(c', A'\varepsilon^3\right), \cdot\right) \end{cases}
\end{aligned} \tag{7.94}
$$

7.2.5. Inner products of the penalty term

At last, we have to discuss the term $b_2(d(z))$ for $z \in \mathbb{R}^{\delta_z}$ (confer (7.19)) for Slepian functions as well as Abel–Poisson low and band pass filters. These terms were given in (3.23), (3.26) and (3.28) in Section 3.5. It remains to consider their derivatives. For the Slepian functions, we have

$$\begin{aligned}
&\frac{\partial}{\partial z_j} b_2\left(g^{(k,L)}\left(\left(c, A\varepsilon^3\right), \cdot\right)\right) \\
&\quad = \sum_{l=0}^{L} \sum_{i=-l}^{l} \left(l+\frac{1}{2}\right)^4 \frac{\partial}{\partial z_j} \left(g_{l,i}^{(o,L)}\left(c, A\varepsilon^3\right)\right)^2 \\
&\quad = 2 \sum_{l=0}^{L} \sum_{i=-l}^{l} \left(l+\frac{1}{2}\right)^4 g_{l,i}^{(o,L)}\left(c, A\varepsilon^3\right) \frac{\partial}{\partial z_j} g_{l,i}^{(o,L)}\left(c, A\varepsilon^3\right)
\end{aligned} \tag{7.95}$$

which is known due to (7.23) and (7.24). For the Abel–Poisson low and band pass filters, we see that a $\mathcal{H}_2(\Omega)$-norm is given by a series. We discussed that the limits can be exchanged here in Section 7.2.4. Before we give an explicit formulation of the terms $\nabla b_2(K(x,\cdot))$ and $\nabla b_2(W(x,\cdot))$, recall (7.88). Then we have for the gradient of the term $\nabla b_2(d(z))$ with respect to the Abel–Poisson low and band pass filters

$$\begin{aligned}
\nabla b_2(K(x,\cdot)) &= \nabla \left(\sum_{n=0}^{\infty} \left(n+\frac{1}{2}\right)^4 |x|^{2n} \frac{2n+1}{4\pi} P_n(1) \right) \\
&= \sum_{n=0}^{\infty} \left(n+\frac{1}{2}\right)^4 \frac{2n+1}{4\pi} \nabla |x|^{2n} \\
&= \sum_{n=0}^{\infty} \left(n+\frac{1}{2}\right)^4 \frac{2n+1}{4\pi} 2nx|x|^{2n-2} \\
&= \sum_{n=0}^{\infty} \left(n+\frac{1}{2}\right)^4 \frac{2n^2+n}{2\pi} x|x|^{2n-2}
\end{aligned} \tag{7.96}$$

and

$$\begin{aligned}
\nabla b_2(W(x,\cdot)) &= \nabla \left(\sum_{n=0}^{\infty} \left(n+\frac{1}{2}\right)^4 \left(|x|^n - |x|^{2n}\right)^2 \frac{2n+1}{4\pi} P_n(1) \right) \\
&= \sum_{n=0}^{\infty} \left(n+\frac{1}{2}\right)^4 \frac{2n+1}{4\pi} \nabla \left(|x|^n - |x|^{2n}\right)^2 \\
&= \sum_{n=0}^{\infty} \left(n+\frac{1}{2}\right)^4 \frac{2n+1}{2\pi} \left(|x|^n - |x|^{2n}\right) \nabla \left(|x|^n - |x|^{2n}\right) \\
&= \sum_{n=0}^{\infty} \left(n+\frac{1}{2}\right)^4 \frac{2n^2+n}{2\pi} \left(|x|^n - |x|^{2n}\right) \left(x|x|^{n-2} - 2x|x|^{2n-2}\right).
\end{aligned} \tag{7.97}$$

7.3. Additional features

Next, we describe some additional strategies that are either needed or, as we experienced, are possibly improving the runtime and / or the results. We start with a necessary feature for the LROFMP algorithm.

Splines for the LROFMP algorithm The objective function $\mathrm{ROFMP}(\cdot; N)$ is not well-defined for previously chosen dictionary elements. In the ROFMP algorithm, we can manually avoid the evaluation for these trial functions because the dictionary is finite. This is not possible for the LROFMP algorithm because the optimization problems are solved in an infinite set of trial functions and we cannot foresee which points are passed in the iterations of the optimization methods. In order to avoid neighbourhoods of previously chosen dictionary elements in future evaluation, we could implement one additional constraint for each chosen function. However, this is not sensible in practice because constraints usually make it harder to find an solution.
Thus, we pursue the following approach: we multiply the function $\mathrm{ROFMP}(\cdot; N)$ with a spline in order to set it to zero in the mentioned neighbourhoods. Note that it suffices to consider one spline per trial function class as mixed cases do not yield an ill-definedness of $\mathrm{ROFMP}(\cdot; N)$.
To be specific, by means of a spline, the objective function $\mathrm{ROFMP}(\cdot; N)$ must be altered such that it is well-defined and the transition to zero is twice continuously differentiable in each iteration of the LROFMP algorithm. Let $d(z^{(1)})$ be the first chosen dictionary element with $z^{(1)} \in \mathbb{R}^{\delta_z}$. Thus, in the second iteration, we exchange the objective function with

$$\mathrm{ROFMP}_S(d(z); 1) \coloneqq \mathrm{ROFMP}(d(z); 1) \cdot S^*_{z^{(1)}}(z).$$

In the case that $d(z)$, $z \in \mathbb{R}^3$, and $d(z^{(1)})$ are of the same type of Abel–Poisson (low or band pass) filters, the function $S^*_{z^{(1)}}$ is given by

$$S^*_{z^{(1)}} \coloneqq S^*_{d\left(z^{(1)}\right)} \in \mathrm{C}^{(2)}\left(\mathring{\mathbb{B}}\right)$$

with

$$\begin{aligned}
&S^*_{z^{(1)}}(z) = 0, && z \in \overline{\mathbb{B}}_\varepsilon\left(z^{(1)}\right),\\
&S^*_{z^{(1)}}(z) = 1, && z \in \mathring{\mathbb{B}} \setminus \mathring{\mathbb{B}}_{2\varepsilon}\left(z^{(1)}\right),\\
&S^*_{z^{(1)}}(z) \in (0,1), && z \in \mathring{\mathbb{B}}_{2\varepsilon}\left(z^{(1)}\right) \setminus \overline{\mathbb{B}}_\varepsilon\left(z^{(1)}\right).
\end{aligned}$$

If $d(z)$, $z \in \mathbb{R}^4$, and $d(z_1)$ are Slepian functions, we have

$$S^*_{z^{(1)}} \coloneqq S^*_{d\left(z^{(1)}\right)} \in \mathrm{C}^{(2)}\left([-1,1] \times [0, 2\pi[\times [0, \pi] \times [0, 2\pi[\right)$$

with

$$\begin{aligned} S^*_{z^{(1)}}(z) &= 0, && z \in \overline{\mathbb{B}}_\varepsilon\left(z^{(1)}\right), \\ S^*_{z^{(1)}}(z) &= 1, && z \in [-1,1] \times [0,2\pi[\times [0,\pi] \times [0,2\pi[\setminus \mathring{\mathbb{B}}_{2\varepsilon}\left(z^{(1)}\right), \\ S^*_{z^{(1)}}(z) &\in (0,1), && z \in \mathring{\mathbb{B}}_{2\varepsilon}\left(z^{(1)}\right) \setminus \overline{\mathbb{B}}_\varepsilon\left(z^{(1)}\right). \end{aligned}$$

In both cases, the function $S^*_{z^{(1)}}$ only depends on the distance between z and $z^{(1)}$ for $z,\ z^{(1)} \in \mathbb{R}^{\delta_z}$. In the case of an Abel–Poisson low or band pass filter, this distance is naturally given by $\|z - z^{(1)}\|^2_{\mathbb{R}^3}$. In the case of Slepian functions, any $z \in \mathbb{R}^4$ consists of the size of the spherical cap $c \in [-1,1]$ as well as the Euler angles $\alpha,\ \gamma \in [0,2\pi[$ and $\beta \in [0,\pi]$. The dependence on the Euler angles can be substituted by

$$\overline{z} := A(\alpha,\ \beta,\ \gamma)\varepsilon^3 = \begin{pmatrix} \cos(\alpha)\sin(\beta) \\ \sin(\alpha)\sin(\beta) \\ \cos(\beta) \end{pmatrix} \tag{7.98}$$

due to (3.5). That means, we consider $z = (c, \overline{z}) \in \mathbb{R}^4$, where the vector $\overline{z}$ gives us the centre of the localization region. We can measure the distance between z and $z^{(1)}$ as

$$\left|z - z^{(1)}\right|^2 := \left(c - c^{(1)}\right)^2 + \arccos\left(\overline{z} \cdot \overline{z}^{(1)}\right).$$

Note that we use $\arccos(\overline{z} \cdot \overline{z}^{(1)})$ in order to measure the angle between the centre vectors $\overline{z}$ and $\overline{z}^{(1)}$. To consider both cases (i. e. filters and Slepian functions) simultaneously, we write $\|z - z^{(1)}\|^2$ for $\|z - z^{(1)}\|^2_{\mathbb{R}^3}$ as well as $|z - z^{(1)}|^2$. Then, for $z,\ z^{(1)} \in \mathbb{R}^{\delta_z}$, we have

$$\mathrm{ROFMP}_S(d(z);1) = \mathrm{ROFMP}(d(z);1) \cdot S_{z^{(1)}}\left(\left\|z - z^{(1)}\right\|^2\right)$$

for a function $S_{z^{(1)}} \colon [0,2] \to \mathbb{R}$ with

$$\begin{aligned} S_{z^{(1)}}(\tau) &= 0, && \tau = \left\|z - z^{(1)}\right\|^2 \in [0,\varepsilon], \\ S_{z^{(1)}}(\tau) &= 1, && \tau = \left\|z - z^{(1)}\right\|^2 \geq 2\varepsilon, \\ S_{z^{(1)}}(\tau) &\in (0,1), && \tau = \left\|z - z^{(1)}\right\|^2 \in (\varepsilon, 2\varepsilon). \end{aligned}$$

The restriction

$$\left(S_{z^{(1)}}\right)_{|(\varepsilon,2\varepsilon)}$$

can be modelled as an one-dimensional polynomial. Then the function $S_{z^{(1)}}$ is a spline as the composition of this polynomial and constant zero functions on $[0, \varepsilon]$ and $[\varepsilon, 2]$, respectively. To ensure that

$$\mathrm{ROFMP}_S(d(z);1) \in \mathrm{C}^{(2)}(D), \quad D = \begin{cases} \mathring{\mathbb{B}}, & z \in \mathbb{R}^3 \\ [-1,1] \times [0, 2\pi[\times [0, \pi] \times [0, 2\pi[, & z \in \mathbb{R}^4, \end{cases}$$

the spline $S_{z^{(1)}}$ must be twice continuously differentiable. For this, only the points $\tau = \varepsilon$ and $\tau = 2\varepsilon$ are problematic. This can be treated, if we compute the polynomial $\left(S_{z^{(1)}}\right)_{|(\varepsilon,2\varepsilon)}$ via the following conditions

$$\begin{aligned} &\left(S_{z^{(1)}}\right)_{|(\varepsilon,2\varepsilon)}(\tau) \\ &\quad := a + b(\tau - \tau_0) + c(\tau - \tau_0)^2 + d(\tau - \tau_0)^3 + e(\tau - \tau_0)^4 + f(\tau - \tau_0)^5 \\ &\tau_0 = \varepsilon, \qquad S_{z^{(0)}}(\tau_0) = 0, \qquad S'_{z^{(0)}}(\tau_0) = 0, \qquad S''_{z^{(0)}}(\tau_0) = 0, \\ &\tau_1 = 2\varepsilon, \qquad S_{z^{(0)}}(\tau_1) = 1, \qquad S'_{z^{(0)}}(\tau_1) = 0, \qquad S''_{z^{(0)}}(\tau_1) = 0. \end{aligned}$$

The derivatives are given by

$$\begin{aligned} \left(S'_{z^{(1)}}\right)_{|(\varepsilon,2\varepsilon)}(\tau) &:= b + 2c(\tau - \tau_0) + 3d(\tau - \tau_0)^2 + 4e(\tau - \tau_0)^3 + 5f(\tau - \tau_0)^4, \\ \left(S''_{z^{(1)}}\right)_{|(\varepsilon,2\varepsilon)}(\tau) &:= 2c + 6d(\tau - \tau_0) + 12e(\tau - \tau_0)^2 + 20f(\tau - \tau_0)^3. \end{aligned}$$

The boundary values give us the system of linear equations

$$\begin{aligned} 0 &= a, & 1 &= a + b\varepsilon + c\varepsilon^2 + d\varepsilon^3 + e\varepsilon^4 + f\varepsilon^5 \\ & & &= d\varepsilon^3 + e\varepsilon^4 + f\varepsilon^5, \\ 0 &= b, & 0 &= b + 2c\varepsilon + 3d\varepsilon^2 + 4e\varepsilon^3 + 5f\varepsilon^4 \\ & & &= 3d\varepsilon^2 + 4e\varepsilon^3 + 5f\varepsilon^4, \\ 0 &= 2c, & 0 &= 2c + 6d\varepsilon + 12e\varepsilon^2 + 20f\varepsilon^3 \\ & & &= 6d\varepsilon + 12e\varepsilon^2 + 20f\varepsilon^3. \end{aligned} \tag{7.99}$$

We compute the spline for $\varepsilon = 1$ and then dilate it to an arbitrary $\varepsilon > 0$. For $\varepsilon = 1$, the system of linear equations (7.99) rearranges to

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 3 & 4 & 5 \\ 6 & 12 & 20 \end{pmatrix} \begin{pmatrix} d \\ e \\ f \end{pmatrix}.$$

The solution of this system of linear equations is given by

$$d = 10, \qquad e = -15, \qquad f = 6$$

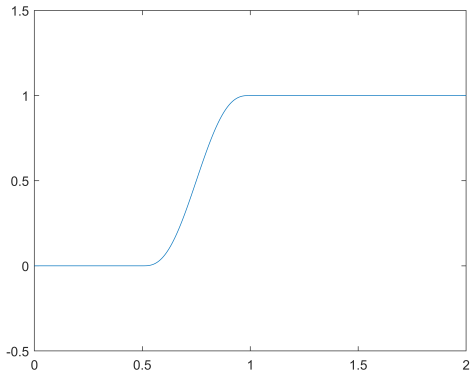


Figure 7.2.: The spline $S_{z_1}(t)$ for $t \in [0,2]$ and $\varepsilon = 0.5$.

and we have obtained the polynomial $\left(S_{z^{(1)}}\right)_{|(1,2)}$ as

$$\left(S_{z^{(1)}}\right)_{|(1,2)}(v) := 10(v-1)^3 - 15(v-1)^4 + 6(v-1)^5.$$

For an arbitrary $\varepsilon > 0$, we map the interval $[\varepsilon, 2\varepsilon]$ to $[1,2]$ by

$$[\varepsilon, 2\varepsilon] \ni \tau \mapsto v = \frac{\tau}{\varepsilon} \in [1,2].$$

All in all, we have the sigmoidal-like spline $S_{z^{(1)}}$ (see also Figure 7.2) given by

$$\begin{aligned} \left(S_{z^{(1)}}\right)_{|[0,\varepsilon]} &\equiv 0, \\ \left(S_{z^{(1)}}\right)_{|(\varepsilon,2\varepsilon)}(\tau) &:= 10\left(\frac{\tau}{\varepsilon} - 1\right)^3 - 15\left(\frac{\tau}{\varepsilon} - 1\right)^4 + 6\left(\frac{\tau}{\varepsilon} - 1\right)^5 \end{aligned}$$

and

$$\left(S_{z^{(1)}}\right)_{|[2\varepsilon,2]} \equiv 1.$$

The spline $S_{z^{(1)}}$ serves our needs: if it is multiplied with $\mathrm{ROFMP}(\cdot;1)$, we obtain a for all feasible argument values well-defined objective function $\mathrm{ROFMP}_S(\cdot;1)$ for the LROFMP algorithm. By induction, we obtain analogously for the iteration $N \in \mathbb{N}$

$$\mathrm{ROFMP}_S(d(z);N) = \mathrm{ROFMP}(d(z);N) \cdot \prod_{n=1}^{N} S_{z^{(n)}}\left(\left\|z - z^{(n)}\right\|^2\right). \tag{7.100}$$

For gradient-based optimization algorithms, we have to consider the gradient of $\mathrm{ROFMP}_S(d(z);N)$. Note that the partial derivatives of $\mathrm{ROFMP}(d(z);N)$ are given

in (7.21). We obtain

$$\frac{\partial}{\partial z_j}\mathrm{ROFMP}_S(d(z);N) := \left(\frac{\partial}{\partial z_j}\mathrm{ROFMP}(d(z);N)\right)\prod_{n=1}^{N} S_{z^{(n)}}\left(\left\|z-z^{(n)}\right\|^2\right) + \mathrm{ROFMP}(d(z);N)\left(\frac{\partial}{\partial z_j}\prod_{n=1}^{N} S_{z^{(n)}}\left(\left\|z-z^{(n)}\right\|^2\right)\right), \tag{7.101}$$

where

$$\frac{\partial}{\partial z_j}\prod_{n=1}^{N} S_{z^{(n)}}\left(\left\|z-z^{(n)}\right\|^2\right) = \sum_{n=1}^{N}\left(\prod_{\nu=1,\nu\neq n}^{N} S_{z^{(\nu)}}\left(\left\|z-z^{(\nu)}\right\|^2\right)\right)\frac{\partial}{\partial z_j}S_{z^{(n)}}\left(\left\|z-z^{(n)}\right\|^2\right) \tag{7.102}$$

with

$$\frac{\partial}{\partial z_j}S_{z^{(n)}}\left(\left\|z-z^{(n)}\right\|^2\right) = 0$$

if $\|z-z^{(n)}\|^2 \leq \varepsilon$ or $\|z-z^{(n)}\|^2 \geq 2\varepsilon$ and

$$\begin{aligned}
&\frac{\partial}{\partial z_j}S_{z^{(n)}}\left(\left\|z-z^{(n)}\right\|^2\right)\\
&= \frac{\partial}{\partial z_j}\left(10\left(\frac{\left\|z-z^{(n)}\right\|^2}{\varepsilon}-1\right)^3 - 15\left(\frac{\left\|z-z^{(n)}\right\|^2}{\varepsilon}-1\right)^4 + 6\left(\frac{\left\|z-z^{(n)}\right\|^2}{\varepsilon}-1\right)^5\right)\\
&= 30\left(\frac{\left\|z-z^{(n)}\right\|^2}{\varepsilon}-1\right)^2\frac{\partial}{\partial z_j}\frac{\left\|z-z^{(n)}\right\|^2}{\varepsilon} - 60\left(\frac{\left\|z-z^{(n)}\right\|^2}{\varepsilon}-1\right)^3\frac{\partial}{\partial z_j}\frac{\left\|z-z^{(n)}\right\|^2}{\varepsilon}\\
&\quad + 30\left(\frac{\left\|z-z^{(n)}\right\|^2}{\varepsilon}-1\right)^4\frac{\partial}{\partial z_j}\frac{\left\|z-z^{(n)}\right\|^2}{\varepsilon}
\end{aligned}$$

if $\|z-z^{(n)}\|^2 \in (\varepsilon, 2\varepsilon)$. In particular, in the case of Abel–Poisson low or band pass filters, we have ($z = x \in \mathbb{R}^3$)

$$\left\|x-x^{(n)}\right\|_{\mathbb{R}^3}^2 = \sum_{l=1}^{3}\left(x_l - x_l^{(n)}\right)^2 \quad \text{and} \quad \frac{\partial}{\partial x_j}\left\|x-x^{(n)}\right\|_{\mathbb{R}^3}^2 = 2\left(x_j - x_j^{(n)}\right)$$

which yields for the spline S_{x_n}

$$\begin{aligned}
&\frac{\partial}{\partial x_j} S_{x^{(n)}}\left(\left\|x-x^{(n)}\right\|_{\mathbb{R}^3}^2\right) \\
&\quad = 60\left(\frac{\left\|x-x^{(n)}\right\|_{\mathbb{R}^3}^2}{\varepsilon}-1\right)^2 \frac{x_j-x_j^{(n)}}{\varepsilon} - 120\left(\frac{\left\|x-x^{(n)}\right\|_{\mathbb{R}^3}^2}{\varepsilon}-1\right)^3 \frac{x_j-x_j^{(n)}}{\varepsilon} \\
&\qquad + 60\left(\frac{\left\|x-x^{(n)}\right\|_{\mathbb{R}^3}^2}{\varepsilon}-1\right)^4 \frac{x_j-x_j^{(n)}}{\varepsilon}.
\end{aligned}$$

if $\varepsilon < \|x-x^{(n)}\|_{\mathbb{R}^3}^2 < 2\epsilon$. Similarly, in the case of Slepian functions, we have

$$\begin{aligned}
&\frac{\partial}{\partial c} S_{z^{(n)}}\left(\left|z-z^{(n)}\right|^2\right) \\
&\quad = 60\left(\frac{\left|z-z^{(n)}\right|^2}{\varepsilon}-1\right)^2 \frac{c-c^{(n)}}{\varepsilon} - 120\left(\frac{\left|z-z^{(n)}\right|^2}{\varepsilon}-1\right)^3 \frac{c-c^{(n)}}{\varepsilon} \\
&\qquad + 60\left(\frac{\left|z-z^{(n)}\right|^2}{\varepsilon}-1\right)^4 \frac{c-c^{(n)}}{\varepsilon}.
\end{aligned}$$

For the partial derivatives with respect to a Euler angles $\alpha,\ \beta,\ \gamma$, we obtain

$$\begin{aligned}
\frac{\partial}{\partial z_j}\left|z-z^{(n)}\right|^2 &= \frac{\partial}{\partial z_j}\left(\left(c-c^{(n)}\right)^2 + \arccos\left(\overline{z}\cdot\overline{z}^{(n)}\right)\right) \\
&= \frac{\partial}{\partial z_j}\arccos\left(\overline{z}\cdot\overline{z}^{(n)}\right) \\
&= -\frac{1}{\sqrt{1-\overline{z}\cdot\overline{z}^{(n)}}}\frac{\partial}{\partial z_j}\left(\overline{z}\cdot\overline{z}^{(n)}\right) \\
&= -\frac{1}{\sqrt{1-\overline{z}\cdot\overline{z}^{(n)}}}\left(\frac{\partial}{\partial z_j}\overline{z}\right)\cdot\overline{z}^{(n)}
\end{aligned}$$

with

$$\frac{\partial}{\partial\alpha}\overline{z} = \begin{pmatrix}-\sin(\alpha)\sin(\beta)\\ \cos(\alpha)\sin(\beta)\\ 0\end{pmatrix},\ \frac{\partial}{\partial\beta}\overline{z} = \begin{pmatrix}\cos(\alpha)\cos(\beta)\\ \sin(\alpha)\cos(\beta)\\ -\sin(\beta)\end{pmatrix},\ \frac{\partial}{\partial\gamma}\overline{z} = \begin{pmatrix}0\\0\\0\end{pmatrix}.$$

All in all, in this last case, we have

$$\begin{aligned}
&\frac{\partial}{\partial z_j} S_{z^{(n)}}\left(\left|z - z^{(n)}\right|^2\right) \\
&= -\frac{1}{\varepsilon\sqrt{1 - \overline{z}\cdot\overline{z}^{(n)}}}\left[30\left(\frac{\left|z - z^{(n)}\right|^2}{\varepsilon} - 1\right)^2\left(\frac{\partial}{\partial z_j}\overline{z}\right)\cdot\overline{z}^{(n)}\right. \\
&\quad \left. - 60\left(\frac{\left|z - z^{(n)}\right|^2}{\varepsilon} - 1\right)^3\left(\frac{\partial}{\partial z_j}\overline{z}\right)\cdot\overline{z}^{(n)} + 30\left(\frac{\left|z - z^{(n)}\right|^2}{\varepsilon} - 1\right)^4\left(\frac{\partial}{\partial z_j}\overline{z}\right)\cdot\overline{z}^{(n)}\right]
\end{aligned}$$

for $z_j \in \{\alpha,\ \beta,\ \gamma\}$ and $|z - z^{(n)}|^2 \in (\varepsilon, 2\varepsilon)$.

Features for guiding the learning process Taking into account that, in each iteration, we solve optimization problems with a complicated objective function and – in some cases – non-linear constraints, it seems expectable that the learning process might need some guidance. During the development, we saw that the following techniques may be helpful to improve the results. Each technique addresses one out of two challenges:

(1) accurately solving the optimization problems in an acceptable time;
(2) obtaining an approximation with a low data and approximation error (if the latter one is available).

With respect to challenge (1), there are several aspects that have to be considered: is any solution of the respective optimization problem obtained? How accurate is this solution? How long did the solver take? And, as we discovered, is the learnt dictionary well applicable in an independent run of the IPMP?

Usually, published libraries or other software packages are built to solve even very complicated optimization problems. However, they might need some guidance as well. Most of such software expects at least one termination criterion. Possible criteria rank from measuring the distance between values of the objective function between two successive iteration points, measuring the distance between such points themselves, having a maximally allowed number of iterations or a maximally allowed time. Note that a theoretically based termination criterion would be that the current iterate fulfils (numerically) the Karush-Kuhn-Tucker (KKT) conditions. Furthermore, published software packages usually come with all kinds of expert level techniques that provide for a best possible solving routine. For instance, it may be supportive to use scaling techniques. As our objective functions depend on given data, we discovered that such a scaling approach may result in better approximations dependent on the structure of the data. We described such a scaling in Michel and Schneider (2020). Note that, in previous dictionary learning publications, the structure of the (training) data was taken into account as well (see e. g. Prünte, 2008).

Remark 7.3.1. When using published software, it is helpful to investigate their options and experiment with them to obtain better results in the learning algorithms.

In the sequel, we concentrate on the effect of termination criteria. We set at least one to ensure the termination of the numerical optimization process. However, dependent on the specific criteria, we may lose valuable accuracy in our solution, e. g. if we allow too few iterations. In particular with the LIPMP algorithms, if we set a time limit, we might run into problems. At a first glance, it seems favourable to set such a limit because it ensures that each iteration of an LIPMP algorithm takes at most a pre-defined time. However, we have to evaluate one of the objective functions $\mathrm{RFMP}(\cdot; N)$ or $\mathrm{ROFMP}_S(\cdot; N)$ in each iteration of the optimization process at least once. Both of these functions depend on the linear combination of previously chosen basis elements (see, for instance, the term $a_2(d(z))$). That means, in each iteration of the LIPMP algorithm, both objective functions contain more summands in certain terms.

Remark 7.3.2. Each iteration of the optimization process takes longer than in the optimization problem of the previous LIPMP iteration.

Hence, if we set a time limit for the optimization processes, the solvers can take less steps in later iterations of the LIPMP algorithm which inevitably leads to a loss in accuracy.

Remark 7.3.3. All in all, with task (1), we have to make a trade between runtime cost and accuracy.

However, as one can expect, less accuracy leads to further problems. If the learnt dictionary shall be used in an IPMP algorithm, its application may not behave as expected for less accurate solutions. In general, it can be expected that the IPMP algorithm picks the basis elements in the same order as the LIPMP algorithm does because the learnt dictionary is a subset of the infinite dictionary $\mathcal{D}^{\mathrm{Inf}}$. However, this order would not preserved if the learnt dictionary elements were not computed accurately enough, i. e. if they were not near enough to the global optimum. Unfortunately, this would ultimately lead to non-competitive results.

Remark 7.3.4. We suggest to favour more accuracy in general. If this is not possible due to high runtime costs, the application of the learnt dictionary in an IPMP algorithm should be guided with an iterated application. In each step $N \in \mathbb{N}_0$ of the IPMP algorithm, only the learnt dictionary elements $d_n,\ n = 1, ..., N+1$ are allowed to be chosen. If $N+1$ exceeds the number of learnt dictionary elements, all possible elements can be chosen.

We also experimented with a restart technique similar (but not identical) to the one from the iterated ROFMP algorithm. The evaluation of the objective function

is responsible for the main part of the runtime of each optimization iteration and, thus, of each optimization. As we outlined before, this time increases with the iterations of the LIPMP algorithm.

Remark 7.3.5. In order to cut runtime costs, it seems sensible to reset all sums that depend on previously chosen basis elements after a certain number of iterations. Alternatively, we could also limit the number of summands and, thus, obtain a L-BFGS-like reset method. Nonetheless, it must be evaluated for a specific problem whether a trade between speed-up and increasing approximation error is acceptable.

With this restart, the LIPMP algorithm is similar to minimizing an iterated Tikhonov functional, see Example 2.5.17.
With respect to challenge (2), we consider the approximation we obtained and whether it suits our expectation. It is well-known that, in any method for ill-posed inverse problems, the choice of the regularization parameter is critical for the obtained approximation. For the LIPMP algorithm, we discovered that the choice of the regularization parameter is also critical for the application of the learnt dictionary.

Remark 7.3.6. The learnt dictionary can only be well applied for the regularization parameter with which it was learnt.

This still leaves us with the determination of a suitable regularization parameter. From theory, we know that, if the regularization parameter is too high, any method concentrates more on the penalty term and, thus, the obtained approximation is rather smooth. On the contrary, if the regularization parameter is too low, the regularization has only little effect which means the method concentrates only on the data fidelity term. This, however, may lead to artefacts in the approximation. This also coincides with our experiences with the IPMP algorithms. If the regularization parameter is chosen too high, the IPMP algorithms do not reproduce many local structures which is caused by the fact that they choose more global trial functions or not suitable local ones. If the regularization parameter, however, is chosen very low, we obtain local artefacts because the algorithm chooses local trial functions with very high scales.

Remark 7.3.7. It seems that the LIPMP algorithm is very sensitive to the choice of the particular regularization parameter. Thus, if, for any reason, a constant value for $\lambda(\delta, y^\delta)$ is not favoured, we suggest to use a non-stationary regularization parameter of the form

$$\lambda_N\left(\delta, y^\delta\right) := \frac{\lambda_0(\delta)\left\|y^\delta\right\|_{\mathbb{R}^\ell}}{N+1}$$

for the N-th iteration of the LIPMP algorithm and with a constant $\lambda_0(\delta) \in \mathbb{R}$.

In this way, the regularization parameter decreases in each iteration. For a suitable value $\lambda_0(\delta)$, this leads to smoother choices of dictionary elements in the beginning of the LIPMP algorithm and supports choices of local dictionary elements in later iterations. Depending on the given data, approximating global structures first appears to have a positive effect on how easily the optimization routines determine candidates.
If a non-stationary regularization parameter is still not providing enough smoothness in the approximation, we suggest a further mechanism.

Remark 7.3.8. It may also be sensible not to choose from all possible trial function classes in $\mathcal{D}^{\mathrm{Inf}}$ in each iteration.

For instance, we only choose spherical harmonics, possibly in addition with Slepian functions, in the first iterations of the LIPMP algorithm. In later iterations, we allow the LIPMP algorithm to also choose Abel–Poisson low and band pass filters. This technique resembles to run the LIPMP algorithm twice but with different dictionaries and different initial approximations. First, we allow only possibly smoother trial function classes (e. g. spherical harmonics and Slepian functions) and start the first run of the LIPMP algorithm with any initial approximation f_0. The approximation obtained from this first run is denoted by f_{N_0} for a $N_0 \in \mathbb{N}_0$. In the second run of the LIPMP algorithm, we also allow local trial functions. However, we start the algorithm now with an initial approximation $\tilde{f}_0 = f_0 + f_{N_0}$. If $f_0 \equiv 0$, the second run yields a refinement of f_{N_0}. However, note that this smoothing mechanism might influence the applicability of the learnt dictionary. In this case, it should be combined with the iterated approach mentioned above. We state which of these additional features are applied in our experiments when we describe their settings in Chapter 9. In the following pseudo-code of the LIPMP algorithms, we do without any of the features.

7.4. Pseudo-codes for the LIPMP algorithms

At the end of this chapter, we summarize the novel learning algorithm in pseudo-codes. A description of an efficient implementation can be found in Appendix B. The learning algorithms generalize a respective IPMP algorithm to an infinite dictionary $\mathcal{D}^{\mathrm{Inf}}$. The main question of each iteration $N \in \mathbb{N}_0$ is how to determine the next basis element $d_{N+1} \in \mathcal{D}^{\mathrm{Inf}}$.
We consider a finite number of spherical harmonics and determine a candidate for d_{N+1} among the spherical harmonics by trial and error in order to learn a maximal degree. Thus, the learning algorithm uses a starting dictionary which contains at least these spherical harmonics. The determination of the maximizing Abel–Poisson low and band pass filter as well as Slepian function is modelled as optimization problems. To support numerical optimization routines, it is sensible to insert a few of those trial functions into the starting dictionary as well. In the

Data: $R^N \in \mathbb{R}^\ell$, $N \in \mathbb{N}_0$, $\mathcal{D}^{\mathrm{s}}$
Result: choice of next basis element d_{N+1}

(1) determine

$$d_{N+1}^{\mathrm{SH}} = \underset{d \in [\cdot]_{\mathrm{SH}}^{\mathrm{s}}}{\arg\max}\, \mathrm{IPMP}(d; N);$$

(2) compute starting choices $d_{N+1}^{(\mathrm{s},\bullet)}(z)$, $\bullet \in \{\mathrm{APK},\ \mathrm{APW},\ \mathrm{SL}\}$:

$$d_{N+1}^{(\mathrm{s},\bullet)} = \underset{d(z) \in [\cdot]_{\bullet}^{\mathrm{s}}}{\arg\max}\, \mathrm{IPMP}(d(z); N);$$

(3) compute global choices $d_{N+1}^{(\mathrm{g},\bullet)}(z)$, $\bullet \in \{\mathrm{APK},\ \mathrm{APW},\ \mathrm{SL}\}$, $z \in \mathbb{R}^{\delta_z}$, using starting choices as starting points in the optimization processes:

$$d_{N+1}^{(\mathrm{g,APK})} = \underset{d(z) \in \left[\mathring{\mathbb{B}}\right]_{\mathrm{APK}}}{\arg\max}\ \mathrm{IPMP}(d(z); N);$$

$$d_{N+1}^{(\mathrm{g,APW})} = \underset{d(z) \in \left[\mathring{\mathbb{B}}\right]_{\mathrm{APW}}}{\arg\max}\ \mathrm{IPMP}(d(z); N);$$

$$d_{N+1}^{(\mathrm{g,SL})} = \underset{d(z) \in \left[\mathbb{B}^4\right]_{\mathrm{SL}}}{\arg\max}\ \mathrm{IPMP}(d(z); N);$$

(4) compute local solutions $d_{N+1}^{(\mathrm{l},\bullet)}(z)$, $\bullet \in \{\mathrm{APK},\ \mathrm{APW},\ \mathrm{SL}\}, z \in \mathbb{R}^{\delta_z}$, using global choices as starting points in the optimization processes:

$$d_{N+1}^{(\mathrm{l,APK})} = \underset{d(z) \in \left[\mathring{\mathbb{B}}\right]_{\mathrm{APK}}}{\arg\max}\ \mathrm{IPMP}(d(z); N);$$

$$d_{N+1}^{(\mathrm{l,APW})} = \underset{d(z) \in \left[\mathring{\mathbb{B}}\right]_{\mathrm{APW}}}{\arg\max}\ \mathrm{IPMP}(d(z); N);$$

$$d_{N+1}^{(\mathrm{l,SL})} = \underset{d(z) \in \left[\mathbb{B}^4\right]_{\mathrm{SL}}}{\arg\max}\ \mathrm{IPMP}(d(z); N);$$

(5) choose

$$d_{N+1} = \underset{\hat{d} \in \mathcal{C}}{\arg\max}\, \mathrm{IPMP}\left(\hat{d}; N\right),$$

$$\mathcal{C} := \left\{ d_{N+1}^{\mathrm{SH}},\ d_{N+1}^{(\mathrm{s},\bullet)},\ d_{N+1}^{(\mathrm{g},\bullet)},\ d_{N+1}^{(\mathrm{l},\bullet)} \;\middle|\; \bullet \in \{\mathrm{APK},\ \mathrm{APW},\ \mathrm{SL}\} \right\};$$

return d_{N+1};

Algorithm 5: Pseudo-code for choosing a dictionary element from $\mathcal{D}^{\mathrm{Inf}}$ for the LIPMP algorithms.

Data: $y \in \mathbb{R}^\ell$
Result: approximation f_N
initialization: $\mathcal{D}^{\mathrm{s}}$, f_0, $R^0 = y - T\neg f_0$;
$N = 0$;
while *(stopping criteria not fulfilled)* **do**
determine d_{N+1} via Algorithm 5 with $\mathrm{IPMP}(\cdot; N) = \mathrm{RFMP}(\cdot; N)$;
compute

$$\alpha_{N+1} = \frac{\left\langle R^N, T\neg d_{N+1}\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\langle f_N, d_{N+1}\rangle_{\mathcal{H}_2(\Omega)}}{\|T\neg d_{N+1}\|^2_{\mathbb{R}^\ell} + \lambda\left(\delta, y^\delta\right)\|d_{N+1}\|^2_{\mathcal{H}_2(\Omega)}};$$

$$R^{N+1} = R^N - \alpha_{N+1} T\neg d_{N+1};$$

N be increased by 1;

end
return $f_N = \sum_{n=1}^N \alpha_n d_n$;

Algorithm 6: Pseudo-code for the learning regularized functional matching pursuit (LRFMP) algorithm.

sequel, we denote the starting dictionary as

$$\mathcal{D}^{\mathrm{s}} = \left[\widehat{N}\right]^{\mathrm{s}}_{\mathrm{SH}} + [S]^{\mathrm{s}}_{\mathrm{SL}} + [B_K]^{\mathrm{s}}_{\mathrm{APK}} + [B_W]^{\mathrm{s}}_{\mathrm{APW}}$$

for certain finite sets $\widehat{N}$ (see Section 7.2.1), B_K, B_W and S as given in Definition 4.2.2. We first compute the best candidate among the spherical harmonics as well as starting choices for the other trial function classes from the starting dictionary $\mathcal{D}^{\mathrm{s}}$. Then we solve the occurring optimization problems, i. e. dependent on which LIPMP algorithm we run and using the starting choices, we maximize the objective function

$$\mathrm{IPMP}(d(z); N) \coloneqq \begin{cases} \mathrm{RFMP}(d(z); N) \\ \mathrm{ROFMP}_S(d(z); N) \end{cases} \tag{7.103}$$

subject to the constraints

$$z \in \mathring{\mathbb{B}}$$

in the case of the Abel–Poisson low and band pass filters and

$$z \in \mathbb{B}^4$$

in the case of the Slepian functions. Note that we use the objective function ROFMP_S in order to avoid ill-defined function values. We first solve these problems with global optimization routines in order to ensure to obtain a solution for

each problem that is at least located in the correct region. To improve these global solutions, we then solve each optimization problem again with a local optimization routine which uses the respective global solution as its starting point. In this way, we obtain a set of candidates $\mathcal{C}$ given by

$$\mathcal{C} := \left\{ d_{N+1}^{\mathrm{SH}},\, d_{N+1}^{(\mathrm{s},\bullet)},\, d_{N+1}^{(\mathrm{g},\bullet)},\, d_{N+1}^{(\mathrm{l},\bullet)},\, \middle|\, \bullet \in \{\mathrm{APK},\ \mathrm{APW},\ \mathrm{SL}\} \right\},$$

where the candidates with superscript $(\mathrm{s},\bullet)$ denotes the starting choices, with superscript $(\mathrm{g},\bullet)$ the global solutions and with superscript $(\mathrm{l},\bullet)$ the local solutions for $\bullet \in \{\mathrm{APK},\ \mathrm{APW},\ \mathrm{SL}\}$. We insert the starting choices in $\mathcal{C}$ as well to ensure to have a candidate for every trial function class in case that the optimization routine may fail. The set $\mathcal{C}$ is obviously finite and small. Thus, for each of these candidates, we can evaluate the function $\mathrm{IPMP}(\cdot; N)$. We compare these values and set

$$d_{N+1} := \operatorname*{arg\,max}_{\hat{d} \in \mathcal{C}} \left\{ \mathrm{IPMP}\left(\hat{d}; N\right) \right\}.$$

In this way, we obtain the maximizing trial function from $\mathcal{D}^{\mathrm{Inf}}$. This approach of choosing a basis element from $\mathcal{D}^{\mathrm{Inf}}$ is described in Algorithm 5. It replaces the corresponding choice in Algorithm 3 and Algorithm 4, respectively. Thus, for the LRFMP, we obtain the pseudo-code Algorithm 6, and, for the LROFMP, the pseudo-code Algorithm 7. Note that, in Algorithm 7, we inserted the restart procedure from the iterated ROFMP algorithm (and not the restart method from the additional features). As the iterated ROFMP algorithm usually supersedes the (stationary) ROFMP algorithm, this strategy is inherited by the LROFMP algorithm.

Data: $y \in \mathbb{R}^\ell$
Result: approximation $f_N^{(N)}$
initialization: $\mathcal{D}^s$, f_0, $R^0 = y - T\neg f_0$, $K \in \mathbb{N}$;
$N = 0$;
while *(stopping criteria not fulfilled)* **do**
determine ${}_k d_{N+1}$ via Algorithm 5 with $\mathrm{IPMP}(\cdot; N) = \mathrm{ROFMP}_S(\cdot; N)$;
compute

$$ {}_k\alpha_{n+1}^{(N+1)} = \frac{\left\langle {}_kR^N, \mathcal{P}_{{}_k\mathcal{V}_N^\perp} T\neg d_{N+1} \right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle {}_kf_N^{(N)}, {}_kd_{N+1} - {}_kb_N^{(N)}({}_kd_{N+1}) \right\rangle_{\mathcal{H}_2(\Omega)}}{\left\| \mathcal{P}_{{}_k\mathcal{V}_N^\perp} T\neg {}_kd_{N+1} \right\|_{\mathbb{R}^\ell}^2 + \lambda\left(\delta, y^\delta\right)\left\| {}_kd_{N+1} - {}_kb_N^{(N)}({}_kd_{N+1}) \right\|_{\mathcal{H}_2(\Omega)}^2}; $$

$$ {}_kR^{N+1} = {}_kR^N - {}_k\alpha_{N+1}^{(N+1)} \mathcal{P}_{{}_k\mathcal{V}_N^\perp} T\neg\, {}_kd_{N+1}; $$

if $(N \geq 1)$ **then**
${}_k\alpha_n^{(N+1)} = {}_k\alpha_n^{(N)} - {}_k\alpha_{N+1}^{(N+1)} \beta_n^{(N)}({}_kd_{N+1})$, $n = 1, ..., N$;
end
N be increased by 1;
if $(N == K)$ **then**
${}_k\mathcal{V}_N \coloneqq \{0\}$;
$N = 0$;
k increased by 1;
end
end
return ${}_kf_N^{(N)} = \sum_{\kappa=1}^{k-1} \sum_{n=1}^{K} {}_\kappa\alpha_n^{(N)}\, {}_\kappa d_n + \sum_{n=1}^{N} {}_k\alpha_n^{(N)}\, {}_kd_n$;

Algorithm 7: Pseudo-code for the learning regularized orthogonal functional matching pursuit (LROFMP) algorithm. Note that it includes the same restart mechanism as the iterated ROFMP, see Algorithm 4.

8. Theoretical aspects

In this chapter, we outline the main theoretical aspects of the algorithms presented in Section 4.3, Section 4.4 and Chapter 7. We divide these aspects into two topics. First, we discuss the convergence of the (L)IPMP algorithms. Note that we did not summarize such results for the IPMP algorithms previously. As we now see that the learning algorithms are generalizations of these methods, we discuss the results at this point. The second topic considers how the LIPMP algorithms are related to previous works on dictionary learning as well as the dictionary properties developed in Section 6.2.

8.1. On the convergence

This section is based on Kontak (2018); Kontak and Michel (2019); Michel (2013, 2020); Michel and Orzlowski (2017); Michel and Telschow (2016); Telschow (2014). First of all, we note the setting under which we investigate convergence.

Remark 8.1.1. We will discuss the convergence of the (L)IPMP algorithms without any additional features but the spline in the case of the learning methods. Hence, in particular, we consider a stationary regularization parameter $\lambda(\delta, y^\delta)$ in the sequel. Note that, as we are considering results for infinitely many iterations, the, in practice, possible iterated use of the dictionary (confer Remark 7.3.4) has no influence here.

We start the discussion for the (L)RFMP algorithm. For the description of the LRFMP algorithm, it was helpful to introduce the objective function $\mathrm{RFMP}(\cdot;\cdot)$ as a quotient, see Definition 7.1.2. However, for the theoretical discussion, it is more convenient to reformulate it first. Recall the objective function of the N-th iteration as given in (7.6). With

$$C(d) := \sqrt{\|T\neg d\|_{\mathbb{R}^\ell}^2 + \lambda(\delta, y^\delta)\|d\|_{\mathcal{H}_2(\Omega)}^2}\,, \tag{8.1}$$

we obtain

$$\begin{aligned}\mathrm{RFMP}(d;N) &= \left(\frac{\langle y - T\neg f_N, T\neg d\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\langle f_N, d\rangle_{\mathcal{H}_2(\Omega)}}{C(d)}\right)^2 \\ &= \left(\left\langle y - T\neg f_N, T\neg\left(\frac{d}{C(d)}\right)\right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right)\left\langle f_N, \frac{d}{C(d)}\right\rangle_{\mathcal{H}_2(\Omega)}\right)^2.\end{aligned}$$

Thus, for any dictionary $\mathcal{D}$ and $d_{N+1} \in \mathcal{D}$ chosen via (4.25) and (7.4), respectively, we see that d_{N+1} is equivalent to a choice

$$\tilde{d}_{N+1} \in \widetilde{\mathcal{D}} := (C(\cdot))^{-1})\mathcal{D} := \left\{ \tilde{d} \;\middle|\; \tilde{d} = \frac{d}{C(d)},\ d \in \mathcal{D} \right\}.$$

Note that it holds $C(\tilde{d}) = 1$ due to the bilinearity of the norms and the linearity of the operator. Hence, if we consider the dictionary $\widetilde{\mathcal{D}}$, we obtain the coefficient α_{N+1} corresponding to d_{N+1} by

$$\begin{aligned}\tilde{\alpha}_{N+1} &= \sqrt{\mathrm{RFMP}\left(\tilde{d}_{N+1}; N\right)} \\ &= \left\langle y - T_\neg f_N, T_\neg \tilde{d}_{N+1} \right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right) \left\langle f_N, \tilde{d}_{N+1} \right\rangle_{\mathcal{H}_2(\Omega)}.\end{aligned}$$

For these considerations, see also Kontak (2018); Michel (2020). In the sequel, we will consider the formulations

$$\mathrm{RFMP}(d; N) = \left(\left\langle y - T_\neg f_N, T_\neg \tilde{d} \right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right) \left\langle f_N, \tilde{d} \right\rangle_{\mathcal{H}_2(\Omega)} \right)^2 \tag{8.2}$$

and

$$\alpha_{N+1} = \left\langle y - T_\neg f_N, T_\neg d_{N+1} \right\rangle_{\mathbb{R}^\ell} - \lambda\left(\delta, y^\delta\right) \left\langle f_N, d_{N+1} \right\rangle_{\mathcal{H}_2(\Omega)} \tag{8.3}$$

for a better readability in this chapter. Note that, analogously to (4.24), we still have

$$\mathcal{J}\left(f_{N+1}; T_\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) = \mathcal{J}\left(f_N; T_\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) - \mathrm{RFMP}(d_{N+1}; N). \tag{8.4}$$

The equations (8.2) and (8.3) enable another helpful reformulation. First of all, for $v,\ w \in \mathbb{R}^\ell$ and $f,\ g \in \mathcal{H}_2(\Omega)$, we equip the product space $\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)$ with the inner product

$$\langle (v, f), (w, g) \rangle_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)} := \langle v, w \rangle_{\mathbb{R}^\ell} + \langle f, g \rangle_{\mathcal{H}_2(\Omega)}.$$

As $\mathbb{R}^\ell$ and $\mathcal{H}_2(\Omega)$ are both complete, $\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)$ is also a Hilbert space. Note that $\|\cdot\|_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)}$ is then the induced norm. In this space, we consider the operator

$$\widetilde{T}_{\neg,\lambda\left(\delta,y^\delta\right)} \colon \mathcal{H}_2(\Omega) \to \mathbb{R}^\ell \times \mathcal{H}_2(\Omega), \qquad \widetilde{T}_{\neg,\lambda\left(\delta,y^\delta\right)} f = \begin{pmatrix} T_\neg f \\ \sqrt{\lambda\left(\delta, y^\delta\right)} f \end{pmatrix}. \tag{8.5}$$

We note the following, rather obvious aspects.

Lemma 8.1.2. *For the operator $\widetilde{T}_{\neg,\lambda\left(\delta,y^\delta\right)}$, we have the following properties with respect to the (L)RFMP algorithm.*

a) With $C(d)$ as in (8.1)*, we have for a dictionary element $d \in \mathcal{D}$*

$$C(d) = \left\| \widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} d \right\|_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)}$$

which justifies to speak of a normalization of the dictionary when considering $\widetilde{\mathcal{D}} = (C(\cdot))^{-1}\mathcal{D}$.

b) The Tikhonov functional can be re-written as

$$\mathcal{J}\left(f; T_\urcorner, \lambda\left(\delta, y^\delta\right), y^\delta\right) = \left\| (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} f \right\|^2_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)}.$$

Thus, we have

$$\begin{aligned} &\left\| (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} f_{N+1} \right\|^2_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)} \\ &\quad = \left\| (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} f_N \right\|^2_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)} - \mathrm{RFMP}(d_{N+1}; N). \end{aligned}$$

c) The approximations of two different iterations M and N ($M > N$) are related by

$$\widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} f_M = \widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} f_N + \sum_{n=N+1}^{M} \alpha_{n+1} \widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} d_n. \tag{8.6}$$

d) The coefficient of an iteration N is given by

$$\alpha_{N+1} = \left\langle (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} f_N, \widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} d_{N+1} \right\rangle_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)}. \tag{8.7}$$

Now we consider first results that will lead to the main convergence theorem of the (L)RFMP algorithm.

Lemma 8.1.3. *The sequence*

$$\left(\mathcal{J}\left(f_N; T_\urcorner, \lambda\left(\delta, y^\delta\right), y^\delta\right) \right)_{N \in \mathbb{N}_0}$$

produced by the (L)RFMP algorithm is monotonically decreasing and convergent.

Proof. The sequence is monotonically decreasing due to (8.4). Further, it is obviously bounded from below by 0. Hence, it must be convergent. □

Lemma 8.1.4. *The sequence $(\alpha_{N+1})_{N \in \mathbb{N}_0}$ produced by the (L)RFMP algorithm is square summable.*

Proof. We consider

$$\begin{aligned}\sum_{n=0}^{N} \alpha_{n+1}^2 &= \sum_{n=0}^{N} \mathrm{RFMP}(d_{n+1}; n) \\ &= \sum_{n=0}^{N} \left[\mathcal{J}\left(f_n; T_\urcorner, \lambda\left(\delta, y^\delta\right), y^\delta\right) - \mathcal{J}\left(f_{n+1}; T_\urcorner, \lambda\left(\delta, y^\delta\right), y^\delta\right)\right] \\ &= \mathcal{J}\left(f_0; T_\urcorner, \lambda\left(\delta, y^\delta\right), y^\delta\right) - \mathcal{J}\left(f_{N+1}; T_\urcorner, \lambda\left(\delta, y^\delta\right), y^\delta\right),\end{aligned}$$

where the latter equality holds due to the telescoping sum. For N to infinity, the right-hand side is convergent due to Lemma 8.1.3. □

Corollary 8.1.5. *For the sequence $(\alpha_{N+1})_{N\in\mathbb{N}_0}$ produced by the (L)RFMP algorithm, it holds*

$$\liminf_{N\to\infty} |\alpha_{N+1}| \sum_{n=0}^{N-1} |\alpha_{n+1}| = 0.$$

For the proof, see Jones (1987, Lemma 2). With these results, we are able to show the theorem needed for the convergence of the (L)RFMP algorithm.

Theorem 8.1.6. *The sequence produced by the (L)RFMP algorithm*

$$\left((y,0)^\mathrm{T} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} f_N\right)_{N\in\mathbb{N}_0}$$

is convergent.

Proof. See also Kontak (2018); Michel (2020). We show that the sequence is a Cauchy sequence. As $\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)$ is a Hilbert space, the sequence is convergent in this case. We show the Cauchy property by contradiction and, thus, assume that

$$\left((y,0)^\mathrm{T} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} f_N\right)_{N\in\mathbb{N}_0}$$

is not a Cauchy sequence, i. e.

$$\begin{aligned}&\exists \varepsilon > 0\ \forall \nu \in \mathbb{N}_0\ \exists N_1(\nu), N_2(\nu) \geq \nu : \\ &\left\| \widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} f_{N_1(\nu)} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} f_{N_2(\nu)} \right\|_{\mathbb{R}^\ell\times\mathcal{H}_2(\Omega)} > \varepsilon.\end{aligned}$$

Let $\gamma > 0$ be fixed and, with Lemma 8.1.3, let

$$L := \lim_{N\to\infty} \left\| (y,0)^\mathrm{T} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^\delta\right)} f_N \right\|_{\mathbb{R}^\ell\times\mathcal{H}_2(\Omega)}.$$

In particular, we have

$$L \leq \left\| (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_N \right\|_{\mathbb{R}^{\ell}\times\mathcal{H}_2(\Omega)} \tag{8.8}$$

for all $N \in \mathbb{N}_0$. Then there exists $\kappa \in \mathbb{N}_0$ such that

$$\left\| (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_N \right\|^2_{\mathbb{R}^{\ell}\times\mathcal{H}_2(\Omega)} < L^2 + \gamma \qquad \forall N \geq \kappa.$$

Thus, for $N_1(\kappa),\ N_2(\kappa) \geq \kappa$, we have

$$\begin{aligned} \left\| \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_1(\kappa)} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_2(\kappa)} \right\|_{\mathbb{R}^{\ell}\times\mathcal{H}_2(\Omega)} &> \varepsilon, \\ \left\| (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_1(\kappa)} \right\|^2_{\mathbb{R}^{\ell}\times\mathcal{H}_2(\Omega)} &< L^2 + \gamma \end{aligned} \tag{8.9}$$

and

$$\left\| (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_2(\kappa)} \right\|^2_{\mathbb{R}^{\ell}\times\mathcal{H}_2(\Omega)} < L^2 + \gamma.$$

Further, there exist $N_3(\kappa) \in \mathbb{N}_0$ with $N_3(\kappa) > \max(N_1(\kappa), N_2(\kappa))$ such that

$$|\alpha_{N_3(\kappa)+1}| \sum_{n=0}^{N_3(\kappa)-1} |\alpha_{n+1}| < \gamma \tag{8.10}$$

due to Corollary 8.1.5. With the triangle equality, we then obtain

$$\begin{aligned} \varepsilon &< \left\| \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_1(\kappa)} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_2(\kappa)} \right\|_{\mathbb{R}^{\ell}\times\mathcal{H}_2(\Omega)} \\ &\leq \left\| \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_1(\kappa)} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_3(\kappa)} \right\|_{\mathbb{R}^{\ell}\times\mathcal{H}_2(\Omega)} \\ &\quad + \left\| \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_3(\kappa)} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_2(\kappa)} \right\|_{\mathbb{R}^{\ell}\times\mathcal{H}_2(\Omega)}. \end{aligned} \tag{8.11}$$

We consider one of the summands in detail:

$$\begin{aligned} &\left\| \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_1(\kappa)} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_3(\kappa)} \right\|^2_{\mathbb{R}^{\ell}\times\mathcal{H}_2(\Omega)} \\ &\quad = \left\| (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_1(\kappa)} \right\|^2_{\mathbb{R}^{\ell}\times\mathcal{H}_2(\Omega)} + \left\| (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_3(\kappa)} \right\|^2_{\mathbb{R}^{\ell}\times\mathcal{H}_2(\Omega)} \\ &\qquad - 2\left\langle (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_1(\kappa)}, (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_{N_3(\kappa)} \right\rangle. \end{aligned}$$

Due to (8.6) and the bilinearity of the inner product, this equals

$$\begin{aligned}
&\left\|(y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_{N_1(\kappa)}\right\|^2_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)} + \left\|(y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_{N_3(\kappa)}\right\|^2_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)} \\
&- 2\left\langle (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_{N_3(\kappa)} + \sum_{n=N_1(\kappa)+1}^{N_3(\kappa)} \alpha_n \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} d_n, (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_{N_3(\kappa)} \right\rangle \\
&= \left\|(y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_{N_1(\kappa)}\right\|^2_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)} - \left\|(y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_{N_3(\kappa)}\right\|^2_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)} \\
&\quad - 2 \sum_{n=N_1(\kappa)+1}^{N_3(\kappa)} \alpha_n \left\langle \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} d_n, (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_{N_3(\kappa)} \right\rangle \\
&\leq \left\|(y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_{N_1(\kappa)}\right\|^2_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)} - \left\|(y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_{N_3(\kappa)}\right\|^2_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)} \\
&\quad + 2|\alpha_{N_3(\kappa)+1}| \sum_{n=N_1(\kappa)+1}^{N_3(\kappa)} |\alpha_n| \\
&\leq L^2 + \gamma - L^2 + 2\gamma \\
&= 3\gamma
\end{aligned} \tag{8.12}$$

where the first inequality holds due to $-a \leq |a|$ for real a, the triangle inequality and the maximization of (8.2) in an iteration of the (L)RFMP algorithm and the second inequality holds due to (8.9), (8.8) and (8.10) (in that order). Note that (8.12) holds analogously for $N_2(\kappa)$ instead of $N_1(\kappa)$. If we now choose $\gamma < \varepsilon^2/12$, we have

$$\begin{aligned}
&\left\|\widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_{N_1(\kappa)} - \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_{N_3(\kappa)}\right\|_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)} \\
&\quad + \left\|\widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_{N_3(\kappa)} - \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_{N_2(\kappa)}\right\|_{\mathbb{R}^\ell \times \mathcal{H}_2(\Omega)} \quad \leq 2\sqrt{3\gamma} = 2\sqrt{\varepsilon^2/4} = \varepsilon
\end{aligned}$$

which contradicts (8.11). □

With this, the convergence of the approximation is easily shown.

Theorem 8.1.7. *Let the regularization parameter $\lambda(\delta, y^\delta)$ be positive. Then the sequence of approximations $(f_N)_{N\in\mathbb{N}_0}$ produced by the (L)RFMP algorithm is convergent. We denote the limit by*

$$f_\infty := \lim_{N\to\infty} f_N.$$

Proof. We have seen that the sequence

$$\left((y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda(\delta,y^\delta)} f_N\right)_{N\in\mathbb{N}_0}$$

is convergent. As this convergence is component-by-component, we immediately see the convergence of $-\sqrt{\lambda\left(\delta, y^{\delta}\right)} f_N$ for $N \to \infty$ in the second component, see (8.5). The strictly positive regularization parameter guarantees the convergence of f_N for $N \to \infty$. □

Furthermore, we are also able to characterize this limit.

Theorem 8.1.8. *For a complete dictionary $\mathcal{D} \subseteq \mathcal{H}_2(\Omega)\backslash\{0\}$, a data vector $y \in \mathbb{R}^{\ell}$, a regularization parameter $\lambda\left(\delta, y^{\delta}\right) > 0$ and the limit f_{∞} of the sequence $(f_N)_{N\in\mathbb{N}_0}$ of approximations produced by the (L)RFMP algorithm, it holds*

$$\left(T_{\urcorner}^{*} T_{\urcorner} + \lambda\left(\delta, y^{\delta}\right) I\right) f_{\infty} = T_{\urcorner}^{*} y,$$

i. e. the limit of the sequence $(f_N)_{N\in\mathbb{N}_0}$ produced by the (L)RFMP algorithm solves the regularized normal equation.

Note that this solution is unique as we noted in Example 2.5.17.

Proof. Due to the square-summability (Lemma 8.1.4) of $(\alpha_N)_{N\in\mathbb{N}}$, we have the convergence of $\alpha_{N+1} \to 0$ for $N \to \infty$. With the formulation (8.7), this means

$$\left\langle (y,0)^{\mathrm{T}} - \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} f_N, \widetilde{T}_{\urcorner,\lambda\left(\delta,y^{\delta}\right)} d_{N+1} \right\rangle_{\mathbb{R}^{\ell}\times\mathcal{H}_2(\Omega)} \to 0 \qquad \text{for} \qquad N \to \infty$$

which is equivalent to

$$\langle y - T_{\urcorner} f_N, T_{\urcorner} d_{N+1}\rangle_{\mathbb{R}^{\ell}} - \lambda\left(\delta, y^{\delta}\right)\langle f_N, d_{N+1}\rangle_{\mathcal{H}_2(\Omega)} \to 0 \qquad \text{for} \qquad N \to \infty. \tag{8.13}$$

The (L)RFMP algorithm aims to maximize

$$\left|\langle y - T_{\urcorner} f_N, T_{\urcorner} d\rangle_{\mathbb{R}^{\ell}} - \lambda\left(\delta, y^{\delta}\right)\langle f_N, d\rangle_{\mathcal{H}_2(\Omega)}\right| \tag{8.14}$$

for $d \in \mathcal{D}$. Thus, due to the Sandwich Theorem using (8.13) and (8.14), for any $d \in \mathcal{D}$, we have

$$\langle y - T_{\urcorner} f_N, T_{\urcorner} d\rangle_{\mathbb{R}^{\ell}} - \lambda\left(\delta, y^{\delta}\right)\langle f_N, d\rangle_{\mathcal{H}_2(\Omega)} \to 0 \qquad \text{for} \qquad N \to \infty.$$

In particular, for a $d \in \mathcal{D}$, we obtain

$$\begin{aligned}
&\langle y - T_{\urcorner} f_N, T_{\urcorner} d\rangle_{\mathbb{R}^{\ell}} - \lambda\left(\delta, y^{\delta}\right)\langle f_N, d\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \langle T_{\urcorner}^{*} y - T_{\urcorner}^{*} T_{\urcorner} f_N, d\rangle_{\mathcal{H}_2(\Omega)} - \lambda\left(\delta, y^{\delta}\right)\langle f_N, d\rangle_{\mathcal{H}_2(\Omega)} \\
&\quad = \left\langle T_{\urcorner}^{*} y - \left(T_{\urcorner}^{*} T_{\urcorner} + \lambda\left(\delta, y^{\delta}\right) I\right) f_N, d\right\rangle_{\mathcal{H}_2(\Omega)} \to 0
\end{aligned}$$

for $N \to \infty$. As we have seen that $(f_N)_{N\in\mathbb{N}_0}$ converges and because $T\urcorner$ and $T\urcorner^*$ are continuous (and, in particular, bounded), we obtain that

$$\left(T\urcorner^* y - \left(T\urcorner^* T\urcorner + \lambda\left(\delta, y^\delta\right) I\right) f_N\right)_{N\in\mathbb{N}_0} \tag{8.15}$$

is bounded. Hence, for a convergent sequence $d_m \to d \in \mathcal{H}_2(\Omega)$, $m \to \infty$, of dictionary elements $d_m \in \mathcal{D}$, we can interchange the limits and have

$$\begin{aligned}&\lim_{N\to\infty}\left\langle T\urcorner^* y - \left(T\urcorner^* T\urcorner + \lambda\left(\delta, y^\delta\right) I\right) f_N, d\right\rangle_{\mathcal{H}_2(\Omega)}\\ &\quad= \lim_{m\to\infty}\lim_{N\to\infty}\left\langle T\urcorner^* y - \left(T\urcorner^* T\urcorner + \lambda\left(\delta, y^\delta\right) I\right) f_N, d_m\right\rangle_{\mathcal{H}_2(\Omega)} = 0.\end{aligned}$$

Due to the bilinearity of the inner product and the algebraic limit theorem, this holds also for a convergent sequence $(d_m)_{m\in\mathbb{N}_0} \subset \operatorname{span}\mathcal{D}$ with $d_m \to d \in \mathcal{H}_2(\Omega)$ with $m \to \infty$. Thus, the sequence (8.15) is weakly convergent to zero. Further, from Theorem 8.1.6, we already know that the (strong) limit f_∞ of the approximation exists. Hence, also the strong limit of (8.15) exists and coincides with the weak limit:

$$T\urcorner^* y - \left(T\urcorner^* T\urcorner + \lambda\left(\delta, y^\delta\right) I\right) f_\infty = 0,$$

which is equivalent to the regularized Tikhonov-Philipps normal equation. □

There are several other results for the convergence of $(f_N)_{N\in\mathbb{N}_0}$ produced by the RFMP algorithm in particular situations which all follow straight-forwardly from the last result. Hence, they can easily be transferred to the LRFMP algorithm as well. Summarized, we obtain the following results:

- the stability of the solution, i. e. for decreasing noise level (see e. g. Michel, 2015a, Theorem 7),
- the convergence of the regularization, i. e. for a particularly decreasing regularization parameter (see e. g. Michel, 2015a, Theorem 8), and
- a characterization of the solution if $\mathcal{D}$ is only a spanning set for a proper subspace of $\mathcal{H}_2(\Omega)$ (see e. g. Michel and Orzlowski, 2017, Theorem 6).

Furthermore, we consider here only the truly regularized case $\lambda(\delta, y^\delta) > 0$ as it is more relevant in practice. Nonetheless, also for the unregularized case $\lambda(\delta, y^\delta) = 0$ (the so-called FMP algorithm) similar convergence results were obtained (see e. g. Fischer, 2011; Kontak, 2018; Kontak and Michel, 2019). Note that, currently, we only know that the sequence $(f_N)_{N\in\mathbb{N}_0}$ obtained from the FMP algorithm converges if the so-called semi-frame condition holds. This assumption states that there should exist a positive constant c and an $M \in \mathbb{N}$ such that the inequality

$$c\|H\|^2_{\mathcal{H}_2(\Omega)} \le \sum_{k=1}^{\infty} \beta_k^2$$

holds true for all expansions $H = \sum_{k=1}^{\infty} \beta_k d_k$ of $H \in \mathcal{H}_2(\Omega)$ with $\beta_k \in \mathbb{R}$ and $d_k \in \mathcal{D}$ from an (over-) complete dictionary $\mathcal{D}$ and where $|\{j \in \mathbb{N} \mid d_j = d_k\}| \leq M$ for all $k \in \mathbb{N}$. Obviously, the semi-frame condition is not easily verified. However, if the sequence converges, the algorithm yields the exact solution of the inverse problem $T\neg f = y$.
Now we consider the (L)ROFMP algorithm. Obviously, we cannot make a similar a-priori normalization of the dictionary in the case of the ROFMP objective function as the denominator of ROFMP$(\cdot;\cdot)$ (confer (7.8)) depends on previous iterations of the algorithm. Further, as there exist less theoretical results for the ROFMP algorithm, we cannot expect that we have more results for the LROFMP algorithm.
First of all, similar as before, we obtain the convergence of the Tikhonov-Philipps sequence.

Lemma 8.1.9. *The sequence*

$$\left(\mathcal{J}_O\left(f_N; T\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right)\right)_{N\in\mathbb{N}_0}$$

produced by the (L)ROFMP algorithm is monotonically decreasing and convergent.

Proof. The proof follows the same argumentation as in the case of the (L)RFMP algorithm in Lemma 8.1.3. The sequence is monotonically decreasing due to (4.33) and is bounded from below by 0. Hence, it must be convergent. □

A unique result of the (L)ROFMP algorithm discusses the convergence of the residual.

Theorem 8.1.10. *The sequence of residuals $(R^N)_{N\in\mathbb{N}_0}$ produced by the ROFMP algorithm stagnates, i. e. $R^N = R^{N_0}$ for $N \geq N_0 \in \mathbb{N}_0$.*

Proof. We have either $\mathcal{P}_{\mathcal{V}_{N-1}^\perp} T\neg d_N = 0$ or $\mathcal{P}_{\mathcal{V}_{N-1}^\perp} T\neg d_N \neq 0$. Furthermore, we compute the next residual via $R^N = R^{N-1} - \alpha_N \mathcal{P}_{\mathcal{V}_{N-1}^\perp} T\neg d_N$ (confer Algorithm 4 and Algorithm 7, respectively). Thus, we have $R^N = R^{N-1}$ in the former case. Because $T\neg$ maps to the finite-dimensional space $\mathbb{R}^\ell$, it is possible that $\mathcal{P}_{\mathcal{V}_{N-1}^\perp} T\neg d_N$ vanishes while $\mathcal{V}_N \neq \mathbb{R}^\ell$ or $\mathcal{V}_N = \mathbb{R}^\ell$. In the former case, it is still possible that it holds $\mathcal{P}_{\mathcal{V}_N^\perp} T\neg d_{N+1} \neq 0$ in the next iteration. On the contrary, in the latter case, we have $\mathcal{P}_{\mathcal{V}_N^\perp} T\neg d = 0$ for all dictionary elements d and all future iterations. Thus, the residual stagnates at the latest if it holds $\mathcal{V}_N = \mathbb{R}^\ell$ for some iteration N. □

Unfortunately, the case that $R^{N_0} \neq 0$ but $R^N = R^{N_0}$ for all $N \geq N_0$ might not favourable for our approximation if it is caused by the orthogonalization procedure and not by the usual behaviour of the regularization. Note that, in this case, we are unable to retrieve the whole signal. As this situation depends on the orthogonality of the current basis element and all previously chosen basis elements,

it is sensible to reset the space $\mathcal{V}_N$ after a number of iterations to prevent the residual from stagnation in practice. Hence, the iterated ROFMP algorithm is much more relevant in practice than the ROFMP algorithm. Further, the LROFMP algorithm also generalizes the iterated ROFMP algorithm to an infinite dictionary. Nonetheless, in both cases the objective function depends on choices made in previous iterations. Therefore, the square-summability of the sequence of coefficients is still an open question. As the convergence of the approximation produced by the (L)RFMP algorithm mainly depends on this property, an equivalent proof for the case of the (L)ROFMP algorithm must follow a different routine for now.
The only result in this respect obtained in previous works on the ROFMP algorithm contained certain technical assumptions. The result and the proof can be found in Telschow (2014). However, it remains unclear which of the assumptions hold for the infinite dictionary $\mathcal{D}^{\mathrm{Inf}}$ and, thus, can be transferred to the LROFMP algorithm. In contrast to the (L)RFMP algorithm, it is also an open question what a limit of the approximation actually is. As the additional results (for instance, with respect to the stability of the solution or the convergence of the regularization) rely on this knowledge, we are also lacking such results for the (L)ROFMP algorithm.
If further theoretical results for the ROFMP algorithm will be discovered in future research, we may be able to transfer them to the learning case as well. However, the aim of this thesis was to introduce a dictionary learning algorithm for the RFMP and ROFMP algorithms and not to fill the gap in the theory of the latter one. Therefore, we leave a deeper investigation of the theoretical background of the (L)ROFMP algorithm to future research.

8.2. On the learning algorithm

It remains to consider how the learning algorithms are related to machine learning and, in particular, dictionary learning as we introduced these topics in Chapter 5 and Chapter 6.
First of all, we also notice that, by generalizing the IPMP algorithms, we simultaneously develop a very basic reinforcement learning algorithm, also named "adhoc strategy" (see e. g. Kaelbling et al., 1996). The author recognizes that the connection to reinforcement learning techniques may enable an advanced learning algorithm in future research.
Secondly, we consider the use of the data. In general, we build the LIPMP algorithm in such a way that we are able to learn a dictionary for specific input data, i. e. without the necessity of a set of training data. This enables the algorithm to produce a most suitable best basis for a certain signal. This should be convenient for spherical ill-posed inverse problems for which we only have a one-time measurement like the EGM2008. Additionally, if we have given a set of training data (e. g. from GRACE), we suggest to compute a best basis for each training input

and combine those bases to a general dictionary. This general dictionary should be able to project a suitable approximation from unknown data of the same type as the training data. For instance, let the training data be a set of 12 pairwise distinct GRACE measurements, e. g. the GRACE measurements of a year. If we run an LIPMP algorithm for each of these measurements and combine the dictionaries, we obtain a general dictionary for GRACE.
However, the use of the data can be also considered from another viewpoint. In each iteration of the LIPMP algorithm, we include the current residual. In particular, this means that we consider the "not-learnt-yet" part of the signal values y. Figuratively speaking, in each iteration, we peel off one layer of the input data. On the one hand, we produce a whole set of training data from one measurement in this way. On the other hand, we consider the input data part by part similarly as is done in online dictionary learning.
Thirdly, in Section 6.2, we already developed a well-defined objective for the LRFMP algorithm. However, it remains to investigate whether the LRFMP algorithm fulfils it.

Remark 8.2.1. By construction, the infinite dictionary $\mathcal{D}^{\mathrm{Inf}}$ used in the LRFMP algorithm is an optimal dictionary. Due to Theorems 8.1.7 and 8.1.8, the approximation produced by this algorithm converges for $N \to \infty$ and this limit is the unique solution of the Tikhonov-regularized normal equation.

This enables the following result.

Theorem 8.2.2. *Let f_∞ be the solution of the regularized normal equation with respect to the regularization parameter $\lambda(\delta, y^\delta)$ for perturbed data y^δ and the operator $T\neg$. Further, let*

$$f_\infty = \sum_{n=1}^{\infty} \alpha_n d_n$$

be a representation in a complete dictionary $\mathcal{D}^{\mathrm{Inf}}$ with decreasingly ordered values $|\alpha_n|$ for $n \in \mathbb{N}$ produced by the LRFMP algorithm with respect to the initial approximation $f_0 = 0$. Then the sequence of learnt dictionaries $\left(\mathcal{D}_N^*\left(f_0, T\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right)\right)_{N\in\mathbb{N}_0}$ as defined in Definition 6.1.1 is a sequence of well-working dictionaries.

Proof. The existence of f_∞ was noted in Remark 8.2.1. By construction, the sequence of learnt dictionaries is defined by

$$\mathcal{D}_{N+1}^*\left(f_0, T\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) = \mathcal{D}_N^*\left(f_0, T\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) \cup \{d_{N+1}\}.$$

Thus, it is a nested sequence. As the trial functions $d_1, ..., d_N \in \mathcal{D}^{\mathrm{Inf}}$ from the representation of f_∞ are contained in the learnt dictionary $\mathcal{D}_N^*\left(f_0, T\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right)$, also the second property is fulfilled. □

However, this is unfortunately only a motivationally based characterization of the dictionary. For a mathematical one, we need to consider the limit of the sequence. Naturally, such a limit would be defined by

$$\begin{aligned}\mathcal{D}^*\left(f_0, T\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) &:= \bigcup_{n=0}^{\infty} \mathcal{D}_n^*\left(f_0, T\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right) \\ &=: \lim_{n\to\infty} \mathcal{D}_n^*\left(f_0, T\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right).\end{aligned}$$

It would be interesting to investigate the properties of this limit. In particular, with respect to Section 6.2, it is an open question whether $\mathcal{D}^*\left(f_0, T\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right)$ is optimal, i. e. complete. At a first glance, it seems unlikely that this limit can span the whole space $\mathcal{H}_2(\Omega)$. As the dictionary is dependent on the operator $T\neg$, a starting point may be to investigate whether $T\neg\mathcal{D}^*\left(f_0, T\neg, \lambda\left(\delta, y^\delta\right), y^\delta\right)$ spans the image of $T\neg$. However, note that the dictionary also depends on the data y^δ. Hence, it might be more likely that

$$\bigcup_{i\in\mathbb{N}_0} T\neg\mathcal{D}^*\left(f_0, T\neg, \lambda\left(\delta, \left(y^{(i)}\right)^\delta\right), \left(y^{(i)}\right)^\delta\right)$$

spans the whole image of $T\neg$ if $\{(y^{(i)})^\delta\}_{i\in\mathbb{N}_0}$ is a basis of the image of $T\neg$. However, note that, in practice, it might be impractical to compute that many dictionaries and we have to settle for a particular subset of the image of $T\neg$ again.
Nonetheless, in this line of thought, if we consider a set of pairwise distinct training data $\{(y^{(i)})^\delta\}_{i=1,\dots,I}$ for $I \in \mathbb{N}$, it would be interesting to consider whether the dictionary continuously depends on the data. That means, if we compute

$$\mathcal{D}^*\left(f_0, T\neg, \lambda\left(\delta, \left(y^{(i)}\right)^\delta\right), \left(y^{(i)}\right)^\delta\right) \quad \text{and} \quad \mathcal{D}^*\left(f_0, T\neg, \lambda\left(\delta, \left(y^{(j)}\right)^\delta\right), \left(y^{(j)}\right)^\delta\right)$$

for $i,\ j \in \mathbb{N},\ i,\ j \leq I,\ i \neq j$ with $\|(y^{(i)})^\delta - (y^{(j)})^\delta\|_{\mathbb{R}^\ell} < \varepsilon$, are the two dictionaries similarly close as the data? Note that, for this consideration, we first have to define how to measure the distance between two dictionaries. A first approach for this could be the Hausdorff metric. However, due to the limited project time, we leave these mathematically very challenging theoretical considerations of the characterization of dictionaries to future research.
What is left to consider at this point are the criteria for our learning algorithm that we proposed in Section 6.1. Immediately, we see that the flexibility and simplicity are fulfilled by the LIPMP algorithms. The efficiency can only be checked with the respective numerical results.
At last, we additionally consider the properties of the state-of-the-art dictionary design (confer Rubinstein et al., 2010).

- efficiency II
  The learnt dictionary contains well-known trial functions which can be efficiently computed due to their analytic nature.

- localization
  We include trial functions with a varying degree of localization such that the learnt dictionary includes such basis functions whose localization matches the data.

- multiresolution
  In previous works (see e. g. Michel and Telschow, 2014), it was shown that the IPMP algorithms enable a multiresolution of the approximation if the latter one is sorted for the different trial function classes and a significant number of iterations has been performed. As our learning algorithm is defined closely to the structure of these methods, it could be expected that it inherits the multiresolution. Note that its own approximation can be represented as a multiresolution as well as the learnt dictionary enables a multiresolution in future use.

- adaptivity
  We recognize that modifying established trial functions in order to learn a dictionary may yield a more adaptive dictionary. However, in comparison to a manually chosen dictionary, the learnt dictionary is, in all probability, more adaptive to the data. This is due to the search for the most suitable basis element based on continuous optimization problems.

We see that, in general, the LRFMP algorithm also contains these aspects.

Part III.

Numerical experiments

9. Experiment setting in general

In the third part of this thesis, we present our numerical results. In particular, we consider the application of a learnt dictionary in Chapter 10 and the results of the standalone LIPMP algorithms in Chapter 11.
In this chapter, we describe the general setting of our experiments. That means, for a particular experiment, we set the parameters to the values given next if not stated otherwise in the respective section. We also consider the different data in use here: the Earth Gravitational Model 2008 (EGM2008) as well as satellite data from the Gravity Recovery And Climate Experiment (GRACE). We give a detailed explanation of the origin and the use of these data next.

9.1. The Earth Gravitational Model 2008 (EGM2008)

As a first example, we compute our data from the EGM2008 (see e. g. Pavlis et al., 2012). This is a reference model for the gravitational potential and has already been introduced at the end of Section 2.4. In particular, we compute the values of the gravitational potential $y = \mathcal{T}\neg V$ via (2.26) for discrete grid points η from a Reuter grid (see Definition A.1.2). The EGM2008 provides then the Fourier coefficients. The full model can be obtained from the National Geospatial-Intelligence Agency, Office of Geomatics (SN), EGM Development Team (2008). Here, we use only the coefficients from degree 3 to 2190 and up to order 2159. The degree 2 mainly makes up for the global shape of the Earth. We see in Figure 2.2 that if we start the model at degree 3, we also see local structures like the Andean region, the Himalayas as well as the Pacific Ring of Fire. Hence, we have data coming from a signal with strongly varying smoothness. As it is a reference model it can be interpreted as a static mean field of the gravitational potential.

9.2. The Gravity Recovery And Climate Experiment (GRACE)

The GRACE satellite mission and its successor GRACE Follow On (GRACE-FO) were launched in March 2002 and May 2018, respectively (see e. g. Flechtner et al., 2014a; NASA Jet Propulsion Laboratory, 2020; Schmidt et al., 2008; Tapley et al., 2004; The University of Texas at Austin, Centre for Space Research, 2020). The original GRACE mission is a joint project of the National Aeronautics and Space

Administration (NASA) and the German Aerospace Center. It was decommissioned in 2017 after having successfully monitored changes in the Earth's gravity field with high accuracy for 15 years.

The mission is built as a low-low satellite-to-satellite tracking (lo-lo SST) (see e. g. Freeden et al., 2002). That means, two spacecrafts fly in the same polar orbit above the Earth. For a visualization of GRACE by an artist, see Figure 9.1. The satellites are about 220 kilometres apart and started at an altitude of 500 kilometres. Both of them measure each others distance using K-band frequencies (see e. g. GFZ Potsdam, 2000) and GPS satellites give the locations of the GRACE satellites. From these data, the gravitational potential can be obtained.

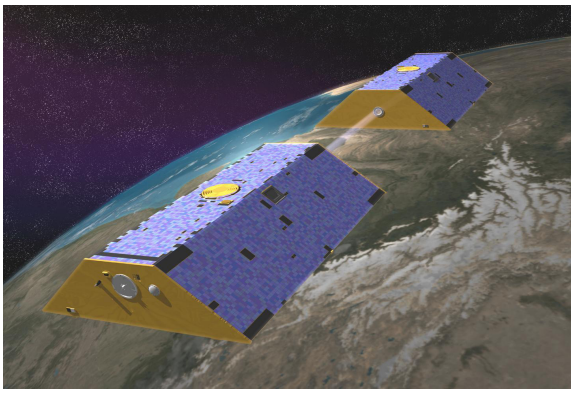


Figure 9.1.: Visualization of GRACE by an artist. Image credit: NASA/JPL-Caltech.

Diverse representations of the potential are available from the GFZ German Research Centre for Geosciences (GFZ), the Jet Propulsion Laboratory (JPL) and the Center for Space Research at the University of Texas (UTCSR).
Here, we choose the Level 2 Release 05 products which contain the coefficients of an expansion in spherical harmonics up to degree 60. We use degree 3 and higher. Hence, we can, again, use the upward continuation operator as given in (2.26). Due to Sakumura et al. (2014), we use the arithmetic mean of the data obtained from the three origins GFZ, JPL and UTCSR. The long life-span of the mission produced an enormous amount of available data. In particular, the years 2006 to 2010 contain the least data gaps (NASA Jet Propulsion Laboratory, 2020). The monthly data enable a new look on the gravitational potential. From the years of measurement, we can compute the (arithmetic) mean field of the measured potential. If we subtract this mean field from the data of each month, we obtain the monthly deviation. This is interesting because it allows us to have a look on the mass transports on the Earth.
However, even for these high-end products, the satellite tracks are usually visible. There exist profound methods to erase these remainders (see e. g. Davis et al., 2008; Klees et al., 2008; Kusche, 2007). As we are concerned with a novel approximation algorithm for spherical inverse problems and not primarily with the discovery of geophysical insights here, we abstain from such high-profile approaches and simply smooth the obtained data by means of the Cubic Polynomial Scaling Function (see Example 3.3.13) as was done in Fengler et al. (2007). In particular, we multiply each summand in (2.26) with $\kappa(2^{-5}n)^2$ for the respective degree $n = 3, ..., 60$.

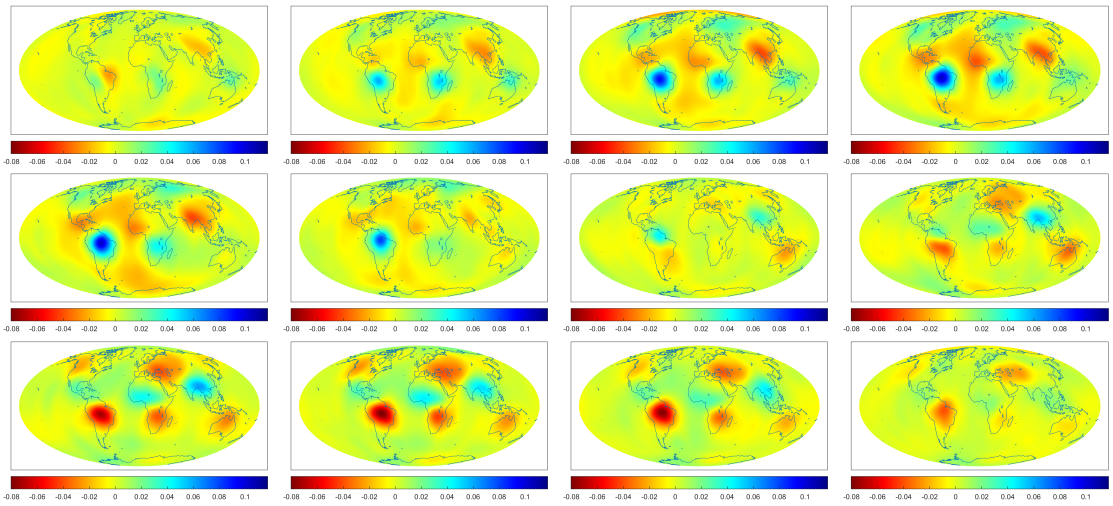


Figure 9.2.: Deviation of the mass transport on the Earth in 2008 captured by the GRACE satellite mission (left to right: first row: January to April, second row: May to August, third row: September to December). Note that the scales were adapted to improve the visualization. All values in m^2/s^2.

The monthly deviations from the mean field in January to December 2008 measured by GRACE are shown in Figure 9.2. We see, in particular, the change of the wet and dry season in the Amazon basin as well as the emerging and lessening summer on the Northern hemisphere. Note that we use reversed (in comparison to the EGM2008 Figure 2.2) colours in order to emphasize wetter soil with blue colour. Further, we equalize the scales for all months in order to provide a better comparability. At last, note that the presented monthly differences obtained from the GRACE data differ strongly from the EGM2008 with respect to their structure. The monthly GRACE data contain overall much less structure and attain much smaller values. In this way, it also provides an interesting new challenge for the novel LIPMP algorithms.

9.3. Further experiment setting

Point grids For the evaluation of our data, we use a Reuter grid (see Definition A.1.2) with $G = 100$. Thus, we work with $\ell = 12684$ grid points. For considering the approximation error, we compute the solutions at a Driscoll-Healy grid (see Definition A.1.1) with $G_\phi = 361$ and $G_\theta = 181$, i. e. 65341 grid points.

Ill-posedness of the problem We consider satellite data with a satellite height of 500 km. Note that, for computing the arithmetic mean of the monthly GRACE data, we adapt the satellite height slightly in the latest experiments such that we

obtain data on the same relative satellite orbit. Furthermore, we add 5% Gaussian noise to each data point. That means, for a perfect value y_i, $i = 1, ..., \ell$, we compute the perturbed data y_i^δ via

$$y_i^\delta = y_i \cdot (1 + 0.05 \cdot \epsilon_i), \qquad i = 1, ..., \ell, \tag{9.1}$$

for a Gaussian distributed random number ϵ_i. For the penalty term of the Tikhonov functional, we use the Sobolev norm corresponding to the Sobolev space $\mathcal{H}_2(\Omega)$ (see Definition 3.2.1). Furthermore, we approximate the inner products (3.23), (3.24) as well as (3.26) with the first 600 summands. At last, the choice of the regularization parameter is of major importance for the results. In general, the more perfectly the value is chosen, the better the expected results are. However, for the determination of a good parameter, usually several ones have to be tested. Hence, the search for a perfect value is very time-consuming. As we pointed out in Section 4.5, Task 3, determining an optimal parameter choice strategy for the (L)IPMP algorithms is a challenge for future research. Thus, for our experiments, we test different magnitudes for the regularization parameter and choose the one which minimized the relative root means square error (RMSE) after the relative data error has fallen below or equal to the noise level (see also termination criteria). The chosen values will be given with the respective experiments.

Termination Each (L)IPMP algorithm terminates if the relative data error falls equal to or below 5%, i. e. the noise level. As safety criteria, we also terminate the algorithm after 1000 iteration or if the relative data error rises above 200%. When we run an orthogonal variant of the (L)IPMP algorithms, we execute the restart procedure after 100 iterations.

Learning The optimization problems in the LIPMP algorithms are solved using methods from the NLopt library (see Johnson, 2019). In particular, for determining a global solution, we use the implementation ORIG_DIRECT_L of the DIRECT algorithm (see Appendix A.4.2). For refining this solution, we use the local SLSQP approach (see Appendix A.4.3). For both methods, we narrow the inequality constraints by 10^{-8} in order to avoid numerical problems at the boundaries. In particular, this means, we have

$$c \in \left[-1 + 10^{-8}, 1 - 10^{-8}\right], \ \beta \in \left[10^{-8}, \pi - 10^{-8}\right] \text{ and } \alpha, \ \gamma \in \left[10^{-8}, 2\pi - 10^{-8}\right]$$

for the Slepian functions and

$$\|x\|_{\mathbb{R}^3}^2 \leq 1 - 10^{-8}$$

for the Abel–Poisson low and band pass filters, confer (7.13), (7.14), (7.15), (7.16), (7.17) and (7.18), respectively. Note that this sharpening of constraints is advised in the documentation of the NLopt library. Additionally, in the case that the solution of the optimization process is infeasible, we do not consider it as a candidate.

Moreover, we set additional termination values for the optimization methods as it is advised. We limit the absolute tolerance of the change in the objective function between two successive iterates to 10^{-8}. Analogously, the absolute tolerance between two successive iterates shall be at least of the same value. We also limit the number of function evaluations to 5000 and the time spent in one optimization to 200 seconds.

Features Further, when running an LIPMP algorithm, we utilize the following features (see Section 7.3). We set the size ε of the splines for the LROFMP algorithm to $5 \cdot 10^{-4}$. We choose a constant regularization parameter and demand that the learnt dictionary is only applied with the same regularization parameter. We allow all considered trial function classes in all iterations. However, we apply the learnt dictionary only iteratively as described in Remark 7.3.4. Moreover, we do without any further restart procedure (confer Remark 7.3.5). We do so in order to present results obtained in a setting that is in accordance with the presented theory of Chapter 8. However, note that this will probably lead to a higher runtime of the learning algorithm. We tried to optimize our implementation with respect to the runtime, but there are surely improvements possible. Nonetheless, we will discuss the runtime for certain experiments in the sequel.

Dictionaries in use We work with all four trial function classes presented in this thesis: spherical harmonics, Slepian functions as well as Abel–Poisson low and band pass filters. By construction, the LIPMP algorithms choose from the infinite dictionary as given in (7.3). However, as we explained in Chapter 7 (see e. g. Algorithm 5), we need to define a starting dictionary. In the DIRECT algorithm (see Appendix A.4.2), which we use for determining a global solution of each optimization problem, the starting solutions of the Slepian functions as well as the Abel–Poisson low and band pass filters will not be used. Thus, we can keep the starting dictionary very small with respect to these trial function classes. Note that the starting dictionary contains all considered spherical harmonics. Hence, we define the starting dictionary as follows.

$$
\begin{aligned}
[N^{\mathrm{s}}]_{\mathrm{SH}} &= \left\{ Y_{n,j} \mid n = 0, ..., 100;\ j = -n, ..., n \right\} \\
[S^{\mathrm{s}}]_{\mathrm{SL}} &= \Big\{ g^{(k,5)} \left(\left(c, A(\alpha, \beta, \gamma)\varepsilon^3 \right), \cdot \right) \Big| \\
&\qquad c \in \left\{ \frac{\pi}{4}, \frac{\pi}{2} \right\},\ \alpha \in \left\{ 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2} \right\},\ \beta \in \left\{ 0, \frac{\pi}{2}, \pi \right\}, \\
&\qquad\qquad \gamma \in \left\{ 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2} \right\},\ k = 1, ..., 36 \Big\} \\
[B_K^{\mathrm{s}}]_{\mathrm{APK}} &= \left\{ \frac{K(x, \cdot)}{\|K(x, \cdot)\|_{\mathrm{L}^2(\Omega)}} \;\middle|\; |x| = 0.94,\ \frac{x}{|x|} \in X^{\mathrm{s}} \right\}
\end{aligned}
$$

$$[B_W^{\mathrm{s}}]_{\mathrm{APW}} = \left\{ \frac{W(x,\cdot)}{\|W(x,\cdot)\|_{\mathrm{L}^2(\Omega)}} \;\middle|\; |x| = 0.94,\ \frac{x}{|x|} \in X^{\mathrm{s}} \right\}$$

$$\mathcal{D}^{\mathrm{s}} = [N^{\mathrm{s}}]_{\mathrm{SH}} + [S^{\mathrm{s}}]_{\mathrm{SL}} + [B_K^{\mathrm{s}}]_{\mathrm{APK}} + [B_W^{\mathrm{s}}]_{\mathrm{APW}}$$

with a Reuter grid X^{s} with $G = 10$ (i. e. 123 grid points). All in all, this starting dictionary contains 13903 trial functions.
In certain experiments, we will compare the results of an LIPMP algorithm with the results of the respective IPMP algorithm. Note that we abstain from comparisons with classical approaches like spline approximation as those were made for the IPMP algorithms in previous publications (see e. g. Fischer and Michel, 2013b; Michel and Telschow, 2014; Leweke, 2018). The reference run of the respective IPMP algorithm needs a manually chosen dictionary. We define this dictionary similarly to those of previous publications. Note that we are aware that this approach may be arguable. A more independent ansatz would be to run an IPMP algorithm with a set of randomly chosen dictionaries and compare our results with the best one among them. Such a set of random dictionaries would need to be sufficiently large in order to be meaningful. Then, however, this ansatz is not practical due to its runtime. Note that this thesis emerged, in particular, because this strategy cannot seriously be put into practice. Thus, we get back to our experience. In a way, such a randomized search of dictionaries was already done manually in the development of the IPMP algorithms. Hence, we build our manually chosen dictionary in analogy to one from Telschow (2014). In particular, we choose the same scales for the Abel–Poisson low pass filters. However, for the centres, we choose a smaller Reuter grid due to reasons of memory capacity. Further, we also use these scales and centres for the Abel–Poisson band pass filters. Note that these trial functions were not included in the dictionary in Telschow (2014).
It follows that we compare the results of the LIPMP algorithms with a possibly already well-done dictionary choice. Our manually chosen dictionary is defined as follows:

$$\begin{aligned}
[N^{\mathrm{m}}]_{\mathrm{SH}} &= \left\{ Y_{n,j} \;\middle|\; n = 0,...,25;\ j = -n,...,n \right\} \\
[S^{\mathrm{m}}]_{\mathrm{SL}} &= \Big\{ g^{(k,5)} \left(\left(c, A(\alpha,\beta,\gamma)\varepsilon^3 \right), \cdot \right) \Big| \\
&\qquad c \in \left\{ \frac{\pi}{4}, \frac{\pi}{2} \right\},\ \alpha \in \left\{ 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2} \right\},\ \beta \in \left\{ 0, \frac{\pi}{2}, \pi \right\}, \\
&\qquad\qquad \gamma \in \left\{ 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2} \right\},\ k = 1,...,36 \Big\} \\
[B_K^{\mathrm{m}}]_{\mathrm{APK}} &= \left\{ \frac{K(x,\cdot)}{\|K(x,\cdot)\|_{\mathrm{L}^2(\Omega)}} \;\middle|\; |x| \in Z,\ \frac{x}{|x|} \in X^{\mathrm{m}} \right\}
\end{aligned}$$

$$[B_W^{\mathrm{m}}]_{\mathrm{APW}} = \left\{ \frac{W(x,\cdot)}{\|W(x,\cdot)\|_{\mathrm{L}^2(\Omega)}} \;\middle|\; |x| \in Z,\ \frac{x}{|x|} \in X^{\mathrm{m}} \right\}$$

$$\mathcal{D}^{\mathrm{m}} = [N^{\mathrm{m}}]_{\mathrm{SH}} + [S^{\mathrm{m}}]_{\mathrm{SL}} + [B_K^{\mathrm{m}}]_{\mathrm{APK}} + [B_W^{\mathrm{m}}]_{\mathrm{APW}}$$

with a Reuter grid X^{m} with $G = 60$ (i. e. 4551 grid points) and

$$Z = \{0.75,\ 0.80,\ 0.85,\ 0.89,\ 0.91,\ 0.93,\ 0.94,\ 0.95,\ 0.96,\ 0.97\}.$$

All in all, the manually dictionary contains 95152 trial functions.

Presented results We demonstrate our results from different points of view. However, we concentrate on the most prominent aspects for each experiment. That means, in most cases, we choose only some of the representations described next. As we pointed out in Appendix B, we implemented our code in C/C++. The visualization of the results presented in the next chapters, however, was done with MATLAB®. We have to run the reference tests with the manually chosen dictionary of 95152 trial functions on a fat-node with 32 CPUs and 512 GB RAM. The respective experiments with the LIPMP algorithm with all four trial functions can be run on a node with 12 CPUs and 48 GB RAM.
Each algorithm presented in this thesis yields an approximation of the solution to an inverse problem. We visualize these approximations either as they are or by considering the absolute approximation error, i. e. the absolute value of the difference between the approximation and the respective solution. Both functions are evaluated at the mentioned Driscoll-Healy grid and displayed on a two-dimensional projection of the sphere. Note that, if we compare the absolute approximation errors of different results, we scale the plots for an improved visibility. With respect to representing the errors, we give the relative data error, i. e.

$$\frac{\left\|R^N\right\|_{\mathbb{R}^\ell}}{\left\|R^0\right\|_{\mathbb{R}^\ell}},$$

as well as the relative root mean square error (RMSE)

$$\frac{\sqrt{\frac{\sum_{i=1}^{65341}\left(f_N\left(\tilde{\eta}^{(i)}\right)-f\left(\tilde{\eta}^{(i)}\right)\right)^2}{65341}}}{\sqrt{\sum_{i=1}^{65341}\left(f\left(\tilde{\eta}^{(i)}\right)\right)^2}}$$

along the iterations. With respect to the relative RMSE, f gives a representation of the solution, f_N the current approximation and $\tilde{\eta}^i$ is a point from the used Driscoll-Healy grid (see above).
Besides the visualization of approximation errors, we also summarize the results of an experiment, possibly in a table, for better comparison. There, we consider

the size of the used dictionary, the number of performed iterations, the final relative RMSE and data error, the maximal spherical harmonic degree in the dictionary as well as, in some cases, the needed CPU-runtime. Note that the maximal spherical harmonic degree of a learnt dictionary is the maximal learnt spherical harmonic degree. Further, note that the size of a learnt dictionary is always given as a "less or equal than" size. To be specific, the size of the dictionary we state is the number of iterations performed in the respective LIPMP algorithm. That is, we may have duplicates in the learnt dictionary even though this is very unlikely for the Abel–Poisson low and band pass filters as well as the Slepian functions. At last, we compare the runtime of an IPMP algorithm using the manually chosen dictionary and the learnt dictionary. In particular, we compare the runtime for the excessive preprocessing and application of the manually chosen dictionary with the learning and application of the learnt dictionary. Note that we certainly tried our best to optimize our implementation with respect to the runtime. Nonetheless, the given values here should be more viewed in relation to each other than as precise values.

10. Comparisons with the IPMP algorithms

In this chapter, we consider experiments in which we compare the IPMP algorithms with a manually chosen and a learnt dictionary. In particular, we consider the downward continuation of the EGM2008 and one month of GRACE data. At last, we present an experiment in which we learn a dictionary from one year of GRACE data and apply it to unseen test data.

10.1. Previously published results

In Michel and Schneider (2020), first results of a simpler form of the LRFMP have already been published.

Particular setting The setting of the experiments in this paper differ from Chapter 9 in the following way.
First of all, a dictionary is learnt only from a variant of the LRFMP algorithm working with spherical harmonics and Abel–Poisson low pass filters. The starting dictionary is given in the publication. Furthermore, only a one-step optimization is done. That means, we do not divide the search into a global and a local optimization. Furthermore, we use a different optimization software, the so-called IPOPT solver (see e. g. Vigerske et al., 2016; Wächter, 2002; Wächter and Biegler, 2005a,b, 2006, confer also Appendix A.4.4) with subroutines from HSL (2018). Though this method is theoretically globally convergent, due to Vigerske et al. (2016), the available implementation does not necessarily fulfil this. Particular settings of its parameters are described in Michel and Schneider (2020).
Moreover, in these experiments, we include all additional features described in Section 7.3, paragraph 2. A detailed description of those is also given in Michel and Schneider (2020). The publication shows that the features have a positive effect on guiding the learning process in this setting. However, it appears that they are also needed due to the one-step optimization.
At last, note that the experiments in Michel and Schneider (2020) terminate when they reach 3000 iterations as this is one of the predefined termination criteria. The regularization parameter is chosen as 10^{-2} for the experiments with the manually chosen dictionary. For the experiments with the learnt dictionary, we use $10^{-4}\|y\|_{\mathbb{R}^\ell}/(n+1)$, when working with the EGM2008, and $10^{-1}\|y\|_{\mathbb{R}^\ell}/(n+1)$

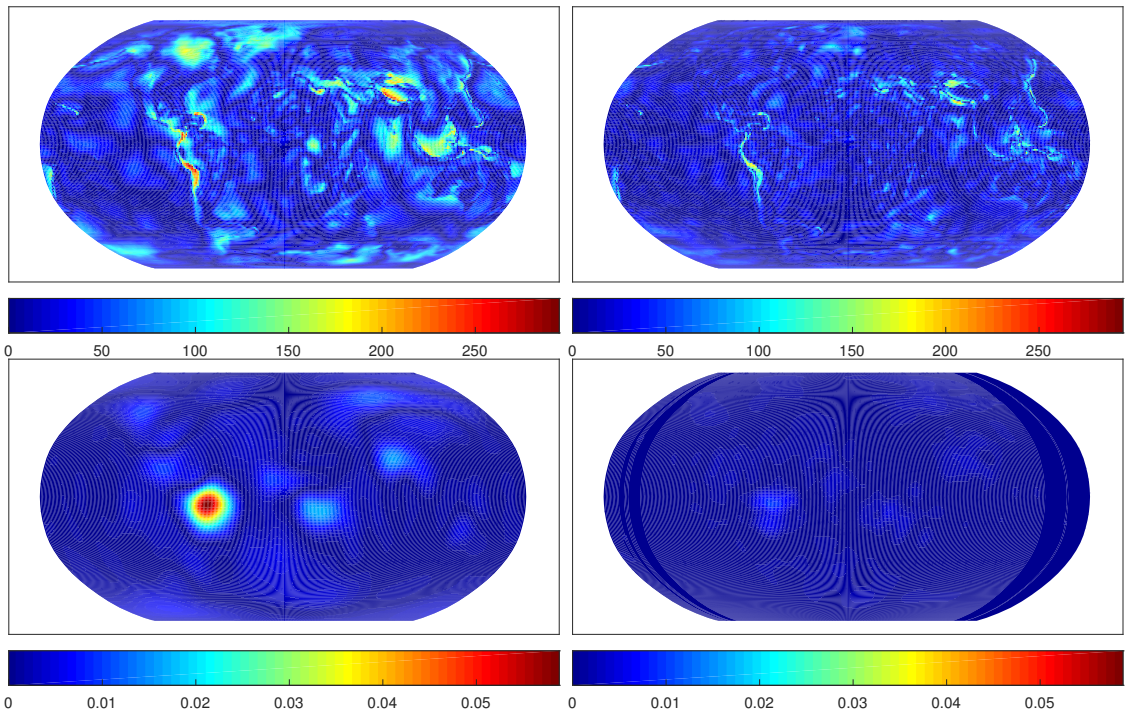


Figure 10.1.: Results of the RFMP with a manually chosen dictionary and the learnt dictionary. Upper row: Results for EGM2008 data. Lower row: Results for GRACE data. Left: Absolute approximation error of RFMP with a manually chosen dictionary. Right: Absolute approximation error of RFMP with learnt dictionary. 3000 iterations allowed in all experiments. The scale is adapated for improved comparability. All values in $\mathrm{m}^2/\mathrm{s}^2$.

in the case of GRACE data. In both cases, we have the current iteration $n = 0, ..., 3000$.

Results Figure 10.1 shows the results of this experiment obtained by the RFMP algorithm. In particular, we show the obtained absolute approximation errors of the RFMP algorithm using the manually chosen dictionary $\mathcal{D}^{\mathrm{m}} = [N^{\mathrm{m}}]_{\mathrm{SH}} + [B_K^{\mathrm{m}}]$ and a learnt dictionary. Note that we use the solution as given in Figure 2.2 and Figure 9.2 (second row, first column). Table 10.1 lists the different error values at termination as well as the CPU-runtime for a better comparison. Note that the number of learnt dictionary elements coincides with the number of iterations this variant of the LRFMP algorithm performed.

Evaluation As we point out in Michel and Schneider (2020), we first notice that in both cases the algorithm is able to construct an approximation and the remaining errors lie within regions of higher local structures. In particular, they can be found in the Andes, the Himalayas and at the Pacific Ring of Fire. More-

	$\#\mathcal{D}$	relative RMSE	relative data error	CPU-runtime in h
RFMP*	46186	0.000794	0.065830	299.82
RFMP**	≤ 3000	0.000455	0.047092	41.34
RFMP*	46186	0.001461	0.057765	295.49
RFMP**	≤ 3000	0.000306	0.046404	7.5

Table 10.1.: Comparison of RFMP with a manually chosen dictionary (RFMP*) and a learnt dictionary (RFMP**). Upper comparison with respect to EGM2008 data. Lower comparison with respect to GRACE data. 3000 iterations allowed in all experiments.

over, we see that the algorithm is able to construct a better approximation with the learnt dictionary. This also holds true for regions with local anomalies. Further, this is underlined by the smaller data and approximation error. At last, we see that learning a dictionary and applying it takes here much less runtime than preprocessing the huge manually chosen dictionary. Thus, in this setting, we obtain overall better results when learning a dictionary and applying it than if we choose manually a dictionary from spherical harmonics and Abel–Poisson kernels. However, note that we used a few also manually chosen additional features to support a better learning result.

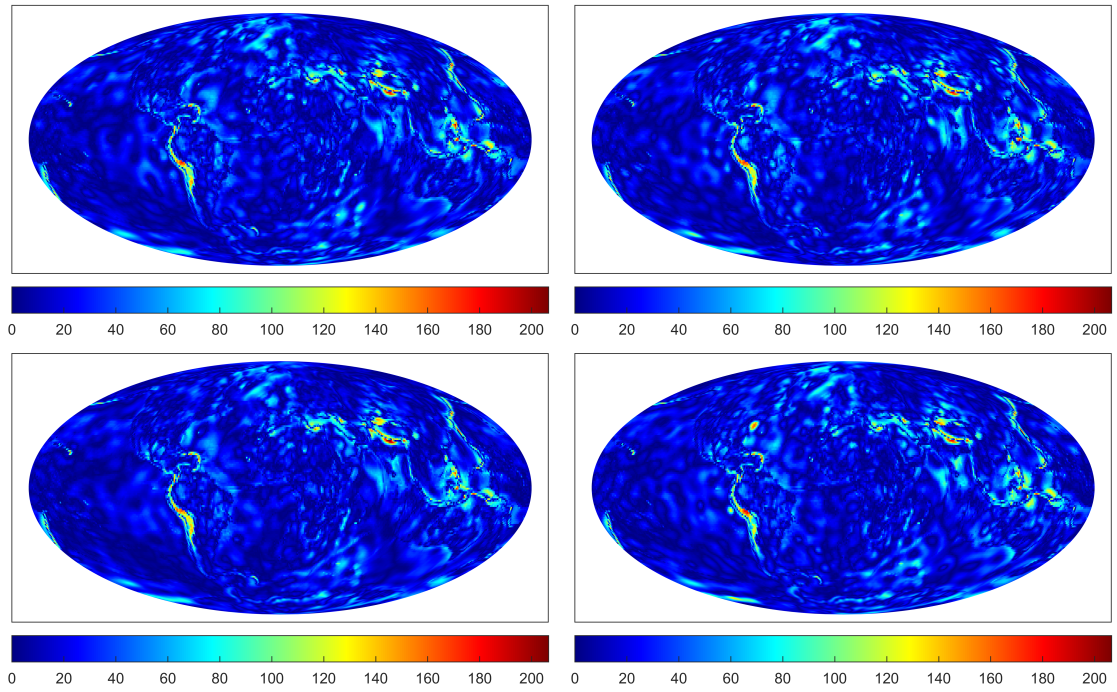
10.2. Downward continuation of regularly distributed global data

In this experiment, we compare the results with respect to the downward continuation of perturbed simulated satellite data of the respective IPMP algorithm with a learnt and a manually chosen dictionary of all presented four trial functions and with a two-step optimization.

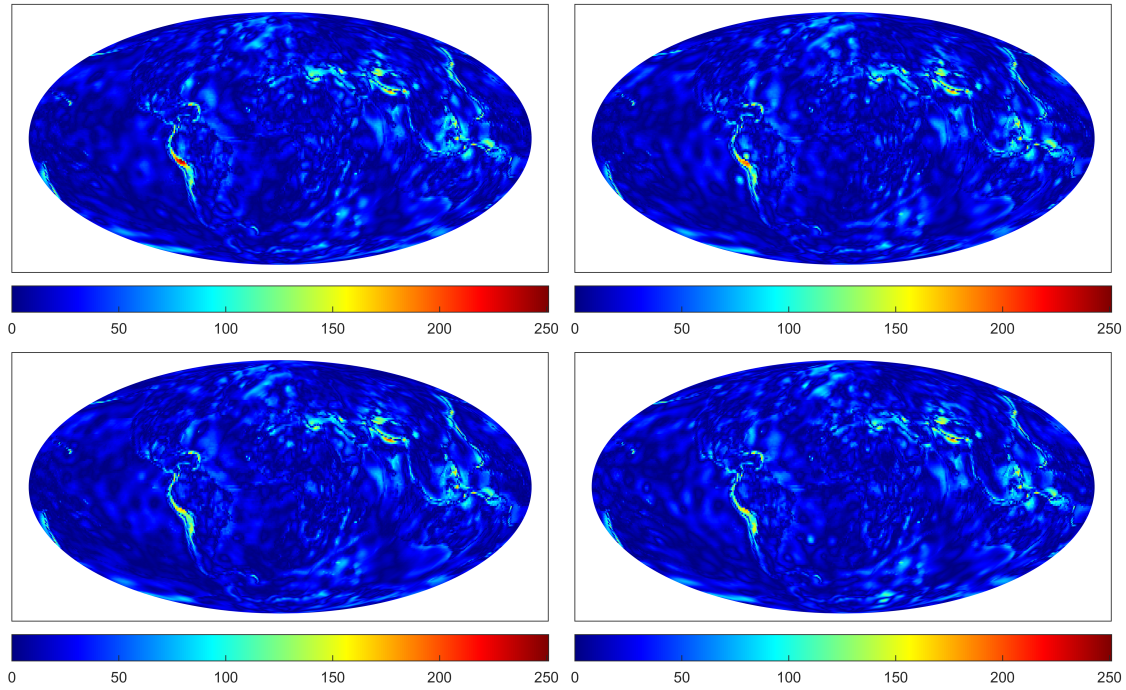
Particular setting We present results of the manually chosen dictionary and of a learnt dictionary of the LIPMP algorithms as described in Chapter 7. Further, we consider a dictionary that is learnt by an LIPMP algorithm using a non-stationary regularization parameter (in short: non-stationary learnt) and a dictionary that is learnt from only spherical harmonics as well as Abel–Poisson low and band pass filters (and again with a constant regularization parameter; in short: learnt-without-Slepian-functions). The regularization parameter is chosen as $10^{-9}\|y\|_{\mathbb{R}^\ell}$ for all (L)IPMP algorithms except when we use a non-stationary regularization parameter. In this case, we choose $\lambda_n = 10^{-6}\|y\|_{\mathbb{R}^\ell}/n$ for the iteration $n \geq 1$. The choices for these magnitudes were made as they minimize the approximation error if the algorithms terminate when the relative data error is below or equal to the noise level.

Results Figure 10.2 shows the results of this experiment obtained by the RFMP and the ROFMP algorithm, respectively. In particular, we show the obtained absolute approximation errors from the IPMP algorithms with the manually chosen (left, upper row), the learnt (right, upper row), the non-stationary learnt (left, lower row) and the learnt-without-Slepian-functions (right, lower row) dictionary (Figure 10.2a for the RFMP and Figure 10.2b for the ROFMP algorithm). Note that we use the solution as given in Figure 2.2. Further, we give the relative data errors and the relative RMSEs along the iterations (Figure 10.3). In the first two rows, Table 10.2 lists the different error values at termination as well as the size of the used dictionary, the needed number of iterations, the maximal spherical harmonic degree and the CPU-runtime for a better comparison. Note that, in the cases where any learnt dictionary is used, the maximal spherical harmonic degree is the learnt maximal degree.

Evaluation In the beginning, we note that these are the first results ever published of the IPMP algorithms when using Slepian functions and that is also without the learning technique. Thus, first of all, we notice that the algorithms are again able to construct a good approximation and the remaining errors lie only in regions with higher local structure. That means, the remaining errors lie within regions where they can be expected if one takes into consideration that we use perturbed upward continued data on a satellite orbit of 500 km. Moreover, in comparison to the results of the RFMP algorithm in Section 10.1, we see that using these four trial function classes improves the results. In particular, with a roughly doubled manually chosen dictionary, we obtain much lower relative data and approximation errors with less than a third of the iterations in the IPMP algorithms. Hence, incorporating more diverse trial functions in the dictionary improves the results of the IPMP algorithms.
Further, in Figure 10.2, we see that the approximation obtained using the manually chosen and any learnt dictionary do not differ that significantly as before. This is underlined by the values given in Table 10.2. If we terminate the algorithms at noise level, we obtain very similar relative RMSEs in all cases. With respect to the different regularization parameters and number of used trial function classes, it is difficult to make a general statement here. From the values we see here, we could assume that using a non-stationary regularization parameter may improve the approximation error slightly while having a higher runtime. On the contrary, using less trial function may lead to a slight loss in the approximation but is undoubtedly faster. In anticipation of the next experiment (i. e. compare with the last two rows in Table 10.2), such assumptions may also depend on the data. It would be interesting to investigate this behaviour in view of statistically relevant evidence in future research. Nonetheless, here we see that all of the different settings for learning a dictionary provide a suitable result.
Besides the relative RMSEs, we note that the CPU-runtimes are also very similar for the RFMP algorithm if we use either a manually chosen dictionary or a learnt dictionary of all four trial function classes. However, if we do without



(a) Absolute approximation errors obtained by the RFMP algorithm.



(b) Absolute approximation errors obtained by the ROFMP algorithm.

Figure 10.2.: Absolute approximation errors of the experiment described in Section 10.2. In both subfigures, the IPMP algorithm uses the manually chosen (left, upper row), the learnt (right, upper row), the non-stationary learnt (left, lower row) and the learnt-without-Slepian-functions (right, lower row) dictionary. The colour scale is adapted for better comparability. All values in $\mathrm{m}^2/\mathrm{s}^2$.

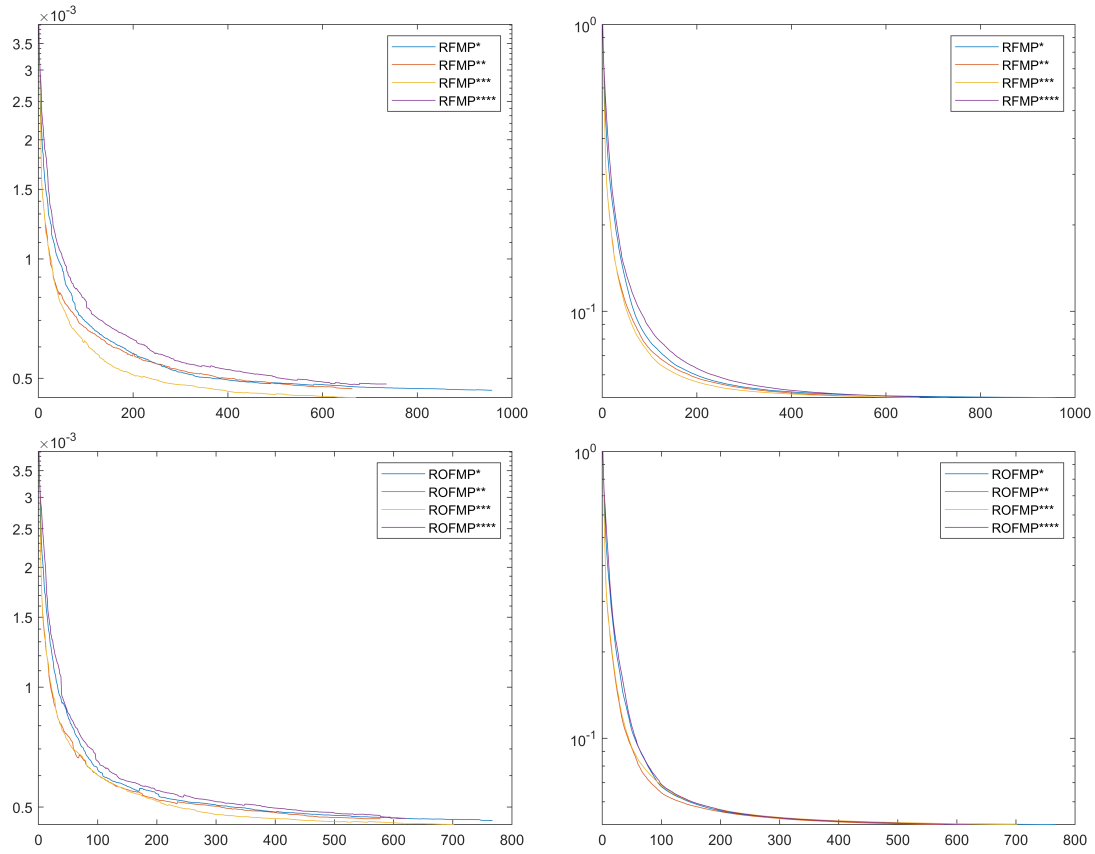


Figure 10.3.: The relative RMSE (left column) and relative data error (right column) of the RFMP (upper row) and of the ROFMP (lower row), respectively, with respect to the experiment described in Section 10.2. IPMP* uses the manually chosen, IPMP** the learnt, IPMP*** the non-stationary learnt and IPMP**** the learnt-without-Slepian-functions dictionary. The x-axis denotes the iterations and the y-axis the logarithm of the error values.

the Slepian functions, we obtain a similarly good approximation but at a much lower runtime (about only 25%). Unfortunately, this does not fully take over to the ROFMP algorithm. We obtain again an equally suitable approximation without Slepian functions at a much lower runtime. However, if we learn Slepian functions as well, the runtime is, in particular, for the case of a non-stationary regularization parameter, significantly higher in this experiment. Note that it is well-known that the orthogonalization process yields higher runtimes also in the non-learning case. Further, in the LROFMP algorithm, the projection coefficients must be recomputed for the current ROFMP step in each iteration of the optimization processes. In contrast, in the ROFMP algorithm, these values can be updated. Thus, besides taking Slepian functions into account, the recomputation could also be a reason for the higher runtime. However, again, we anticipate the

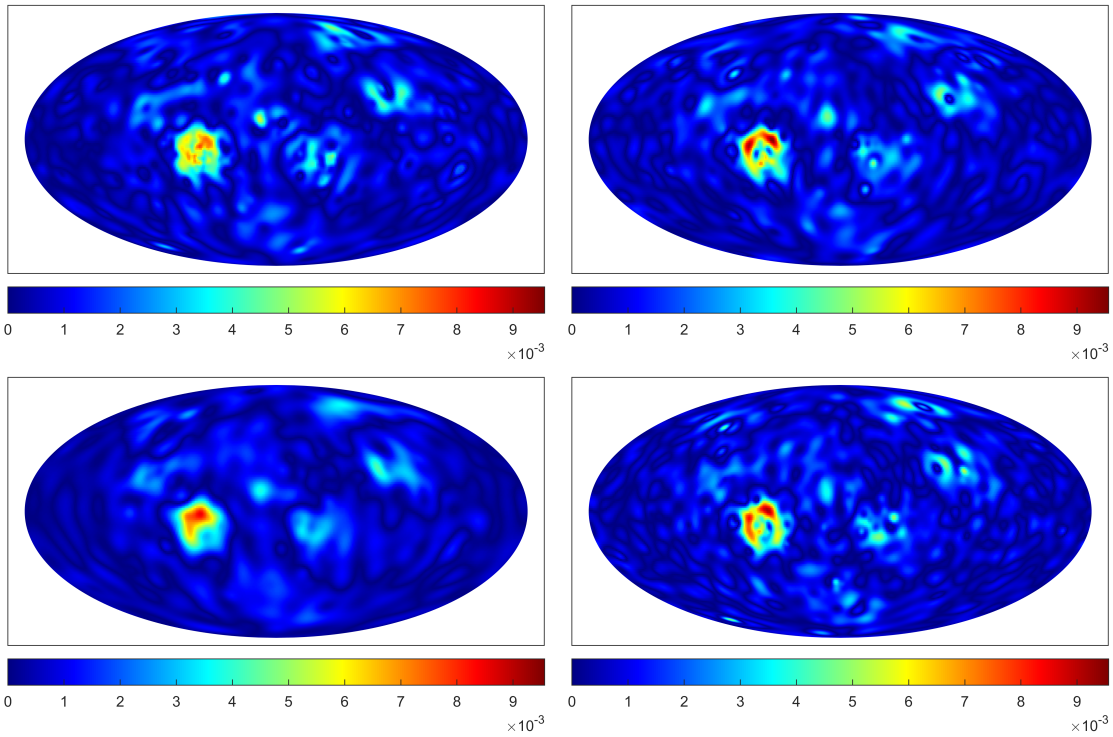
next experiment (i. e. compare with the last two rows in Table 10.2). There we see that the runtimes of learning and applying the learnt dictionary are generally lower than using the manually chosen dictionary. Hence, the runtime could also be heavily influenced by the structure of the data itself as data with more structure may demand more iterations. Note that the runtime of the IPMP algorithms is mostly caused by the size of the manually chosen dictionary, whereas the runtime of the learning algorithms is heavily influenced by solving the optimization problems.
Moreover, in Table 10.2, we see that we are indeed able to learn a maximal spherical harmonic degree as the chosen degrees are generally below 100 (compare with the starting dictionary). Note that, however, they are higher than the maximal spherical harmonic degree used in the manually chosen dictionary. Furthermore, the size of the learnt dictionary is about 1% of the size of the manually chosen dictionary. Note that the IPMP algorithms also need less iterations when using the learnt dictionary. In a certain sense, it produces a sparser approximation. At last, recall that, for using a manually chosen dictionary of this size, a compute node of more than 100 GB RAM is necessary whereas the learning process and the application of the learnt dictionary can be computed with less than 48 GB RAM. Summarized, we can say that learning a dictionary with the LIPMP algorithm enables us to obtain equally good approximations at often less computational costs.

10.3. Downward continuation of monthly Data

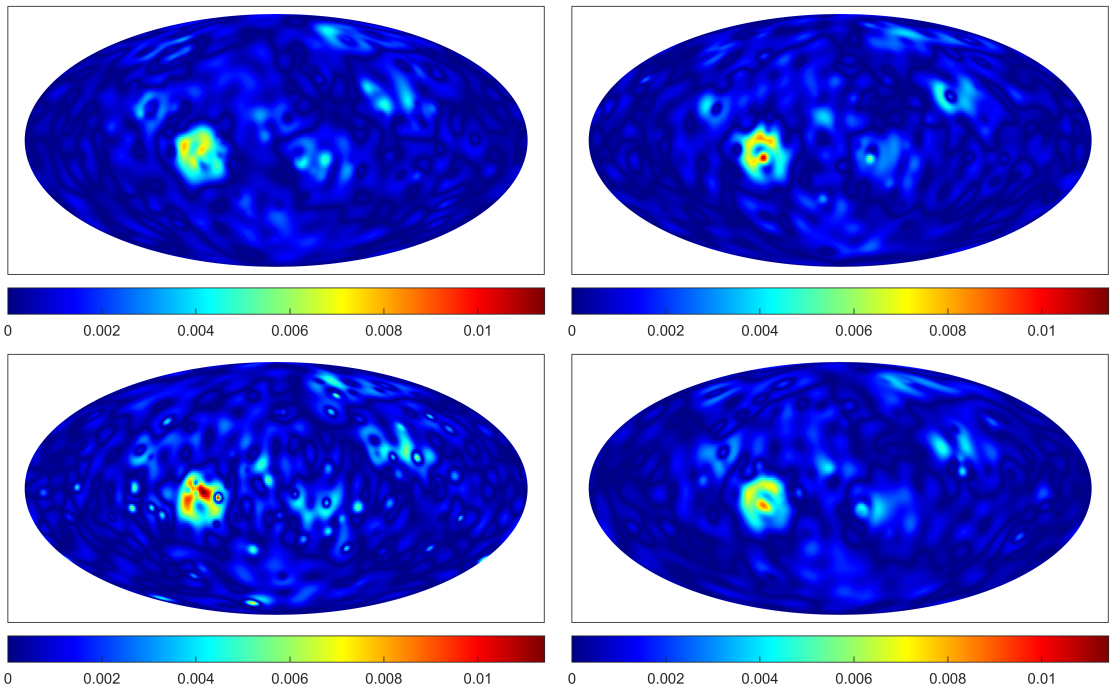
This experiment is analogous to Section 10.2 but with one month of GRACE data. Note that, as seen in Figure 9.2 (second row, first column), this data is of very different structure than the EGM2008.

Particular setting We remain with the setting as described in Chapter 9. Our data values here are given by the GRACE monthly deviation in May 2008 (confer Figure 9.2, second row, first column). Analogously as in Section 10.2, we run the IPMP algorithms with the manually chosen, a learnt, a non-stationary learnt and a learnt-without-Slepian-functions dictionary. The regularization parameter is chosen as $10^{-4}\|y\|_{\mathbb{R}^\ell}$ for all (L)IPMP algorithms except when we use a non-stationary regularization parameter. Then we choose, for the iteration $n \geq 1$, $\lambda_n = 10^{-1}\|y\|_{\mathbb{R}^\ell}/n$ for the LRFMP and $\lambda_n = 10^{-3}\|y\|_{\mathbb{R}^\ell}/n$ for the LROFMP algorithm.

Results The results obtained by the RFMP and the ROFMP algorithm, respectively, in this experiment are shown in Figure 10.4. In particular, we give the absolute approximation errors obtained from the IPMP algorithms using the manually chosen (left, upper row), the learnt (right, upper row), the non-stationary learnt



(a) Absolute approximation error obtained by the RFMP algorithm.



(b) Absolute approximation error obtained by the ROFMP algorithm.

Figure 10.4.: Approximation errors of the experiment described in Section 10.3. In both subfigures, the IPMP algorithm uses the manually chosen (left, upper row), the learnt (right, upper row), the non-stationary learnt (left, lower row) and the learnt-without-Slepian-functions dictionary (right, lower row). The colour scale is adapted for better comparability. All values in m^2/s^2.

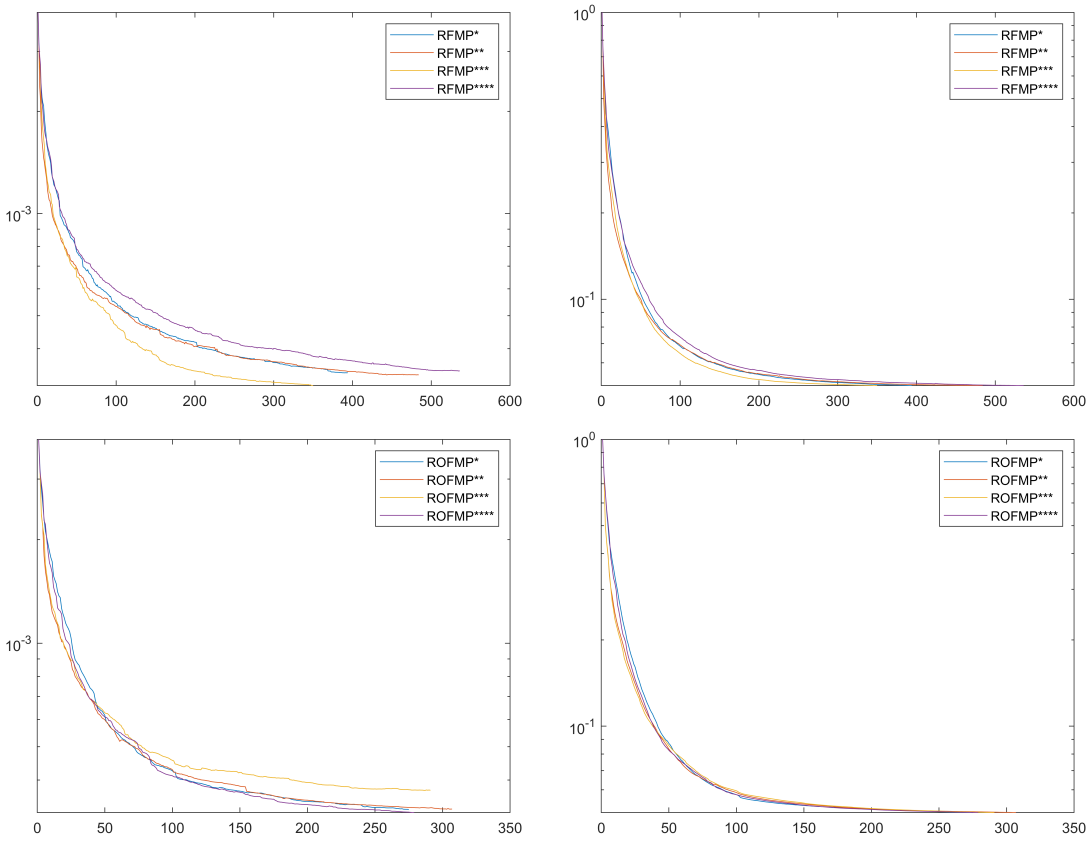


Figure 10.5.: The relative RMSE (left column) and relative data error (right column) of the RFMP (upper row) and the ROFMP (lower row) algorithm. IPMP* uses the manually chosen, IPMP** the learnt, IPMP*** the non-stationary learnt and IPMP**** the learnt-without-Slepian-functions dictionary. The x-axis denotes the iterations and the y-axis the logarithm of the error values.

(left, lower row) and the learnt-without-Slepian-functions (right, lower row) dictionary (Figure 10.4a for the RFMP and Figure 10.4b for the ROFMP algorithm). The solution of this experiment is given in the second row and the first column of Figure 9.2. Further, we give the relative data errors and the relative RMSEs along the iterations in Figure 10.5. In the last two rows of Table 10.2, we list the different error values at termination as well as the size of the used dictionary, the needed number of iterations, the maximal spherical harmonic degree and the CPU-runtime for a better comparison. Note that the maximal spherical harmonic degree equals the maximal learnt spherical harmonic degree in the cases where a learnt dictionary is used.

Evaluation Similarly as in Section 10.2, at first, we notice that the IPMP algorithms using all four trial functions produce a good approximation also for the

monthly GRACE data. Again, the use of all four trial functions already improves the results with respect to the approximation errors and the number of iterations (compare with Section 10.1).
As we consider the data of May 2008, the solution clearly has a local anomaly in the Amazon basin. This is due to the fact that the wet season in this region just ended at the time of the measurements. We see that, for both IPMP algorithms using the manually chosen as well as the learnt dictionaries, the remaining errors are mostly to be found in this area as well. Furthermore, we also see again that the approximation obtained using any learnt dictionary is equally well as the approximation obtained using the manually chosen dictionary. This is underlined by the final error values given in Table 10.2. However, even though we present here the results with the lowest approximation errors we obtained when testing several regularization parameters, it appears as if the choice of non-stationary parameter for the LROFMP algorithm could still be improved.
Moreover, in Table 10.2, we see that the non-stationary regularization parameter as well as the Slepian functions do not influence the results in the same manner as in the experiments with the EGM2008 data (Section 10.2). As we mentioned before, the number of tests we compare here is too little to draw meaningful conclusions. However, the results suggest that an investigation of a significant number of tests would be interesting in the future.
In contrast to Section 10.2, we see here that learning and applying a learnt dictionary has generally a shorter runtime than using the manually chosen dictionary and the saving of time is significant in some cases (up to 80% less runtime). Moreover, we notice that we have again a truly learnt maximal spherical harmonic degree and this degree is usually higher than in the manually chosen dictionary. Further, the number of performed iterations is very similar if we choose the manually chosen dictionary or a learnt dictionary. However, a learnt dictionary is of only about 0.5% of the size of the manually chosen dictionary. Recall that also here the experiments with the manually chosen dictionary have a much higher storage demand than learning a much smaller dictionary.
Hence, the outcome of this experiment underlines the conclusions drawn in Section 10.2: we are able to automatize the dictionary selection at lower computational costs.

		size of dictionary	completed iterations	maximal degree	relative RMSE	relative data error	CPU-runtime in h
EGM2008	RFMP*	95152	957	25	0.000466	0.049998	514.03
	RFMP**	≤ 637	662	46	0.000471	0.049999	507.22
	RFMP***	≤ 670	670	34	0.000447	0.050000	533.87
	RFMP****	≤ 684	734	41	0.000484	0.049999	129.57
EGM2008	ROFMP*	95152	766	25	0.000463	0.049999	561.76
	ROFMP**	≤ 550	577	38	0.000467	0.049998	665.76
	ROFMP***	≤ 686	701	35	0.000452	0.049999	840.06
	ROFMP****	≤ 600	621	46	0.000468	0.049998	386.08
GRACE	RFMP*	95152	393	25	0.000340	0.049997	522.09
	RFMP**	≤ 384	483	32	0.000335	0.049999	341.16
	RFMP***	≤ 352	349	28	0.000311	0.049994	284.85
	RFMP****	≤ 479	535	39	0.000344	0.049994	90.53
GRACE	ROFMP*	95152	274	25	0.000328	0.049989	528.67
	ROFMP**	≤ 303	306	34	0.000330	0.049998	372.31
	ROFMP***	≤ 292	290	33	0.000374	0.049998	358.64
	ROFMP****	≤ 278	278	26	0.000322	0.049996	182.19

Table 10.2.: Overview of the results at termination of the experiment described in Sections 10.2 and 10.3. The IPMP* algorithm uses the manually chosen dictionary, the IPMP** the learnt dictionary, the IPMP*** the non-stationary learnt dictionary and the IPMP**** the learnt-without-Slepian-functions dictionary. All learnt dictionaries are iteratively applied. The maximal degree is the maximal degree of a spherical harmonic included in the used dictionary.

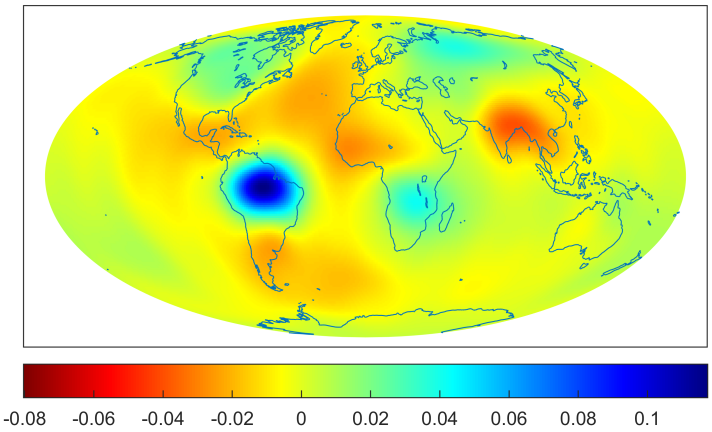


Figure 10.6.: Deviation from the mean field of the modified GRACE data in May 2009. The scale is adapted for better comparability with Figure 9.2. All values in m^2/s^2.

10.4. Learning a GRACE dictionary

In this experiment, we aim to learn a dictionary for the GRACE satellite mission from the LRFMP algorithm.

Particular setting We propose to learn a dictionary for GRACE from the LRFMP algorithm in the following way. We run the algorithm with the data of one year of GRACE measurements. We choose the data from 2008 for this (confer Figure 9.2). In particular, we run the LRFMP algorithm with all 12 monthly data sets separately. In this way, we learn one dictionary for each month of the year. The union of these (monthly) learnt dictionaries is then said to be our GRACE (year-) dictionary. We test the GRACE dictionary by applying it in the RFMP algorithm using test data from a different year. Note that here, naturally, we do not apply the learnt dictionary iteratively. As test data, we choose May 2009 (confer Figure 10.6). We compare the results using the GRACE dictionary with the outcome of the RFMP algorithm using the manually chosen dictionary to approximate the same data from 2009. In order to keep the experiment practical, we choose the same regularization parameter as in Section 10.3. That is, the regularization parameters in all runs are chosen as $10^{-4}\|y\|_{\mathbb{R}^\ell}$.

Results The results of this experiment are shown in Figure 10.7. We give the absolute approximation errors from the RFMP algorithm using the manually chosen dictionary (left) and the learnt GRACE dictionary (right). The solution of this experiment is given in Figure 10.6. Further, Figure 9.2 shows the data from which the learnt dictionary is obtained. The behaviour of the errors along the iterations is similar as in Section 10.3 and is, thus, omitted here. Using the manually

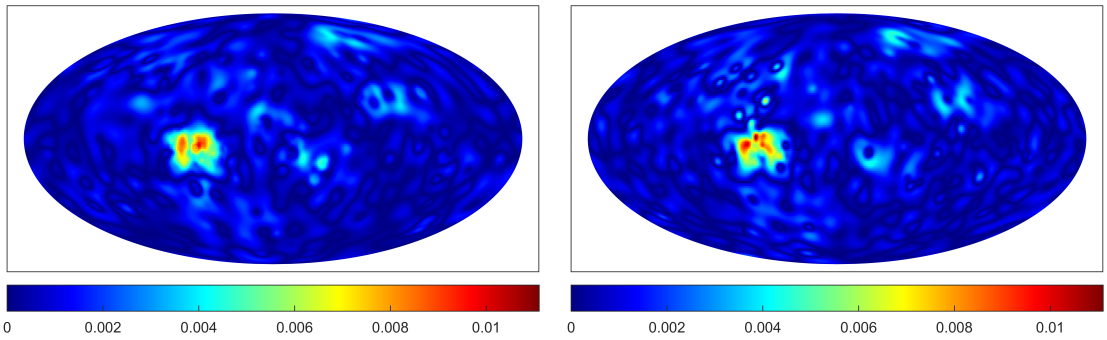


Figure 10.7.: Results of the experiment described in Section 10.4. Absolute approximation error obtained by the RFMP algorithm using the manually chosen dictionary (left) and using the learnt GRACE dictionary (right). The scale is adapted to improve the comparability. All values in m^2/s^2.

chosen dictionary of 95152 trial functions, the algorithm terminates after 399 iterations with a relative data error of 0.049999 and a relative RMSE of 0.000335. The learnt GRACE dictionary consists of not more than 6701 elements and the maximal spherical harmonic degree is 50. Then the RFMP algorithm ends after 620 iterations with a relative data error of 0.049994 and a relative RMSE of 0.000346.

Evaluation As could be expected after Section 10.3, we see that the RFMP algorithm using the manually chosen dictionary also approximates the data from May 2009 well. Furthermore, the remaining errors are still to be found mainly in the Amazon basin. Note again that, in this region, the wet season is about to end around that time of the year. Thus, as we see in Figure 10.6, the most local structures in the data are situated there. However, the maximal learnt spherical harmonic degree of the learnt GRACE dictionary is again higher than the corresponding degree included in the manually chosen dictionary.
Moreover, we see that the proposed approach to learn a GRACE dictionary produces a similar approximation. This shows that, in principle, we can learn a GRACE dictionary from one year of measurements and use it for test data from a different year. In particular, we conclude that, if to a certain extent the training and test data are similar, then we can learn a dictionary from the training data and use it for (unseen) test data. Note that this also hints at that the learnt dictionary could indeed be continuously dependent on the data.
However, the final relative RMSE obtained when using the learnt dictionary is slightly higher than with the manually chosen dictionary. Furthermore, as we see in Figure 10.7, the respective approximation contains some artefacts which might be caused by overfitting. We are aware that we present here merely a first strategy to learn a GRACE dictionary. Refinements (and, thus, improvements) of this dictionary could be done as follows in future research.

Here, we use the same regularization parameter for all months of the training data. Of course, it might be even better to determine a new (and possibly different) parameter for each month. Those parameters could also be determined with higher precision than was done here. However, we have to take into account that, in practice, there is surely a limit to the number of used training data and the accuracy of the determined regularization parameter.
Moreover, in the previously presented tests, we always applied the learnt dictionary iteratively. Naturally, this cannot be taken over to this experiment straightforwardly. Therefore, in the future research, instead of applying the union of the monthly learnt dictionaries at once, we could apply, in the N-th iteration, the union of only the first N learnt dictionary elements of each month.
At last, we might think of learning a dictionary from more than one year and / or only from similar months (here, this could be March to June). For this, it may be sensible to use the data from 2006 to 2010 (confer Section 9.2). This approach might serve even better for test data from the successor mission GRACE-FO. However, note that the there exists a time gap between the GRACE and the GRACE-FO mission. As we assume at least some similarities between the test and the trainings data, we have to bear in mind that, thus, the GRACE data may not be good training data if the test data comes from the GRACE-FO mission. For instance, the influence of the climate change we currently see will be captured by the test data. However, it remains open whether these effects are strongly enough visible in the GRACE training data.
Nonetheless, the union of the separately learnt 12 dictionaries overall has a size of less than 10% of the size of the manually chosen dictionary. Recall that each of the 12 runs of the LIPMP algorithm uses much less memory storage than the run with the manually chosen dictionary. Moreover, if we once have learnt a dictionary, for instance, for GRACE, its pure application will naturally have a shorter CPU-time due to the much smaller dictionary size. Both of these aspects might come in handy in future applications with more data values.

11. Further experiments with the LIPMP algorithms

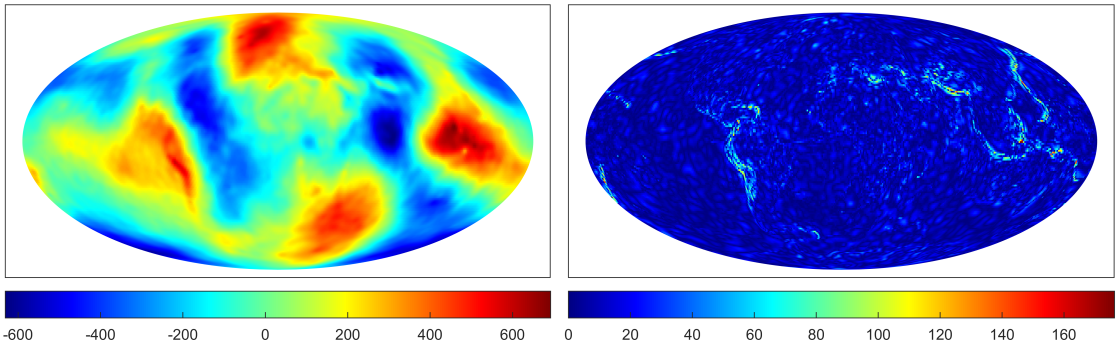
In this chapter, we present the results of the LIPMP algorithms in diverse experiments. That means, we consider whether it is an approximation algorithm on its own as well.

11.1. Approximation

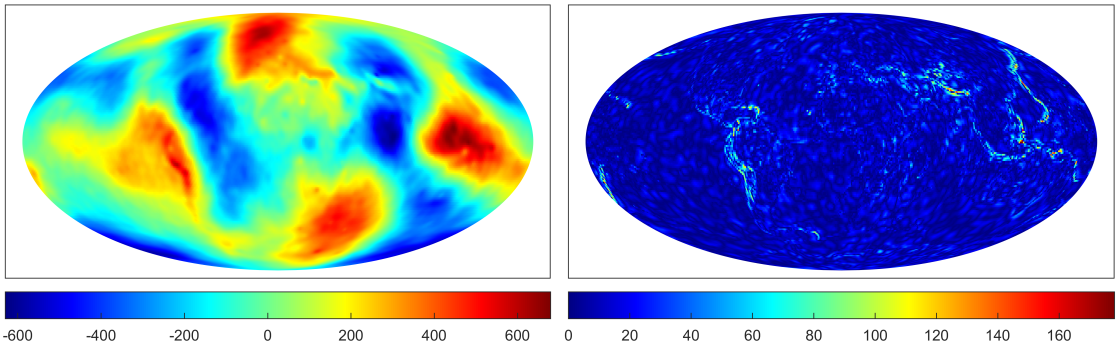
The aim of this experiment is to show that with a – in theory – well-posed problem, the LIPMP algorithm itself yields a good approximation. Further, remaining errors can be explained by the data. That means, the algorithm does not produce intrinsic errors.

Particular setting We approximate a solution from data given at the surface of the Earth, i. e. we have no satellite height. However, note that it was shown in Michel (2020), that the IPMP algorithms yield better results with a non-vanishing regularization parameter even if the problem here is theoretically well-posed. As the infinite dictionary contains trial functions that are very localized ($|x|$ up to $1 - 10^{-8}$ is feasible for the Abel–Poisson low and band pass filters), we present here also results for a positive regularization parameter. In particular, the regularization parameter is chosen as $10^{-9}\|y\|_{\mathbb{R}^\ell}$ for both LIPMP algorithms. Further, in contrast to the previous experiments, here the algorithms actually terminate after 1000 iterations as they do not reach the noise level up to then.

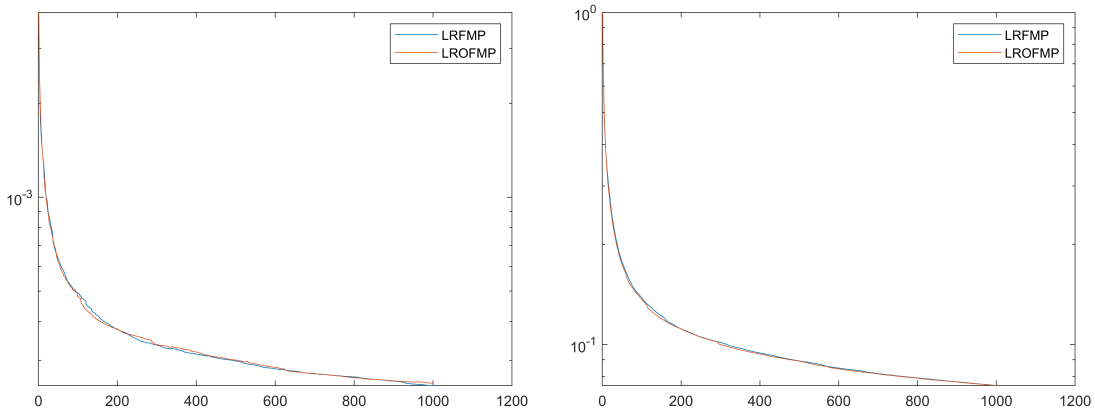
Results We present the results of this experiment in Figure 11.1. In particular, we show the approximations and the respective absolute approximation error obtained from the LIPMP algorithm (Figure 11.1a and Figure 11.1b, respectively). Note that the solution, i. e. the gravitational potential at the surface of the Earth, is given in Figure 2.2. Further, we give the relative data errors and the relative RMSEs along the iterations in Figure 11.1c. For the sake of completion, in Figure 11.2, we also give the absolute approximation errors obtained from the learning algorithms (in the setting with a stationary regularization parameter and all four trial function classes) in the experiments described in Sections 10.2 and 10.3.
For the surface data, the LIPMP algorithms end after 1000 iterations. The LRFMP algorithm terminates with a relative RMSE of 0.000249 and a relative data error of



(a) Approximation (left) and absolute approximation error (right) obtained by the LRFMP algorithm. All values in m^2/s^2.

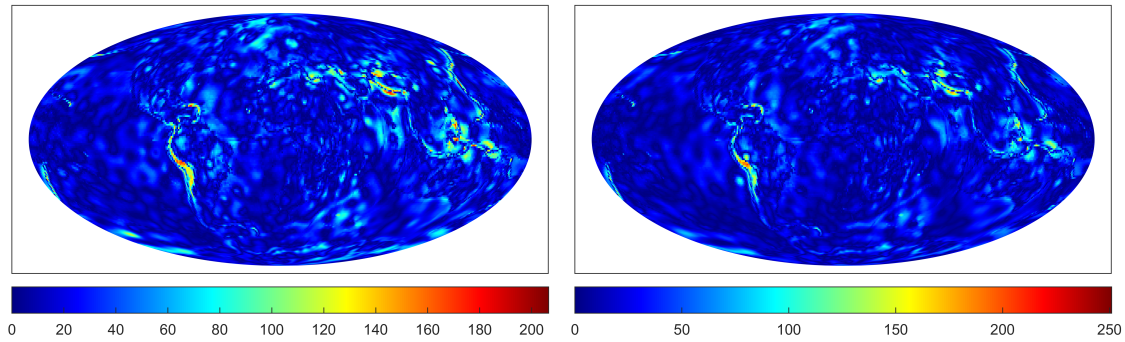


(b) Approximation (left) and absolute approximation error (right) obtained by the LROFMP algorithm. All values in m^2/s^2.

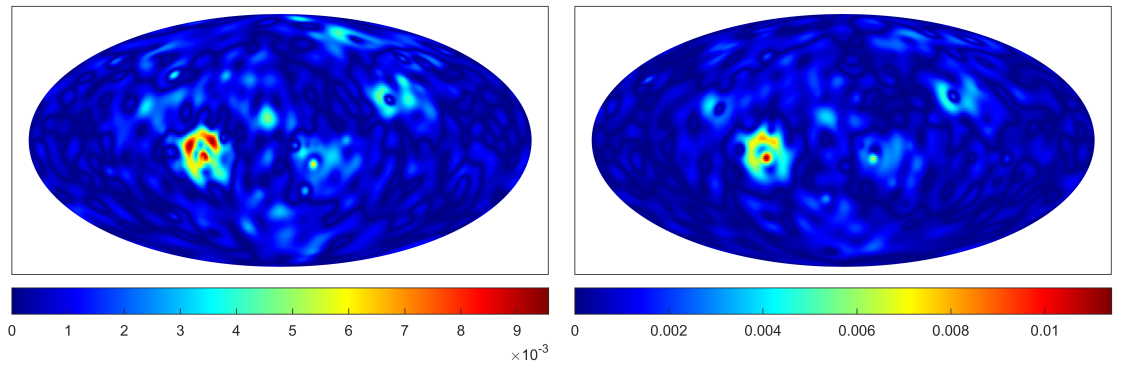


(c) The relative RMSEs (left) and the relative data errors (right) along the iterations. The x-axis denotes the iterations and the y-axis the logarithm of the error values.

Figure 11.1.: Results of the experiment described in Section 11.1.



(a) Absolute approximation error obtained by the LRFMP (left) and LROFMP (right) algorithm in Section 10.2.



(b) Absolute approximation error obtained by the LRFMP (left) and LROFMP (right) algorithm in Section 10.3.

Figure 11.2.: Results of the experiments described in Sections 10.2 and 10.3 obtained by the LIPMP algorithms (using a stationary regularization parameter and all four trial function classes). The scales are adapted for a better comparison with Figures 10.2 and 10.4. All values in m^2/s^2.

0.075293. The LROFMP algorithm finishes with a relative RMSE of 0.000253 and a relative data error of 0.075210. The learnt maximal spherical harmonic degree is 75 for the LRFMP and 83 for the LROFMP algorithm.

Evaluation First of all, note that, as we do not compute an upward continuation in this experiment, more local structures of the EGM2008 are contained in the data used by the LIPMP algorithms. Hence, we can expect to need a higher number of iterations and to remain with a higher relative data error, but also to obtain a better relative RMSE. These expectations are met by the algorithms. In fact, we see that we obtain a good approximation and the few remaining errors are in regions with higher local structures like the Andean region, the Himalayas and the Pacific Ring of Fire. Note that we use a positive regularization parameter, perturb the data and the algorithms terminate after a certain number of iterations.

Hence, it is reasonable that there are still some aspects that are not captured by the approximation. Nonetheless, we see that the LIPMP algorithm can be used as a standalone approximation algorithm as well. Note that we, again, have a truly learnt maximal spherical harmonic degree with both LIPMP algorithms. In comparison to the corresponding downward continuation experiment (confer Table 10.2), we obtain higher values here. On the one hand, this is caused by data given at the Earth's surface. On the other hand, this is surely influenced by the fact that our data is based on spherical harmonics. At last, note that these results were also obtained on a node with 12 CPUs and 48 GB RAM.
For the sake of completeness, we present the absolute approximation errors obtained by the LIPMP algorithms in the experiments described in Section 10.2 and Section 10.3 when using a stationary regularization parameter and all four trial function classes. That is, in Figure 11.2, we consider the absolute approximation errors obtained by the LIPMP algorithms for the EGM2008 and GRACE data of May 2008 upward continued to a satellite orbit. In particular, compare the left-hand side of Figure 11.2a with Figure 10.2a, the right-hand side of Figure 11.2a with Figure 10.2b, the left-hand side of Figure 11.2b with Figure 10.4a and the right-hand side of Figure 11.2b with Figure 10.4b. The LIPMP algorithms terminate when the relative data error fell below the noise level. The final relative RMSEs were 0.000471 (LRFMP algorithm) and 0.000465 (LROFMP algorithm) for the EGM2008 data and 0.000338 (LRFMP algorithm) and 0.000318 (LROFMP algorithm) for the GRACE data. Hence, the approximations obtained by the LIPMP algorithms are very similar to those of the IPMP algorithms which shows that the learning algorithms can also be used as an approximation algorithm for ill-posed inverse problems.

11.2. Downward continuation of irregularly distributed global data

We underline the results obtained by the LIPMP algorithms in Section 10.2 by exchanging the regular (Reuter) point grid with an irregular one.

Particular setting We remain with the overall setting as described in Chapter 9, but replace the Reuter grid by an irregular one that mimics terrestrial data, i. e. grid points that are denser over (approximately) the continents than over the oceans. To be more specific, we evaluate the EGM2008 at the grid shown in Figure 11.3 which contains 6968 grid points. The grid has already been used in some experiments in Michel and Telschow (2014). We gratefully acknowledge that Dr. Roger Telschow handed us this point grid. We choose the regularization parameter as $5 \cdot 10^{-9} \|y\|_{\mathbb{R}^\ell}$ for both LIPMP algorithms.

Results We give the results of this experiment in Figure 11.4 where we show the approximation and the absolute approximation error obtained from the LRFMP (Figure 11.4a) and the LROFMP (Figure 11.4b) algorithm. Note again that the respective solution was presented in Figure 2.2. The LRFMP algorithm terminates after 975 iterations with a relative RMSE of 0.000472 and a relative data error of 0.050000. The LROFMP algorithm ends after 983 iterations with a relative RMSE of 0.000521 and a relative data error of 0.049999. The learnt maximal spherical harmonic degree is 50 for both LIPMP algorithms.

Evaluation First of all, we notice that the approximation obtained from the LRFMP algorithm is similar to the results obtained from a regular data grid (confer Section 10.2) and most remaining errors are in areas with higher local structures. That means they are found where one would expect them to be as we are computing a perturbed downward continuation once again. With respect to the LROFMP algorithm, we see that the final approximation error is higher than for the LRFMP algorithm and for the experiments with regularly distributed data (confer Section 10.2). Nonetheless, the remaining errors are again in areas with higher local structure. Note that, if desired, a more thorough search for an optimal regularization parameter might be helpful.
Moreover, if we look closely at the absolute approximation errors in Figure 11.4, we see that, in comparison to the results presented in Figure 10.2, there are a few additional errors. In particular, this is seen for the results of the LROFMP algorithm. A comparison with Figure 11.3, however, shows that the approximation appears to be slightly worse in areas where we have less data. This is notably seen in the North Atlantic, in the Indian Ocean and at the north-east boundary of South America. Hence, this slight loss in accuracy is naturally caused by the irregular data grid. Note that, however, this shows that the algorithm is able distinguish regions with more or less data on its own and provide an approximation where this is mirrored in the accuracy of the different regions. In particular, this means, data gaps only have a local influence on the approximation of the LIPMP algorithms.

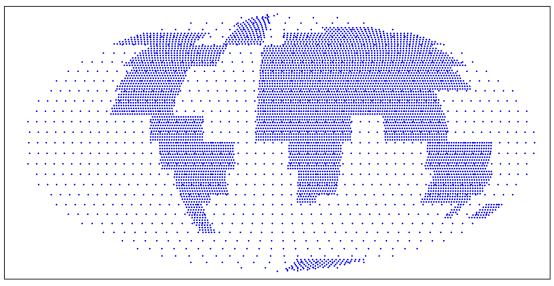
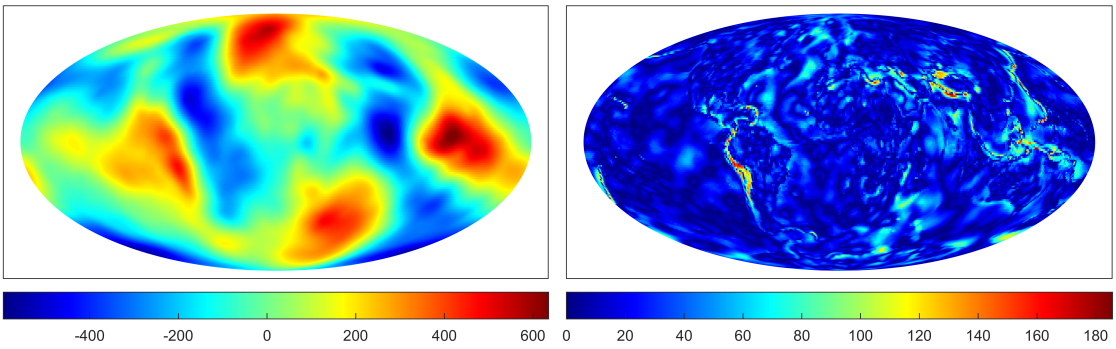
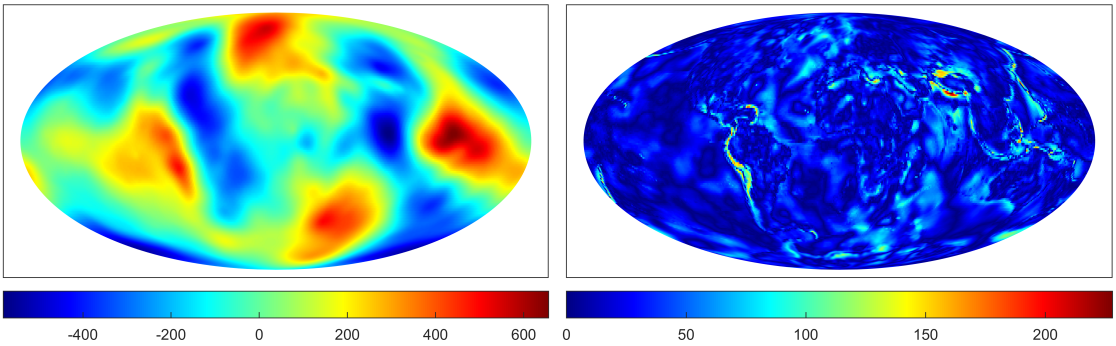


Figure 11.3.: Irregularly distributed point grid used in Section 11.2.



(a) Approximation (left) and absolute approximation error (right) obtained from the LRFMP algorithm.



(b) Approximation (left) and absolute approximation error (right) obtained from the LROFMP algorithm.

Figure 11.4.: Results of the experiments described in Section 11.2. All values in m^2/s^2.

The size of the learnt maximal spherical harmonic degree is much higher as is the number of iterations. In our experiments, we noticed that the algorithm chooses many global functions. We assume this is due to the fact that we compute our data as an expansion in spherical harmonics even though we are adding noise to the data (compare also with the results in Section 11.3). Additionally, we see that the LIPMP algorithm learns only 975 and 983 dictionary elements, respectively. Even though this sparsity is an advantage when applying the learnt dictionary, it prohibits a proper multiscale analysis here. Note that, in previous publications, where a multiscale analysis of the approximation of an IPMP algorithm was considered, the number of iterations was much higher (i. e. an iteration number of order 10^5).

11.3. Experiments with synthetic data

At last, we present experiments with synthetic data. This data is a combination of spherical harmonics and Abel–Poisson low pass filters. In this way, we sim-

ulate to have a signal that contains global trends as well as local anomalies. We investigate whether the LROFMP algorithms can distinguish these differences by choosing dictionary elements correspondingly. Due to the orthogonalization procedure, we expect that the LROFMP algorithm is better suited for this task than the non-orthogonal variant.
First, we explain the exact choice of trial functions which we combine for this use.

Chosen synthetic data As our synthetic data, we evaluate the following linear combination of spherical harmonics and Abel–Poisson low pass filters at the described Reuter and Driscoll-Healy point grids:

$$\begin{aligned} f = {} & Y_{9,5}(\cdot) + Y_{5,5}(\cdot) + Y_{2,0}(\cdot) \\ & + \widetilde{K}\left(x\left(0.5, \frac{3\pi}{2}, \frac{\pi}{4}\right), \cdot\right) + \widetilde{K}\left(x\left(0.75, 2\pi, -\frac{\pi}{4}\right), \cdot\right) + \widetilde{K}\left(x\left(0.9, \frac{\pi}{2}, \frac{\pi}{4}\right), \cdot\right) \end{aligned}$$

where the left-hand notation from (2.1) is used and $\widetilde{K}$ stands for the $\mathrm{L}^2(\Omega)$-normalized Abel–Poisson low pass filter. Then we have $y = \mathcal{T}\neg f$. In this way, we combine trial functions that are rather localized either in the spectral or in the spatial domain.

Particular setting Due to the composition of the synthetic data, we only consider spherical harmonics and Abel–Poisson low pass filters in the learning process and build the starting dictionary as

$$\begin{aligned} [N^{\mathrm{s}}]_{\mathrm{SH}} &= \left\{ Y_{n,j} \;\middle|\; n = 0, ..., 10;\ j = -n, ..., n \right\} \\ [B_K^{\mathrm{s}}]_{\mathrm{APK}} &= \left\{ \frac{K(x,\cdot)}{\|K(x,\cdot)\|_{\mathrm{L}^2(\Omega)}} \;\middle|\; |x| = 0.94,\ \frac{x}{|x|} \in X^{\mathrm{s}} \right\} \\ \mathcal{D}^{\mathrm{s}} &= [N^{\mathrm{s}}]_{\mathrm{SH}} + [B_K^{\mathrm{s}}]_{\mathrm{APK}}, \end{aligned}$$

where X^{s} is a Reuter grid with $G = 2$, i. e. 6 grid points. Then the Abel–Poisson low pass filters included in the synthetic data are not contained in the starting dictionary. We allow a maximum of 100 iterations because the data consists of only six trial functions. We consider perturbed and noise-free data. In the case of noise-free data, we also decrease the maximally allowed relative data error at termination to 10^{-3}. The regularization parameter is chosen as $10^{-11}\|y\|_{\mathbb{R}^\ell}$. In a second experiment, we additionally perturb the data (as described in Chapter 9) with 5% Gaussian noise and terminate the algorithm if the relative data error falls equal to or below this value. The regularization parameter is then chosen as $10^{-8}\|y\|_{\mathbb{R}^\ell}$.

synthetic data	coefficient	h	φ	t
	1	0.50	-1.570796	0.785398
	1	0.75	0	-0.785398
	1	0.90	1.570796	0.785398
unperturbed data				
	0.172076	0.474187	-1.418096	0.983475
	-0.122428	0.499957	-1.350861	0.989289
	3.940949	0.550327	-1.568047	0.743685
	-4.412483	0.578408	-1.564670	0.726832
	1.422986	0.607543	-1.560222	0.712725
	-0.151725	0.720342	-0.005947	-0.806070
	0.002125	0.742486	0.048077	-0.858082
	1.117564	0.743783	-0.001722	-0.788832
	0.033284	0.819294	0.024012	-0.758898
	1.129529	0.904131	1.571221	0.784891
	-0.138560	0.935506	1.571704	0.782854
perturbed data				
	-0.016580	0.475243	-1.421212	0.982664
	1.612762	0.515390	-1.568861	0.775215
	-0.817765	0.550241	-1.565803	0.748109
	0.210981	0.578198	-1.562345	0.730847
	-0.022001	0.683267	-0.365453	-0.698176
	-0.009561	0.683957	0.605419	-0.666855
	1.132485	0.741828	0.000147	-0.792845
	-0.114841	0.742278	0.047724	-0.858224
	0.306316	0.872652	1.567703	0.788411
	0.514873	0.903744	1.567918	0.784647
	0.005352	0.910476	0.080067	0.819269
	0.195857	0.935132	1.568059	0.782514
	-0.012904	0.963790	-0.546163	-0.847209
	0.016372	0.964061	-1.235509	0.772477
	0.016041	0.965315	-2.966837	0.981299
	-0.010882	0.965399	1.329488	0.951066
	-0.008931	0.965480	0.669439	-0.963845
	0.007888	0.967633	2.260697	-0.055792
	0.013679	0.968115	-1.978317	0.761322
	0.011965	0.968206	2.811542	0.774793
	0.011639	0.968313	-1.302818	0.918312

Table 11.1.: With respect to the experiment described in Section 11.3, the chosen Abel–Poisson low pass filters are presented. The filters with negligible coefficients are understated in blue.

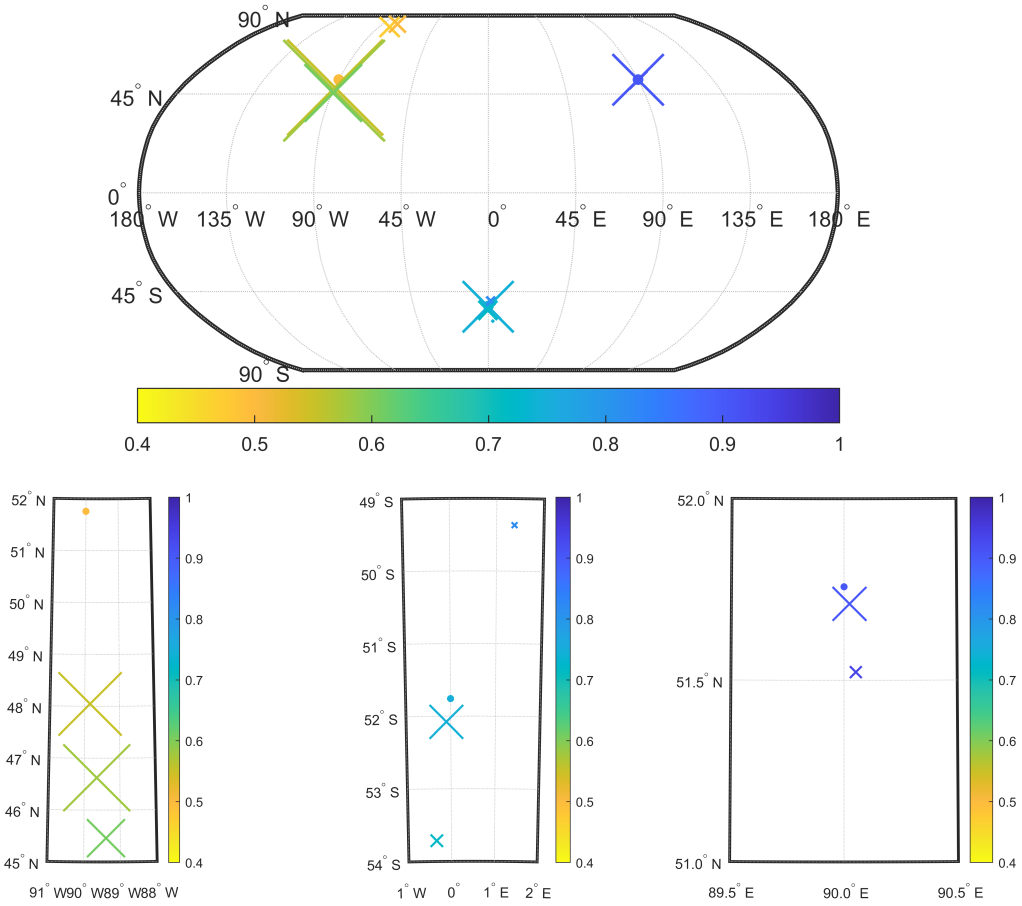


Figure 11.5.: Chosen Abel–Poisson low pass filters for unperturbed data from the experiment described in Section 11.3.

Results The experiment with noise-free data terminates after 14 iterations with a relative data error of 0.000879 and a relative RMSE of 0.000017. The experiment with perturbed data ends after 24 iterations with a relative data error of 0.049975 and a relative RMSE of 0.000076.
The algorithm chooses only the spherical harmonics $Y_{2,0}$, $Y_{5,5}$ and $Y_{9,5}$ with the coefficients 0.999615, 0.999842 and 0.999693 (unperturbed data) as well as 0.997702, 0.999490 and 1.000894 (perturbed data), respectively.
The learnt Abel–Poisson low pass filters are presented in Figures 11.5 and 11.6. The dots stand for the functions used in the synthetic data. The crosses symbolize the chosen filters. The colour of the dots and the crosses stands for the scale $|x|$. The size of the crosses represents the absolute value of the respective chosen coefficients α. The dots and the crosses are scaled for improved visibility. We view the chosen Abel–Poisson low pass filters globally as well as zoom in to regions of interest. Furthermore, we give the exact values of the chosen Abel–Poisson low pass filter in Table 11.1. We sort them by increasing scales and group them according to the scales of the filters used in the solution. Moreover, the wrongly

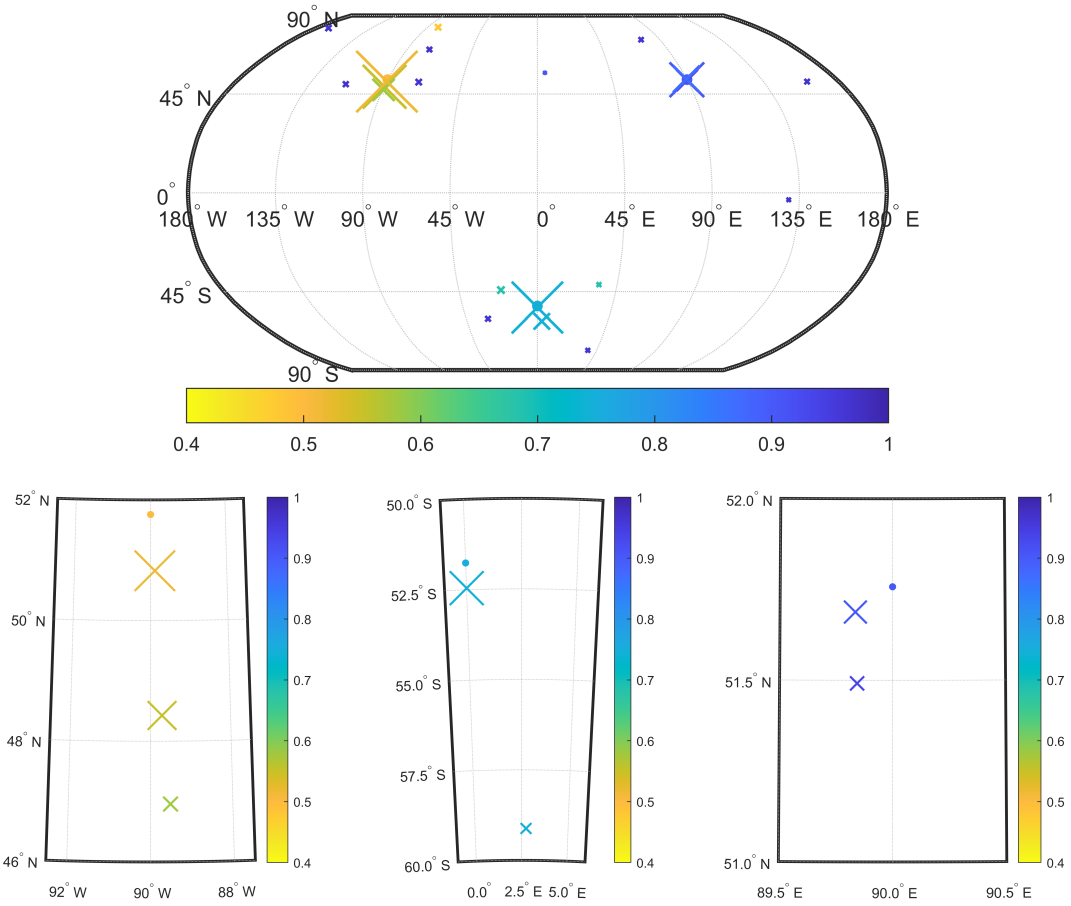


Figure 11.6.: Chosen Abel–Poisson low pass filters for perturbed data from the experiment described in Section 11.3.

chosen functions with lower coefficients are given in blue and, thus, understated in the table.

Evaluation First of all, from the relative error values at termination, we learn that the LROFMP algorithm also approximates a combination of spherical harmonics and Abel–Poisson low pass filters well.
Further, the algorithm is able to determine exactly the spherical harmonics included in the synthetic data for perturbed and unperturbed data. Additionally, the coefficients are suitably good determined for these functions as well.
In the global view on chosen Abel–Poisson low pass filters in Figure 11.5 and Figure 11.6, we see that, in both cases, these functions cluster around those used in the data. This is underlined by the local views on regions of interest. However, in particular with perturbed data, the chosen filters are not clustered as near to the solution as with noise-free data and we see more wrongly chosen functions. Nonetheless, this can be expected due to the present noise.

Note that we cannot expect that the choices are perfect in any of these cases due to several reasons: in contrast to the spherical harmonics, the Abel–Poisson kernels are not pairwise orthogonal. Hence, it is a higher challenge for the algorithm to distinguish between the different filters included in the data. Furthermore, we upward continue the data and use a positive regularization parameter.
Despite these aspects, we see that wrong choices of Abel–Poisson kernels generally have a small corresponding coefficient, i. e. have small crosses (see also Table 11.1). Due to the small coefficients of wrongly chosen trial functions, we assume that the algorithm corrects choices in the orthogonalization process. In particular, this hypothesis is supported as we experienced that they were not only chosen in later iterations.
At last, with respect to the coefficients given in Table 11.1, we see that the chosen coefficients of well chosen filters sum up approximately to 1.
Thus, obviously, the LROFMP algorithm is able to distinguish between global trends and local anomalies of a given signal.

Part IV.

Summary

12. Conclusion and Outlook

The gravitational potential is an important reference in geophysics as it provides information, e. g., of the climate change. Here, we are interested in the IPMP algorithms as methods to approximate this potential from satellite data. These approaches iteratively minimize the Tikhonov functional by choosing dictionary elements. Usually a set of global and local trial functions, such as spherical harmonics, Slepian functions as well as radial basis low and band pass filters, is utilized as a dictionary. However, there exist infinitely many trial functions which can be used as dictionary elements. Most of them can be characterized by certain continuous parameters. Therefore, we developed the LIPMP algorithms in this thesis to avoid selecting manually a dictionary. They follow the same structure as the IPMP algorithms. However, in each iteration, non-linear constrained optimization problems based on the continuous parameters are additionally solved.
The LIPMP algorithms can be seen as enhanced, standalone methods for solving ill-posed inverse problems. The chosen dictionary elements of the LIPMP algorithms define a learnt dictionary which can be used in the IPMP algorithms. Hence, we automatized the selection of dictionary elements and, with the novel methods, are able to work with an infinite dictionary as well.
As we built the LIPMP algorithms closely to the structure of the IPMP algorithms, we have seen that they inherit the latter one's convergence results as far as they currently exist. Moreover, we introduced a first approach to characterizing dictionaries. In particular, we saw that any complete dictionary is optimal by nature. A more practical property may be described by a sequence of well-working dictionaries that is able to reproduce the solution of the regularized normal equation. We showed that the LRFMP algorithm produces such a sequence for infinitely many iterations.
In our numerical experiments, we first note that the (L)IPMP algorithms are used with Slepian functions for the first time. We saw that the use of all four types of trial functions presented in this thesis for the downward continuation of satellite data is in particular interesting as it provides a higher sparsity in the solution. Hence, the experiments underline that all of these trial function classes have certain advantages in the (L)IPMP algorithms.
Moreover, we saw that the learning methods are widely applicable. We obtained suitably good approximations and the remaining errors were only in areas where more local structure is situated. As we mostly computed a downward continuation and, in any case, had to set certain termination criteria, this can be expected. From the experiments regarding the LIPMP algorithms, we see that, on its own, the novel learning variants are indeed approximation algorithms for well- and

ill-posed inverse problems. They can be used for regularly as well as irregularly distributed point grids. Furthermore, if a given signal consists of global trends and local anomalies, the LROFMP algorithm is able to distinguish between those parts. In the experiments where we compared a manually chosen and a learnt dictionary, we first of all saw that a dictionary learnt by the LRFMP algorithm from training data can principally be applied to unseen test data – at least if both data sets are to a certain extent similar. Furthermore, when we used all four trial functions presented in this thesis, we usually obtained a similarly good approximation no matter if we used a manually chosen dictionary or the learnt dictionary. When we used less trial function classes, we saw that the learnt dictionary may supersede the manually chosen one.
However, in general, the LIPMP algorithm needs less storage. Moreover, the runtime is also much smaller in most of our experiments. In particular, it is currently much smaller if no Slepian functions are used. Furthermore, the learnt dictionary usually contains essentially less trial functions than the manually chosen one used here. In particular, a learnt dictionary from and for a certain input data is usually of less than 1% of the size of our manually chosen dictionary. Even the GRACE dictionary we computed from a year of measurements is less than 10% of this size.
Taking all the aspects of this thesis into consideration, we suggest that, if CPU-runtime, storage demand and the implementation of many diverse suitable trial functions is not a challenging factor for a certain problem, then the non-learning variants can be of good use and are easier to implement. If, however, any of these aspects are problematic, we suggest to take the higher effort and implement the LIPMP algorithms. In particular, we favour the LRFMP algorithm because it yields similarly good results than the orthogonal variant while its implementation is easier and it needs less runtime.
However, there remain some open questions. First of all, there are still unsolved tasks for the IPMP algorithms which by construction are now also open for the novel methods. In particular, this is the choice of a perfect regularization parameter and further convergence results for the orthogonal variants of the approaches. The importance of an optimal regularization is well-known for inverse problems. However, there usually must be a trade-off between the optimality of the parameter and the practicability of its determination.
With respect to the learning technique itself, as could be expected, the accuracy of the optimization solver has a huge influence on the results. As we pointed out, in general, there exists a huge amount of optimization algorithms. Thus, if other than the here used methods for optimization are appealing to us in the future, it might be very interesting to see whether our current results can be improved by them. Furthermore, it would be desirable if the efficiency of the LIPMP algorithms with respect to the Slepian optimization problem could be improved.
Unfortunately, we have not been able to develop a theory for learning a dictionary in the limited time of the project. We managed to relate a dictionary and the Tikhonov functional and, thus, define what we understand as an optimal and

a sequence of a well-working dictionary. Next, it would be desirable to prove a relation between the limit of the sequence of learnt and well-working dictionary and an optimal one. However, this relation may depend on the operator and the data because the learnt dictionary does. Additionally, for a fixed operator, it would be interesting to discuss the continuous dependence of the limit of the sequence of learnt dictionary on the data.
Furthermore, we are aware that, in order to raise more interest in the geodetic community, the number of data values needs to be massively increased. Moreover, we are interested in applying the matching pursuit in other applications as well. As we now have developed a strategy to avoid choosing a dictionary manually which is also faster and has a much lower storage demand, we are very interested in approaching these challenges. In particular, both of these last two aspects will be addressed in a follow-up Postdoc-project supported by the German Research Foundation (DFG).

Part V.

Technical Appendix

A. Computational aspects

In the first part of this (technical) appendix, we summarize some aspects for practical purposes. These aspects contain examples for point grids, the Clenshaw algorithm as well as fully normalized spherical harmonics.

A.1. Point grids

Throughout this thesis, we used diverse point grids. Most importantly, we assume that the data y for the (L)IPMPs are given on such a grid. Further, for the computation of the approximation error, we need a second grid. Thus, we summarize some examples which we use in practice at this point.
We begin with the Driscoll-Healy grid (see e. g. Driscoll and Healy, 1994; Michel, 2013) which we use for evaluating the solution of, for instance, the EGM2008 or GRACE data and our approximation.

Definition A.1.1. *For $G_\varphi,\ G_\theta \in \mathbb{N}_0$, the Driscoll-Healy grid*

$$\left\{\eta^{(i,j)}\left(\varphi_i,\theta_j\right) \in \Omega\right\}_{i=1,...,G_\varphi,\ j=1,...,G_\theta}$$

is defined by

$$\begin{aligned} \varphi_i &:= \frac{2\pi}{G_\varphi} i, && i = 1,...,G_\varphi, \\ \theta_j &:= \frac{\pi}{G_\theta} j, && j = 1,...,G_\theta. \end{aligned}$$

The grid points are then obtained via (2.1).

Thus, the Driscoll-Healy grid is a mesh of a partition of the latitude and the longitude. For $G_\varphi = 360$ and $G_\theta = 180$, we obtain one grid point for each combination of longitudinal and latitudinal degrees.
An example is given in Figure A.1 on the left-hand side. Obviously, the point grid is not very well distributed as the grid points around the poles are much closer to each other than at the equator. A point grid with a better equidistribution is the so-called Reuter grid (see e. g. Reuter, 1982; Freeden et al., 1998; Michel, 2013) which is shown on the right-hand side of Figure A.1. Note that both point grids have nearly the same number of grid points in the examples in Figure A.1. The Reuter grid is defined as follows.

Definition A.1.2. *For a parameter $G \in \mathbb{N}$, we set*

$$\begin{aligned}
\theta_i &:= \frac{\pi}{G} i, && i = 0, ..., G, \\
\gamma_0 &:= 1, \\
\gamma_i &:= \left\lfloor 2\pi \left(\arccos\left(\frac{\cos\left(\frac{\pi}{G}\right) - \cos^2(\theta_i)}{\sin^2(\theta_i)} \right) \right)^{-1} \right\rfloor, && i = 1, ..., G-1, \\
\gamma_N &:= 1, \\
\varphi_{0,1} &:= 0, \\
\varphi_{i,j} &:= \left(j - \frac{1}{2} \right) \frac{2\pi}{\gamma_i}, && i = 1, .., G-1,\ j = 1, ..., \gamma_i, \\
\varphi_{N,1} &:= 0
\end{aligned}$$

with the Gaussian bracket $\lfloor \cdot \rfloor$. Then the Reuter grid is given by

$$\left\{ \eta^{(i,j)} \left(\varphi_{i,j}, \theta_i \right) \in \Omega \right\}_{i=1,...,G,\ j=1,...,\gamma_i}$$

with the use of (2.1).

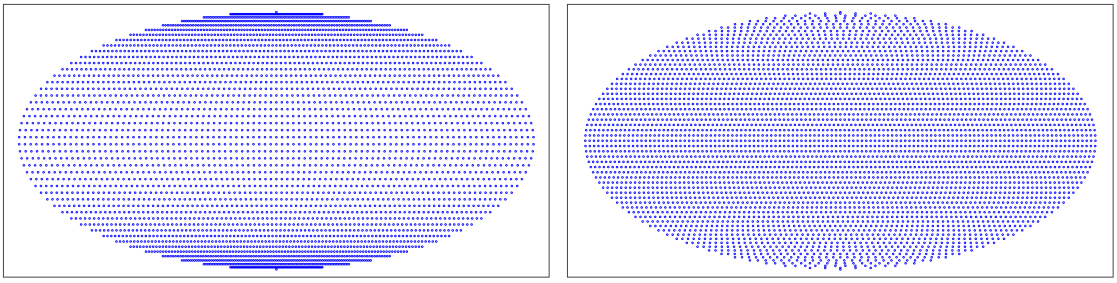


Figure A.1.: Left: Driscoll-Healy grid with $G_\varphi = 91$ and $G_\theta = 46$, i. e. 4186 grid points. Right: Reuter grid with $G = 60$, i. e. 4551 grid points.

A.2. Legendre polynomials and the Clenshaw algorithm

Due to the addition theorem for spherical harmonics, the $\mathcal{H}_2(\Omega)$-inner product of the Abel–Poisson low and band pass filters, see (3.23), (3.24) and (3.26), is built on Legendre polynomials. These polynomials $P_n \colon [-1, 1] \to \mathbb{R}$, $n \in \mathbb{N}_0$, fulfil the

three-term recursion

$$\begin{aligned} P_0(\tau) &= 1, \\ P_1(\tau) &= \tau, \\ P_n(\tau) &= \frac{2n-1}{n}\tau P_{n-1}(\tau) - \frac{n-1}{n}P_{n-2}(\tau), \end{aligned} \tag{A.1}$$

(see e. g. Abramowitz and Stegun, 1972; Freeden et al., 1998). This can be used straightforwardly for implementing the polynomials. The first derivative of the Legendre polynomials with respect to $\tau \in [-1,1]$ is immediately obtained by

$$\begin{aligned} P_0'(\tau) &= 0, \\ P_n'(\tau) &= \begin{cases} \frac{n\tau P_n(\tau) - nP_{n-1}(\tau)}{\tau^2-1}, & \tau \neq \pm 1,\ n \in \mathbb{N} \\ \frac{n(n+1)}{2}, & \tau = 1,\ n \in \mathbb{N},\ n \geq 1 \\ (-1)^{n+1}\frac{n(n+1)}{2}, & \tau = -1,\ n \in \mathbb{N},\ n \geq 1 \end{cases} \end{aligned} \tag{A.2}$$

which likewise can be easily implemented, see, for instance, Abramowitz and Stegun (1972) and Fengler (2005, Technical Appendix). Our aim is to compute the $\mathcal{H}_2(\Omega)$-inner product for the Abel–Poisson low and band pass filters for the (L)IPMPs from (3.23), (3.24) and (3.26). In practice, we can naturally only compute an approximative truncated series for it. Due to the three-term recursion of the Legendre polynomials (A.1), this sum can be efficiently computed with the Clenshaw algorithm. We first state the general algorithm (see e. g. Deuflhard, 1976; Michel, 2013) and then explain how it is applied to our cases.

Theorem A.2.1. ***(Clenshaw algorithm)*** *For functions $X_n \colon \mathbb{R} \to \mathbb{R}$, $n = 0, ..., N$ with $N \in \mathbb{N}_0$, that fulfil a three-term recursion, i. e.*

$$X_n\left(i\left(\tau,\tau'\right)\right) - a_n\left(i\left(\tau,\tau'\right)\right) X_{n-1}\left(i\left(\tau,\tau'\right)\right) - b_n\left(i\left(\tau,\tau'\right)\right) X_{n-2}\left(i\left(\tau,\tau'\right)\right) = 0 \tag{A.3}$$

for $n = 2, ..., N$, with a real-valued function $i(\cdot,\cdot)$ and with known values $a_n(i(\tau,\tau'))$, $b_n(i(\tau,\tau'))$, $X_0(i(\tau,\tau')) \neq 0$ and $X_1(i(\tau,\tau'))$ for all $\tau,\ \tau' \in \mathbb{R}^d$, $d \in \mathbb{N}$, the sum

$$S_N\left(\tau,\tau'\right) \coloneqq \sum_{n=0}^{N} c_n\left(\tau,\tau'\right)\ X_n\left(i\left(\tau,\tau'\right)\right) \tag{A.4}$$

can be calculated by

$$\begin{aligned} u_{N+1}\left(i\left(\tau,\tau'\right)\right) &\coloneqq u_{N+2}\left(i\left(\tau,\tau'\right)\right) \coloneqq 0, \\ u_n\left(i\left(\tau,\tau'\right)\right) &\coloneqq a_{n+1}\left(i\left(\tau,\tau'\right)\right) u_{n+1}\left(i\left(\tau,\tau'\right)\right) + b_{n+2}\left(i\left(\tau,\tau'\right)\right) u_{n+2}\left(i\left(\tau,\tau'\right)\right) \\ &\quad + c_n\left(\tau,\tau'\right),\ n = N, ..., 1 \\ S_N\left(\tau,\tau'\right) &= \left(c_0\left(\tau,\tau'\right) + b_2\left(i\left(\tau,\tau'\right)\right) u_2\left(i\left(\tau,\tau'\right)\right)\right) X_0\left(i\left(\tau,\tau'\right)\right) \\ &\quad + u_1\left(i\left(\tau,\tau'\right)\right) X_1\left(i\left(\tau,\tau'\right)\right). \end{aligned}$$

Data: $N \in \mathbb{N}_0,\ \tau, c_n(\tau,\tau'), n = 1,...,N$
Result: sum S_N
initialization: $u_2 = u_1 = u = 0$;
for $(n = N; n > 0; n\text{- -})$ **do**
  $u_2 = u_1;$
  $u_1 = u;$
  $u \ = \frac{2n+1}{n+1} i(\tau,\tau')\, u_1 - \frac{n+1}{n+2} u_2 + c_n(\tau,\tau');$
end
return $S_N = c_0(\tau,\tau') - \frac{1}{2}u_1 + ui(\tau,\tau');$

Algorithm 8: Code for the Clenshaw algorithm with respect to Abel–Poisson low and band pass filters and with the use of (A.5) for the computation of truncated $\mathcal{H}_2(\Omega)$-inner products (3.23), (3.24) and (3.26).

We use the Clenshaw algorithm for the truncated series of Legendre polynomials given in (3.23), (3.24) and (3.26). Thus, with the use of (A.1), we have for $\tau,\ \tau' \in \mathbb{B}$ and $n \in \mathbb{N}$ with $n \geq 2$

$$i(\tau,\tau') = \frac{\tau}{|\tau|} \cdot \frac{\tau'}{|\tau'|}, \qquad a_n(\tau,\tau') := \frac{2n-1}{n} i(\tau,\tau'), \qquad b_n(\tau) := -\frac{n-1}{n}$$
$$P_0\left(i(\tau,\tau')\right) = 1 \neq 0, \qquad P_1\left(i(\tau,\tau')\right) = i(\tau,\tau')$$

in (A.3). Further, for $c_n(\tau,\tau'),\ n \in \mathbb{N}_0$, from (A.4), we have

$$c_n(\tau,\tau') := \begin{cases} A_n^2 (|\tau||\tau'|)^n \frac{2n+1}{4\pi}, & \text{confer (3.23),} \\ A_n^2 \left(|\tau|^n - |\tau|^{2n}\right) |\tau'|^n \frac{2n+1}{4\pi}, & \text{confer (3.24),} \\ A_n^2 \left(|\tau|^n - |\tau|^{2n}\right)\left(|\tau'|^n - |\tau|^{2n}\right) \frac{2n+1}{4\pi}, & \text{confer (3.26).} \end{cases} \tag{A.5}$$

All in all, this yields an implementation of the Clenshaw algorithm for our needs as given in Algorithm 8.

A.3. Associated Legendre functions and fully normalized spherical harmonics

A.3.1. Evaluation for high-degree and order

Fully normalized spherical harmonics as given in (2.14) can be straightforwardly evaluated if the associated Legendre functions are available. The definition given in (2.13) is probably the shortest possible one for our needs. However, it does not come in handy for an implementation. Furthermore, we have to take into account

the normalization factor

$$p_{n,j} \coloneqq \sqrt{\frac{2n+1}{4\pi}\frac{(n-|j|)!}{(n+|j|)!}} \tag{A.6}$$

of the fully normalized spherical harmonics $Y_{n,j}$. For increasing degrees n and orders j, the stable computation of such terms is known to be problematic.
Therefore, we consider the recursive computation of "fully normalized associated Legendre functions" as was done by e. g. Fengler (2005, Technical Appendix), i. e. we look at the values of the functions

$$\widetilde{P}_{n,j}(\tau) \coloneqq p_{n,j} P_{n,j}(\tau).$$

We recall an algorithm for low degrees and orders in Algorithm 9. Similarly, the first derivative of a fully normalized associated Legendre function can be computed recursively, see Algorithm 10. For both algorithms, see also Fengler (2005, Technical Appendix).
These algorithms work well for computing spherical harmonics and Slepian functions with a low degree and order for dictionaries. However, for the evaluation, in particular, of the EGM2008, we need to evaluate the associated Legendre functions up to high degrees and orders. In this case, these algorithms can produce over- and underflow problems in practice. For this challenge, Fukushima (2012) presented a method that circumvents these problems. We assembled this method in Algorithm 11.

A.3.2. Particular terms to avoid singularities

For the LIPMP algorithms, we have to implement the gradient of inner products of the form $\langle C_n, d(z)\rangle_{\mathcal{H}_2(\Omega)}$ and, thus, the formulas (7.82) and (7.89), respectively. In particular, to maintain the well-definedness of the gradient, we have to implement (7.85) and (7.86), respectively. Hence, we cannot use Algorithm 9 and Algorithm 10 for the computation of associated Legendre functions and their derivatives due to the additional factors $\sqrt{1-t^2}$ and $(\sqrt{1-t^2})^{-1}$, respectively. However, if we take a closer look at them, we see that we can modify these algorithms for our purposes. The results are Algorithm 14 and Algorithm 15. To the knowledge of the author, these algorithms are novel. Therefore, in contrast to the previous algorithms, we give a few explanations for them.
We first consider (7.85). In particular, we are concerned with the computation of

$$\widehat{P}_{n,j}(t) \coloneqq \frac{\widetilde{P}_{n,j}(t)}{\sqrt{1-t^2}} = \frac{p_{n,j} P_{n,j}(t)}{\sqrt{1-t^2}} \tag{A.7}$$

for $j > 0$. Note that, for $j = 0$, the term (7.85) is zero. The Algorithm 14 computes $\widehat{P}_{n,j}(t)$ with the normalization factor $p_{n,j}$ as given in (A.6).

Data: $N \in \mathbb{N}_0,\ \tau \in [-1,1]$
Result: $\widetilde{P}_{n,j}(\tau) = p_{n,j}P_{n,j}(\tau),\ n = 0,...,N,\ j = 0,...,n$
initialize $\widetilde{P}_{0,0}(\tau) = \sqrt{\frac{1}{4\pi}}$;
for *n=1,...,N* **do**
  $\widetilde{P}_{n,n}(\tau) = \sqrt{\frac{2n+1}{2n}}\sqrt{1-\tau^2}\widetilde{P}_{n-1,n-1}(\tau)$;
end
for *j=0,...,N-1* **do**
  $\widetilde{P}_{j+1,j}(\tau) = \sqrt{2j+3}\tau\widetilde{P}_{j,j}(\tau)$;
  for *n=j+2,...,N* **do**
    $\widetilde{P}_{n,j}(\tau) = \sqrt{\frac{(2n-1)(2n+1)}{(n-j)(n+j)}}\tau\widetilde{P}_{n-1,j}(\tau) - \sqrt{\frac{2n+1}{2n-3}}\sqrt{\frac{(n+j-1)(n-j-1)}{(n-j)(n+j)}}\widetilde{P}_{n-2,j}(\tau)$;
  end
end
return $\widetilde{P}_{n,j}(\tau),\ n = 0,...,N,\ j = 0,...,n$;

Algorithm 9: Computation of $\widetilde{P}_{n,j}(\tau)$.

Note that Algorithm 14 is very similar to Algorithm 9: the values for order 0 are neglected for any degree and the start value is set to the next function with equal degree and order which is

$$\widehat{P}_{1,1}(t) = \frac{p_{1,1}P_{1,1}(t)}{\sqrt{1-t^2}} = \sqrt{\frac{3}{8\pi}}\sqrt{1-t^2}\frac{1}{\sqrt{1-t^2}} = \sqrt{\frac{3}{8\pi}}$$

for $t \in (-1,1)$. After that we have the same formulas as in Algorithm 9 for the other function values $\widehat{P}_{n,j}(t),\ n = 2,...,N,\ j = 1,...,n$. This holds true because we have eliminated one square root in $\widehat{P}_{1,1}(t)$ and the recursive formulas carry this elimination to all other function values. All in all, the term

$$\frac{Y_{n,j}\left(\frac{x(r,\varphi,t)}{r}\right)}{\sqrt{1-t^2}},$$

as used in (7.82) for $t \in [-1,1]$, is obtained straightforwardly from the definition of the fully normalized spherical harmonics if $p_{n,j}P_{n,j}$ is substituted by $\widehat{P}_{n,j}$ from Algorithm 14.
Next, we consider the computation of

$$\overline{P}_{n,j}(t) := \sqrt{1-t^2}\widetilde{P}'_{n,j}(t) = \sqrt{1-t^2}p_{n,j}P'_{n,j}(t) \tag{A.8}$$

for $j \geq 0$ and $t \in (-1,1)$. Note that, in Section 7.2, we see that the term vanishes for $t = \pm 1$. We already considered the value $p_{n,j}P'_{n,j}$ in Algorithm 10. We can

Data: $N \in \mathbb{N}_0,\ \tau \in (-1,1)$
Result: $\widetilde{P}'_{n,j}(\tau),\ n = 0,...,N,\ j = 0,...,n$
compute
P'_n for $n = 1,...,N+1$ and
$\widetilde{P}_{n,j}(\tau)$ for $n = 0,...,N,\ j = 0,...,n$;
for *n=0,...,N* **do**
  $\widetilde{P}'_{n,0}(\tau) = \sqrt{\frac{2n+1}{4\pi}} P'_{n+1}(\tau)$;
  for *j=1,...,n-1* **do**
    $\widetilde{P}'_{n,j}(\tau) = -\frac{j\tau}{1-\tau^2}\widetilde{P}_{n,j}(\tau) + \sqrt{\frac{(n+j+1)(n-j)}{1-\tau^2}}\widetilde{P}_{n,j+1}(\tau)$;
  end
  $\widetilde{P}'_{n,n}(\tau) = -\frac{n\tau}{1-\tau^2}\widetilde{P}_{n,n}(\tau)$;
end
return $\widetilde{P}'_{n,j}(\tau),\ n = 0,...,N,\ j = 0,...,n$;

Algorithm 10: Computation of $\widetilde{P}'_{n,k}(\tau)$.

modify this algorithm to compute $\overline{P}'_{n,j}(t)$ for $n = 0,...,N,\ j = 0,...,n$ and for all $t \in (-1,1)$ and obtain Algorithm 15. We give a proof for Algorithm 15.

Proof. First, we consider the case $j = 0$. We have

$$\overline{P}_{n,0}(t) = \sqrt{1-t^2}\,p_{n,0}P'_{n,0}(t) = \sqrt{1-t^2}\,p_{n,0}P'_n(t) = \sqrt{\frac{2n+1}{4\pi}}\sqrt{1-t^2}P'_n(t).$$

Next, we consider the other cases, i. e. we have $j \neq 0$:

$$\begin{aligned}
\overline{P}_{n,j}(t) &= \sqrt{1-t^2}\,p_{n,j}P'_{n,j}(t) \\
&= \sqrt{1-t^2}\,p_{n,j}\frac{\mathrm{d}}{\mathrm{d}t}\left[\left(1-t^2\right)^{j/2}\frac{\mathrm{d}^j}{\mathrm{d}t^j}P_n(t)\right] \\
&= \sqrt{1-t^2}\,p_{n,j}\left[\frac{j}{2}\left(1-t^2\right)^{j/2-1}(-2t)\frac{\mathrm{d}^j}{\mathrm{d}t^j}P_n(t) + \left(1-t^2\right)^{j/2}\frac{\mathrm{d}^{j+1}}{\mathrm{d}t^{j+1}}P_n(t)\right] \\
&= \sqrt{1-t^2}\,p_{n,j}\left[-jt\left(1-t^2\right)^{j/2-1}\frac{\mathrm{d}^j}{\mathrm{d}t^j}P_n(t) + \left(1-t^2\right)^{j/2}\frac{\mathrm{d}^{j+1}}{\mathrm{d}t^{j+1}}P_n(t)\right] \\
&= p_{n,j}\left[-jt\left(1-t^2\right)^{(j-1)/2}\frac{\mathrm{d}^j}{\mathrm{d}t^j}P_n(t) + \left(1-t^2\right)^{(j+1)/2}\frac{\mathrm{d}^{j+1}}{\mathrm{d}t^{j+1}}P_n(t)\right] \\
&= -jtp_{n,j}\left(1-t^2\right)^{(j-1)/2}\frac{\mathrm{d}^j}{\mathrm{d}t^j}P_n(t) + p_{n,j}\left(1-t^2\right)^{(j+1)/2}\frac{\mathrm{d}^{j+1}}{\mathrm{d}t^{j+1}}P_n(t).
\end{aligned}$$

Note that, in the case $j = n$, the second term vanishes because of the $(n+1)$-st

Data: $N \in \mathbb{N}_0,\ \tau \in [-1,1],\ B = 2^{960}$
Result: $\tilde{P}_{n,j}(\tau),\ n = 0,...,N,\ j = 0,...,n$
initialize $x(0,0) = \sqrt{\frac{1}{4\pi}};\ ix(0,0) = 0;$
for *n=1,...,N* **do**
  $x(n,n) = \sqrt{\frac{2n+1}{2n}}\sqrt{1-t^2}x(n-1,n-1);\ ix(n,n) = ix(n-1,n-1);$
  normalize $x(n,n)$ and update $ix(n,n)$ (see Algorithm 12);
end
for *j=0,...,N-1* **do**
  $x(j+1,j) = \sqrt{2j+3}\tau x(j,j);\ ix(j+1,j) = ix(j,j);$
  normalize $x(j+1,j)$ and update $ix(j+1,j)$ (see Algorithm 12);
  for *n=j+2,...,N* **do**
    compute $x(n,j)$ and $ix(n,j)$ via Algorithm 13;
    normalize $x(n,j)$ and update $ix(n,j)$ (see Algorithm 12);
  end
end
compute $\widetilde{P}_{n,j}(\tau)$ via
if $ix(n,j) < 0$ **then**
  $\widetilde{P}_{n,j}(\tau) = x(n,j)B^{-1};$
end
if $ix(n,j) == 0$ **then**
  $\widetilde{P}_{n,j}(\tau) = x(n,j);$
end
if $ix(n,j) > 0$ **then**
  $\widetilde{P}_{n,j}(\tau) = x(n,j)B;$
end
return $\widetilde{P}_{n,j}(\tau),\ n = 0,...,N,\ j = 0,...,n;$

Algorithm 11: Computation of $\widetilde{P}_{n,j}(\tau)$ using Algorithm 12 and Algorithm 13.

Data: $x(n,j),\ ix(n,j),\ B = 2^{960}$
Result: normalized $x(n,j)$ and updated value $ix(n,j)$
if $\left(|x(n,j)| \geq \sqrt{B}\right)$ **then**
  $x(n,j) = x(n,j)B^{-1};\ ix(n,j) = ix(n,j) + 1;$
end
if $\left(|x(n,j)| < \sqrt{B^{-1}}\right)$ **then**
  $x(n,j) = x(n,j)B;\ ix(n,j) = ix(n,j) - 1;$
end
return $x(n,j)$ and $ix(n,j);$

Algorithm 12: Normalization of $x(n,j)$ and corresponding update of $ix(n,j)$.

Data: $x(n-1,j),\ ix(n-1,j),\ x(n-2,j),\ ix(n-2,j),\ B = 2^{960}$
Result: $x(n,j),\ ix(n,j)$
if $ix(n-1,j) \geq ix(n-2,j)$ **then**
  $ix(n,j) = ix(n-1,j)$;
  switch $(ix(n-2,j) - ix(n-1,j))$ **do**
    case *0* **do**
      $x(n,j) = \sqrt{\frac{(2n-1)(2n+1)}{(n-j)(n+j)}}\tau x(n-1,j) - \sqrt{\frac{2n+1}{2n-3}}\sqrt{\frac{(n+j-1)(n-j-1)}{(n-j)(n+j)}}x(n-2,j)$;
    end
    case *-1* **do**
      $x(n,j) = \sqrt{\frac{(2n-1)(2n+1)}{(n-j)(n+j)}}x(n-1,j) - \sqrt{\frac{2n+1}{2n-3}}\sqrt{\frac{(n+j-1)(n-j-1)}{(n-j)(n+j)}}x(n-2,j)B^{-1}$;
    end
    case ≤ -2 **do**
      $x(n,j) = \sqrt{\frac{(2n-1)(2n+1)}{(n-j)(n+j)}}\tau x(n-1,j)$;
    end
  end
end
if $ix(n-1,j) < ix(n-2,j)$ **then**
  $ix(n,j) = ix(n-2,j)$;
  switch $(ix(n-1,j) - ix(n-2,j))$ **do**
    case *0* **do**
      $x(n,j) = \sqrt{\frac{(2n-1)(2n+1)}{(n-j)(n+j)}}\tau x(n-1,j) - \sqrt{\frac{2n+1}{2n-3}}\sqrt{\frac{(n+j-1)(n-j-1)}{(n-j)(n+j)}}x(n-2,j)$;
    end
    case *-1* **do**
      $x(n,j) = \sqrt{\frac{(2n-1)(2n+1)}{(n-j)(n+j)}}x(n-1,j)B^{-1} - \sqrt{\frac{2n+1}{2n-3}}\sqrt{\frac{(n+j-1)(n-j-1)}{(n-j)(n+j)}}x(n-2,j)$;
    end
    case ≤ -2 **do**
      $x(n,j) = -\sqrt{\frac{2n+1}{2n-3}}\sqrt{\frac{(n+j-1)(n-j-1)}{(n-j)(n+j)}}x(n-2,j)$;
    end
  end
end
return $x(n,j)$ and $ix(n,j)$;

Algorithm 13: Computing $x(n,j)$ and $ix(n,j)$ from $x(n-1,j)$, $ix(n-1,j)$, $x(n-2,j)$, $ix(n-2,j)$.

Data: $N \in \mathbb{N}_0,\ \tau \in [-1,1]$
Result: $\widehat{P}_{n,j}(\tau),\ n = 1,...,N,\ j = 1,...,n$
initialize $\widehat{P}_{1,1}(\tau) = \sqrt{\frac{3}{8\pi}}$;
for *n=2,...,N* **do**
  $\widehat{P}_{n,n}(\tau) = \sqrt{\frac{2n+1}{2n}}\sqrt{1-\tau^2}\widehat{P}_{n-1,n-1}(\tau)$;
end
for *j=1,...,N-1* **do**
  $\widehat{P}_{j+1,j}(\tau) = \sqrt{2j+3}\tau\widehat{P}_{j,j}(\tau)$;
  for *n=j+2,...,N* **do**
    $\widehat{P}_{n,j}(\tau) = \sqrt{\frac{(2n-1)(2n+1)}{(n-j)(n+j)}}\tau\widehat{P}_{n-1,j}(\tau)$
    $\quad -\sqrt{\frac{2n+1}{2n-3}}\sqrt{\frac{(n+j-1)(n-j-1)}{(n-j)(n+j)}}\widehat{P}_{n-2,j}(\tau)$;
  end
end
return $\widehat{P}_{n,j}(\tau),\ n = 0,...,N,\ j = 0,...,n$;

Algorithm 14: Computation of $\widehat{P}_{n,j}(\tau)$ from (A.7).

derivative of $P_n(t)$. For the first term, we have

$$\begin{aligned}
-jtp_{n,j}\left(1-t^2\right)^{(j-1)/2}\frac{\mathrm{d}^j}{\mathrm{d}t^j}P_n(t) &= -jt\left(1-t^2\right)^{-1/2}p_{n,j}\left(1-t^2\right)^{j/2}\frac{\mathrm{d}^j}{\mathrm{d}t^j}P_n(t)\\
&= -jt\left(1-t^2\right)^{-1/2}p_{n,j}P_{n,j}(t)\\
&= -jt\left(1-t^2\right)^{-1/2}\widetilde{P}_{n,j}(t)\\
&= -jt\widehat{P}_{n,j}(t).
\end{aligned}$$

For the second term, we use the equality

$$\sqrt{(n+j+1)(n-j)}p_{n,j+1} = p_{n,j}$$

and obtain

$$\begin{aligned}
p_{n,j}\left(1-t^2\right)^{(j+1)/2}\frac{\mathrm{d}^{j+1}}{\mathrm{d}t^{j+1}}P_n(t) &= \sqrt{(n+j+1)(n-j)}p_{n,j+1}P_{n,j+1}(t)\\
&= \sqrt{(n+j+1)(n-j)}\widetilde{P}_{n,j+1}(t).
\end{aligned}$$

All in all, this yields

$$\overline{P}_{n,j}(t) = -jt\widehat{P}_{n,j}(t) + \sqrt{(n+j+1)(n-j)}\widetilde{P}_{n,j+1}(t)$$

for $j \neq n$ and

$$\overline{P}_{n,n}(t) = -nt\widehat{P}_{n,n}(t). \qquad \square$$

Data: $N \in \mathbb{N}_0,\ \tau \in (-1,1)$
Result: $\overline{P}'_{n,j}(\tau),\ n = 0,...,N,\ j = 0,..,n$
compute
$P'_n(\tau), \widetilde{P}_{n,j}(\tau)$ for $n = 0,...,N,\ j = 0,...,n$ and
$\widehat{P}_{n,j}(\tau)$ for $n = 1,...,N,\ j = 1,...,n$;
for *n=0,...,N* **do**
  for *j=0,...,n* **do**
    if *j==0* **then**
      $\overline{P}'_{n,0}(\tau) = \sqrt{\frac{2n+1}{4\pi}}\sqrt{1-t^2}P'_n(\tau)$;
    end
    if $0 < j < n$ **then**
      $\overline{P}'_{n,j}(\tau) = -j\tau\widehat{P}_{n,j}(\tau) + \sqrt{(n+j+1)(n-j)}\widetilde{P}_{n,j+1}(\tau)$;
    end
    if *j==n* **then**
      $\overline{P}'_{n,n}(\tau) = -n\tau\widehat{P}_{n,n}(\tau)$;
    end
  end
end
return $\overline{P}_{n,j}(\tau),\ n = 0,...,N,\ j = 0,..,n$;

Algorithm 15: Computation of $\overline{P}_{n,j}(\tau)$ from (A.8).

A.4. Aspects of optimization

At last, we introduce some optimization algorithms which we use for our numerical experiments. Optimization problems occur in many different tasks – ranging from economics to science. Therefore, with respect to their specific details, algorithms are often tailored for a particular problem. However, their underlying principle may be a common one. Nonetheless, this leads to a huge variety of available software which can and should be exploited for solving optimization problems in practice. In this thesis, we mainly use the NLopt library (see Johnson, 2019) because it contains algorithms for global and local optimization tasks and enables an easy exchange (and, thus, comparison) of algorithms.
Note that, in our learning algorithm, we formulate optimization problems. However, the learning algorithm itself is in general independent of a particular optimization algorithm. Thus, in the sequel, we only summarize the main ideas of the different algorithmic approaches that are of practical relevance in this thesis.

A.4.1. Certain keywords

This section is based on Fletcher (1987); Geiger and Kanzow (2002); Glabonsky and Kelley (2001); Johnson (2019); Jones et al. (1993); Kraft (1988, 1994); Nocedal and Wright (2006); Rinnooy Kan and Timmer (1989); Runarsson and Yao (2000, 2005); Stein (2018); Zörnig (2014).
In an optimization problem, we either aim to minimize or maximize a result. However, every maximization can be transferred into a minimization and vice versa by multiplication with -1. In the sequel and for a mathematical definition, we consider (as usually) minimization problems. We model such problems as

$$f(x) \to \min! \qquad \text{subject to} \qquad g_i(x) \leq 0,\ i \in \mathcal{I}, \qquad g_j(x) = 0,\ j \in \mathcal{E} \tag{A.9}$$

with $f,\ g_i,\ g_j \colon \mathbb{R}^N \supset D \to \mathbb{R}$, $\mathcal{I},\ \mathcal{E} \subset \mathbb{N}$ and $\mathcal{I} \cap \mathcal{E} = \emptyset$. We call f the objective function with respect to the optimization variable $x \in \mathbb{R}^N$. The functions $g_i,\ i \in \mathcal{I}$, are inequality constraints and $g_i,\ i \in \mathcal{E}$, are equality constraints. The set of variables

$$G \coloneqq \left\{ x \in \mathbb{R}^N \;\middle|\; g_i(x) \leq 0,\ i \in \mathcal{I},\ g_i(x) = 0,\ i \in \mathcal{E} \right\}$$

is called the feasible set. For $x \in G$, the set

$$\mathcal{A}(x) \coloneqq \{ i \in \mathcal{I} \mid g_i(x) = 0 \}$$

is called the active set as it represents those inequality constraints which vanish (i. e. become an equality constraint) in a point x. Note that we allow $\mathcal{I} = \emptyset$ and $\mathcal{E} = \emptyset$ as well. If $\mathcal{I} = \mathcal{E} = \emptyset$, the problem is called unconstrained. Otherwise, it is called constrained. The (inequality and equality) constraints as well as the objective function may be linear or non-linear. Depending on f, we speak of a linear or a non-linear optimization problem, respectively. A particular case of linear inequality constraints are bound constraints, i. e. there exist $a,\ b \in \mathbb{R}^N$ such that it holds $a_i \leq x_i \leq b_i$ for all $i = 1, ..., N$.
Note that, for our purpose, we consider optimization problems for a continuous variable $x \in \mathbb{R}^N$. However, there is an extensive theory on discrete optimization ($x \in \mathbb{N}^N$) as well. Another distinction can be made between deterministic and stochastic optimization. In the latter case, the constraints and the objective functions are not fully known (only, e. g., an expectation value). In this thesis, however, we only deal with deterministic optimization problems. Nonetheless, also in this case, stochastic algorithms may be useful.
The solution of the minimization problem (A.9) shall be denoted by x^*. Usually, a solution x^* is categorized as a global solution, i. e. it holds

$$f(x^*) \leq f(x) \qquad \forall\, x \in G,$$

or as a local solution, i. e. there exists an $\varepsilon > 0$ such that we have

$$f(x^*) \leq f(x) \qquad \forall\, x \in \mathbb{B}_\varepsilon(x^*) \cap G.$$

In practice, a global solution is often harder to find than a local one. Further, many implementations of global algorithms are more concerned with finding a neighbourhood of the global solution than the solution itself. Hence, it is recommended to first try a global technique and use its solution as the starting point of a local method afterwards in order to obtain a better approximate of x^*.
With respect to global optimization, we first consider the particular case in which the feasible region G and the objective function f are both convex. In this case, we know that every local solution is also a global solution. Thus, there is no need for specific global optimization algorithms. However, if the problem is not convex and cannot be rearranged to a convex one, there are several other strategies currently available. Many of them, however, rely on local methods. The global aspect is then added either externally (when starting with a local approach and adding a global procedure), internally (when implementing the global aspect in defining formulae) or via a smooth shift of emphasis between global and local search for an optimum. This underlines again that the details of algorithms are often tailored for a specific task at hand.
Next, we summarize the theoretical background of non-linear optimization problems and common, well-known algorithmic techniques in order to give a broader view and, thus, enable a better understanding of the following subsections.
For unconstrained optimization problems, we recall fundamental results from analysis: for a local minimum, a vanishing gradient and a positive semi-definite Hessian is necessary; further, it is sufficient if we know that the gradient vanishes and the Hessian is positive definite. We obtain similar results if we have to take constraints into account as well. Then we consider the Lagrangian function

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i g_i(x)$$

and formulate a first and a second order necessary condition under certain constraint qualifications as follows: if the set of gradients of active constraints

$$\{\nabla g_i(x^*) \mid i \in \mathcal{A}(x^*)\}$$

is linearly independent, the first order necessary condition of a local minimum is given by the so-called Karush-Kuhn-Tucker conditions:

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*) &= 0, \\ g_i(x^*) &= 0,\ i \in \mathcal{E}, \\ g_i(x^*) &\leq 0,\ i \in \mathcal{I}, \\ \lambda_i^* &\geq 0,\ i \in \mathcal{I}, \\ \lambda_i^* g_i(x^*) &= 0,\ i \in \mathcal{I} \cup \mathcal{E}. \end{aligned}$$

Furthermore, the second order necessary condition for a local minimum is given (under the same assumptions) by

$$d^T \nabla_{xx} \mathcal{L}(x^*, \lambda^*) d \geq 0$$

for vectors d in

$$\left\{ d \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} d^T \nabla g_i(x^*) = 0 & \forall\, i \in \mathcal{E}, \\ d^T \nabla g_i(x^*) = 0 & \forall\, i \in \mathcal{A}(x^*) \subseteq \mathcal{I}, \lambda_i^* > 0, \\ d^T \nabla g_i(x^*) \geq 0 & \forall\, i \in \mathcal{A}(x^*) \subseteq \mathcal{I}, \lambda_i^* = 0 \end{array} \right\}, \tag{A.11}$$

i. e. the Hessian of the Lagrangian function is positive semi-definite in (x^*, λ^*) for certain vectors d. The set (A.11) is also called the critical cone. Similarly as in the unconstrained case, the second order sufficient condition is that the Hessian of the Lagrangian function is positive definite for all $d \neq 0$ in the critical cone (A.11).
Optimization algorithms try to tackle the problem of finding x^* with a derivative-free or gradient-based technique. In the former case, the methods usually compare the values of the objective function at different points. In the latter case, the approaches often determine points where the necessary conditions are fulfilled. In both cases, an algorithm usually produces a sequence of points $\{x^{(k)}\}_{k \in \mathbb{N}_0}$ with improving (de- or increasing) function values. The next iterate $x^{(k+1)} \in \mathbb{R}^N$ is determined by adding a step direction $p^{(k)} \in \mathbb{R}^N$ of a certain step size $\gamma_k \in \mathbb{R} \backslash \{0\}$ to $x^{(k)}$:

$$x^{(k+1)} = x^{(k)} + \gamma_k p^{(k)}. \tag{A.12}$$

Classical choices of $p^{(k)}$ in gradient-based methods for unconstrained optimization problems are

$$p_k = -\frac{\nabla_x f\left(x^{(k)}\right)}{\left\| \nabla_x f\left(x^{(k)}\right) \right\|_{\mathbb{R}^n}} \qquad \text{(steepest descent)},$$

$$p_k = -\left(\nabla_{xx} f\left(x^{(k)}\right)\right)^{-1} \nabla_x f\left(x^{(k)}\right) \qquad \text{(Newton method)}$$

or

$$p_k = -B_k^{-1} \nabla_x f\left(x^{(k)}\right) = -H_k \nabla_x f\left(x^{(k)}\right) \quad \text{(Quasi-Newton method)}.$$

Note that the second variant is only possible if the Hessian of f is available, positive definite and the next iterate $x^{(k+1)}$ yields a satisfactory improvement in the objective function. If this is not the case, the third option tries to mimic the direction obtained via the Hessian of f if B_k is chosen as an approximative matrix of $\nabla_{xx} f$. For instance, such an approximation can be obtained via (L-)BFGS updates. The choice of the step size γ_k is in general difficult because it must take into account whether we are looking for a global or local solution, how much improvement in the objective function shall be enforced and whether the new iterate may be infeasible for too large step sizes. However, in this summary, we do without a detailed description of methods for this task. Nonetheless, we bear in mind that this choice should be taken with care.

For constrained optimization problems, there are two main approaches. Either the original problem is quadratically approximated or it is modelled towards an unconstrained one.
In the former case, the original optimization problem is approximated by a quadratic one. A quadratic subproblem is obtained by applying the Newton method to the gradient of the Lagrangian function making use of the current iterate $x^{(k)}$. The subproblem produces a direction $p^{(k)}$ which is used for the next iterate. Either $x^{(k+1)}$ is suitable as the solution of the original problem as it fulfils the Karush-Kuhn-Tucker conditions (A.10). Or it can be used for the formulation of the next quadratic subproblem. Hence, an optimum is obtained via solving a sequence of quadratic approximative problems which gives the approach the name SQP (sequential quadratic programming). Note that, similar to the unconstrained case, also here the Hessian of the Lagrangian function needed in the Newton method can be substituted by an approximative, e. g. (L-)BFGS, matrix. An example of such a method is the SLSQP algorithm summarized in Appendix A.4.3.
For the latter case, either a multi-objective approach optimizes the objective function and the constraints simultaneously in a new vector-valued objective function. Or additional terms based on the constraints are added to the objective or Lagrangian function. The additional terms are either penalty terms (that penalize infeasible points, aka penalty methods) or barrier terms (that forbid infeasible points, aka barrier methods). Independent of whether penalty or barrier terms are used, they are usually defined as linear combinations of the constraints. The coefficients are called penalty parameters. In this way, again, an unconstrained optimization problem is obtained which can be solved with established methods. Mostly, a sequence of these new unconstrained optimization problems is solved in which the penalty parameters are decreased and, thus, a solution of the original problem is approximated. An example of such a method is the IPOPT algorithm summarized in Appendix A.4.4.
These common aspects shall serve as a basis for the next discussion of particular algorithms used in our numerical experiments.

A.4.2. The locally-biased DIRECT algorithm

The DIRECT (DIviding RECTangles) algorithm (Jones et al., 1993; Jones, 2001) as well as its locally-biased variant (Glabonsky, 1998; Glabonsky and Kelley, 2001) are global optimization algorithms. This section is based on these publications. Without loss of generality, the feasible region is dilated into the unit hypercube. The bound constraints then define its outer vertices. The inequality constraints are incorporated separately.
The algorithm implements a branch-and-bound approach. That means, the hypercube is iteratively divided into more and more hyperrectangles which are evaluated via bound values. Thus, the algorithm is a special form of a Lipschitzian optimization technique. A univariate Lipschitzian approach has the

advantages that it is a global method, has easy proofs of convergence, is deterministic, contains only a few parameters and has meaningful termination criteria. However, for classic variants, the Lipschitz constant needs to be known, the speed of convergence is relatively low and the computational complexity of higher dimensions is impractical. The DIRECT algorithm aims to overcome these challenges.
The main routine is dividing the hypercube into more and more hyperrectangles while sampling their centre points as iterates. As a technical matter, the vertex-to-centre diameter is measured in two different ways in the diverse variants of the DIRECT algorithm. Either the Euclidean distance is used or we have the diameter $d = 3^{-l}/2$ if 3^{-l} is the length of the longest edge of the hyperrectangle. Note that the latter is a kind of l^∞ measure.
The main question is how to divide which hyperrectangles. We first state how to divide the hyperrectangles of a certain iteration. For each dimension $i = 1, ..., n$, we store a counter t_i for how often we have split a hyperrectangle along this dimension. Let us consider the j-th hyperrectangle. We determine the set of long sides (i. e. the longest edges) of it. We trisect the side i with the lowest value t_i and increase t_i by 1. If several long sides attain this value of t_i, we choose the lowest dimension. The left- and right-hand centre points (i. e. $c^{(j)} \pm (d_j/3)e^{(i)}$ if we trisect along i and with d_j the length of the long side and $e^{(i)}$ the i-th standard basis vector) are then added to the set of centre points. At the end of the iteration, we determine the new value of $f_{\min}$ among the set of all centre points. If the termination criteria are fulfilled, we stop the iteration process and return the centre point related to $f_{\min}$ as the approximative solution. Possible termination criteria are the approximation error (if the global solution is known), the number of iterations or the number of function evaluations. Note that this algorithm is derivative-free and, thus, cannot check whether the necessary conditions are fulfilled. Thus, a natural termination criterion would also be if there is only small progress in the values of the objective function between successive iterates.
At last, we discuss which hyperrectangles should be divided. For this, the inequality constraints are combined with a penalty term for deviating from the value at the global minimum f^* in an auxiliary function

$$\widetilde{h}_j(f^*) = \max\left(f\left(c^{(j)}\right) - f^*, 0\right) + \sum_{i=1}^{m} b_i \max\left(g_i\left(c^{(j)}\right), 0\right) \geq 0$$

for positive weights b_i and the constraints g_i, $i = 1, ..., m$. In practice, f^* is not known in advance, but we may have a first estimate $f_{\min} - \varepsilon$ which we aim to improve. Hence, we consider $\widetilde{h}$ for all possible values f^* of the global minimum between $f_{\min} - \varepsilon$ and $-\infty$.
In the centre point $c^{(j)}$ of the j-th hyperrectangle/-cube, the auxiliary function is positive as long as $f(c^{(j)})$ is not the global optimum and decreases towards the global minimum. Obviously, in larger hyperrectangles, this decrease does not have to be as steep as in smaller ones. Hence, we consider the minimum rate of

change which is defined by

$$h_j(f^*) = \frac{\widetilde{h}_j(f^*)}{d_j}$$

where d_j denotes the diameter of the j-th hyperrectangle. For this diameter, we can use either one of the previously mentioned measures. As the Euclidean measure is usually larger than the l^∞ one, the latter one incorporates a bias to local search in the algorithm.
Then we plot the curves of $h_j(f^*)$ for all hyperrectangles in a diagram. We start at $(f_{\min} - \varepsilon, 0)$ and go upwards until we intersect the first curve. Then we follow this curve for decreasing possible values of f^* (i. e. to the left) until we intersect the next curve. We do so until we find a curve that is not intersected again. The hyperrectangles related to the curves we followed are then the hyperrectangles we divide in this iteration. For an illustration of this procedure, see Glabonsky and Kelley (2001).
For other optimization problems, there exist other variants of this algorithm. The classic DIRECT algorithm using the Euclidean measure of the diameter considers only bound constrained problems. It is guaranteed to be convergent to the global optimum if the objective function is continuous at least in a neighbourhood of the optimum. For the locally-biased variant for bound constrained problems, explicit convergence statements are not made in the mentioned literature.
The presented variant of the DIRECT algorithm is the for us most interesting one as it incorporates also inequality constraints. Here, we find in the literature again that convergence shall be guaranteed if the objective and constraint functions are continuous in a neighbourhood of the optimum. Note that the objective function and constraints in the LIPMP algorithm are obviously continuous on their domains. Hence, the DIRECT algorithm should converge also in our experiments.

A.4.3. The SLSQP algorithm

In the experiments in which we use the NLopt library, we choose to use the SLSQP algorithm (Kraft, 1988, 1994) for the local optimization problems. It is an SQP method, but it uses a least squares approach (hence, **s**equential **l**east **s**quares **q**uadratic **p**rogramming). This section is based on the mentioned publications as well as (Geiger and Kanzow, 2002).
In each iteration, it is checked whether the current iterate fulfils the KKT conditions of the original problem. If not, a quadratic subproblem is modelled by

$$\frac{1}{2} d^{\mathrm{T}} B^k d + \left(\nabla f\left(x^{(k)}\right)\right)^{\mathrm{T}} d \to \text{min!} \quad \text{subject to} \quad \left(\nabla g_i\left(x^{(k)}\right)\right)^{\mathrm{T}} d + g_i\left(x^{(k)}\right) = 0 \tag{A.13}$$

for $i = 1, ..., m$, $m \in \mathbb{N}_0$, $m \leq |\mathcal{I}| + |\mathcal{E}|$, $d \in \mathbb{R}^N$ and an approximate $B^k \approx \nabla^2_{xx}\mathcal{L}(x^{(k)}, \lambda_k)$. Note that, for the latter, one can fall back on a BFGS approximation or its limited memory variant. The next iterate (confer (A.12)) is then obtained by

$$x^{(k+1)} = x^{(k)} + \gamma_k d^{(k)},$$

where $d^{(k)} \approx d^*$ is the solution of the quadratic problem (A.13) and γ_k is chosen to be either 1 or as the minimizer of a one-dimensional penalty function dependent on $x^{(k)}$ and $d^{(k)}$ in order to guarantee global convergence. Hence, this model evaluates the original objective function and constraints as well as their gradients. We see that, in general, the SQP method follows the idea of a (Quasi-)Newton method for determining KKT points.
However, the subproblem (A.13) only contains equality constraints. For using the model (A.13) also for inequality-constrained optimization problems, a common approach is an active set strategy. The active set of the current iterate enables us to formulate the respective subproblem with only equality constraints (i. e. the equality constraints plus the active inequality constraints of the original problem). The non-active inequality constraints of the current iteration are neglected. Note that the current active set does not need to be computed from scratch for each subproblem (see e. g. Kraft (1988) as well as common literature on constrained optimization such as Geiger and Kanzow (2002) or Nocedal and Wright (2006)).
With the active set strategy, we can use (A.13) also for approximating arbitrary optimization problems. The solutions are obtained in the SLSQP algorithm as follows. It is known that a quadratic optimization problem has an equivalent formulation as a least squares problem when using an LDL^{T} factorization of the matrix B_k. Using this in the SLSQP algorithm, the least squares problem is then solved via certain reformulations. From the dual problem of the final formulation of the least squares problem (a non-negative least squares problem), we easily obtain the next iterate $x^{(k+1)}$ as well as its corresponding Lagrange multipliers.
As a particular SQP approach, the SLSQP algorithm is convergent under certain assumptions: strict complementarity, linear independence constraint qualifications and the second order sufficient condition. However, in practical applications, most SQP-like algorithms do not use the Hessian of the Lagrangian function but an approximation. Furthermore, to the knowledge of the author, the strict complementarity can only be checked during run-time. Hence, unfortunately, we have no proof that the algorithm should converge in our experiments. However, the SQP method is a very efficient approach with respect to function evaluation and time for arbitrary constrained non-linear optimization problems. Moreover, nowadays, it is viewed as a solid method.

A.4.4. The IPOPT algorithm

We started our development of a learning strategy for the IPMP algorithms using the IPOPT software (see e. g. Vigerske et al., 2016; Wächter, 2002; Wächter and Biegler, 2005a,b, 2006). Further, we used certain subroutines from HSL (2018). This section is based on these publications. IPOPT is an optimization software that implements an interior point line search filter algorithm.
Similar as the SLSQP algorithm, it also produces a sequence of iterates of the form (A.12) via a line search approach. That means, first a direction $p^{(k)}$ is determined. Afterwards the step length γ_k is computed. This is a common approach for unconstrained optimization problems. Possible constraints are incorporated by the "interior point" or barrier as well as the filter ansatz.
The optimization problem that is considered by the implemented software is given by

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to} \quad g_i^{(L)} \leq g_i(x) \leq g_i^{(U)},\ i = 1, ..., I \tag{A.14}$$

where $g_i^{(L)} = -\infty$ and $g_i^{(U)} = +\infty$ is possible. Note that, if g_i is the identity, this includes simple bound constraints. Furthermore, for $g_i^{(L)} = g_i^{(U)}$, this also takes equality constraints into consideration. This optimization task is then reformulated in the k-th iteration to a barrier approach. With

$$\varphi_{\mu_k}(x) := f(x) - \mu_k \sum_{i\in I_L} \ln\left(g_i(x) - g_i^{(L)}\right) - \mu_k \sum_{i\in I_U} \ln\left(g_i^{(U)} - g_i(x)\right)$$

we consider

$$\min_{x\in\mathbb{R}^n} \varphi_{\mu_k}(x) \quad \text{subject to} \quad c(x) := g_i(x) - g_i^{(L)} = 0,\ i \in I_{L,U}$$

using

$$\begin{aligned} I_L &= \left\{ i \in I \;\middle|\; g_i^{(L)} \neq -\infty \right\}, \\ I_U &= \left\{ i \in I \;\middle|\; g_i^{(U)} \neq +\infty \right\}, \\ I_{L,U} &= \left\{ i \in I \;\middle|\; g_i^{(L)} = g_i^{(U)} \right\} \end{aligned}$$

and parameters $\mu_k > 0$. Further, we have that $\mu_k \to 0$ for $k \to \infty$ such that in the limit minimizing φ_{μ_k} equals minimizing f.
As usually in line search approaches, the step direction $p^{(k)}$ for the next iterate $x^{(k+1)}$ is obtained from the linearization of the KKT-conditions with φ_{μ_k} as the objective function. The step length γ_k is obtained in a backtracking manner, i. e. a sequence of step lengths $\gamma_{k,l}$ is computed until at least one acceptance criterion is fulfilled.

The acceptance criteria are

$$\left\| c\left(x^{(k)} + \gamma_{k,l} p^{(k)}\right) \right\| \leq (1 - \varepsilon_c) \left\| c\left(x^{(k)}\right) \right\|$$

and

$$\varphi_{\mu_k}\left(x^{(k)} + \gamma_{k,l} p^{(k)}\right) \leq \varphi_{\mu_k}\left(x^{(k)}\right) - \varepsilon_\varphi \left\| c\left(x^{(k)}\right) \right\|$$

for constants $\varepsilon_c,\ \varepsilon_\varphi \in (0,1)$. That means, an iterate is accepted if either the violation of equality constraints is sufficiently decreased or the objective function is significantly improved. However, in order to avoid cycling, e. g., between two points that alternately improve either one of these criteria, a filter is introduced. The filter saves prohibited (e. g. worse) combinations of $\|c(\cdot)\|$ and φ_{μ_k} from some of the previous iterates.
This is the main idea of the algorithm. However, when dealing with a general optimization problem like (A.14), one may encounter a wide range of problematic situations such as that the step direction cannot be computed due to a singular matrix, the Maratos effect (i. e. a step direction increases both the constraint violation and the objective function) or the progress made by the iterate is too small. As we see in the literature mentioned before, the IPOPT algorithm includes a strategy for many kinds of such troubles.
In theory, the IPOPT algorithm is globally convergent if a number of assumptions hold true. They include continuity assumptions regarding the original objective function and constraints as well as linear independence of active constraints but also seemingly technical issues like uniformly boundedness or uniformly positive definiteness of certain matrices on particular sets. However, it does not appear as if, in particular, the technical assumptions can be checked a-priori. Furthermore, as it was pointed out in Vigerske et al. (2016), the available software ensures only local solutions to be found. Hence, unfortunately, also in this case, we can only numerically determine whether the optimization software converges for the tasks in the LIPMP algorithms.

B. Documentation

In this chapter, we explain an implementation routine for the (L)IPMP algorithms introduced in this thesis. For both cases, we make use of the descriptions for practical use given in Telschow (2014) and use the same structure to preprocess certain data, like e. g. the (starting) dictionary or the measurements. We present a near-implementation pseudo-code based description.
We utilize the GNU Scientific Library (GSL; Galassi et al., 2019) as many of its mathematical tools make the programming easier and the code better readable. Moreover, for solving the optimization problems, we include the NLopt library (Johnson, 2019) because it contains various optimization algorithms. Note that, consequentially, we suggest to implement the algorithms in a high-level programming language in order to use these libraries. Furthermore, this also enables more efficiency and a higher possible data amount. Our implementation is written in C/C++ and, hence, we often start counting at 0. Further, note that, if we use objects from the GSL as for instance vectors or matrices, we have to free these objects after use (at least at the end of the programme).
We structure the whole implementation process as follows. The main-function is implemented in a file `exec.c`. There, we first define parameters of the experiment. Then we step into the function `matchingpursuit` which runs all aspects of the respective algorithm. First, we compute some general items, generate the measurements y for the matching pursuit and process specific values such as the inner products of the dictionary elements of a finite dictionary. At last, we run (at least) one of the (learning) inverse problem matching pursuits. Note that with

```
start = clock()
     { do something }
end = clock()
cpu_time_used = ((double) (end - start))/CLOCKS_PER_SEC
```

we can measure the CPU-time for any part of the code (`do something`). Further, with `if`-loops, we distinct the cases in which we either compute or read certain values from files of previous computations (see also below `do_preproc` and `do_save`). At last, we also distinct in this way necessary and unnecessary computations with respect to which trial function classes are in the dictionary contained (see also below `do_sh`, `do_apk`, `do_apw` and `do_sl`). Note that we explain the most simple form of implementing an (L)IPMP algorithm including an efficient preprocessing for the application of the learnt dictionary. Hence, if one aims to implement the IPMP and the LIPMP algorithms simultaneously, the variables need to be doubled in the preprocessing.
We start with the file `exec.c`. Note that this file contains all needed header files.

In the main-function, we first define the experiment parameters.

1. Physical parameters
   (see Bettadpur, 2012; Dahle et al., 2013; National Geospatial-Intelligence Agency, Office of Geomatics (SN), EGM Development Team, 2008; Watkins and Yuan, 2012)

   - `radius_earth`
     The radius of the (spherical) Earth is fixed at 6378136.3 m or 6378136.46 m if the data from the GFZ is used.
   - `satellite_height`
     A variable satellite height. For instance, 500000.0 m as the GRACE orbiter started at roughly this height.
   - `grav_constant`
     The gravitational constant is fixed to be $3986004.415 \cdot 10^8 \ \frac{\mathrm{m}^3}{\mathrm{s}^2}$. This constant corresponds to the product of the gravitational constant and the mass of the Earth.

2. Data parameters

   - `reuter_data`
     We explain the case for using a Reuter grid for the location of the measurements $\eta^{(i)}$, $i = 1, ..., \ell$ (confer Remark 4.2.4). For this grid, we only need to set a parameter for its size (confer Definition A.1.2).
   - `DH_phi, DH_t`
     For the evaluation of the approximation and the computation of the approximation error, we suggest to use a Driscoll-Healy grid (confer Definition A.1.1). For this grid, we need two parameters: one for the number of steps in the longitude, i. e. `DH_phi`, and one for the latitude, i. e. `DH_t`.
   - `degree_data`
     Gives the maximal degree up to which we evaluate the solution f given in spherical harmonics, for instance, if we use the EGM2008 or GRACE data.
   - `name, name_file`
     If `name` is set to 1, the coefficients from the file with filename `name_file` are used.
   - `name_sol, name_dat`
     Filenames where the solution f evaluated at the Driscoll Healy grid (`name_sol`) and the Reuter grid (`name_dat`) is stored.
   - `do_data_sol, do_data_inp`
     If `do_data_sol` is set to 1, the solution f is evaluated at the Driscoll Healy grid and stored in a file. If `do_data_sol` is set to 0, the solution is not computed but read from a previously calculated file of the name `name_sol`. The value `do_data_inp` is similar to `do_data_sol` but with

respect to the computation of the measurements at a Reuter grid and the file `name_dat`.

- `scale_J`
  Gives the scale J for the low or band pass filter ($s = 2^{-J}$) of the Cubic polynomial scaling function (confer Example 3.3.13) which is used for destriping the GRACE data.

3. Dictionary parameters
   - `do_sh, do_apk, do_apw, do_sl`
     If set to 1, we use spherical harmonics (`do_sh`), Abel–Poisson low pass filters (`do_apk`), Abel–Poisson band pass filters (`do_apw`) or Slepian functions (`do_sl`) as dictionary elements.
   - `degree_SH`
     Gives the maximal degree of the spherical harmonics in the dictionary.
   - `reuter_ker, reuter_wav`
     We include Abel–Poisson low and band pass filters with respect to a set of centres $\chi \in \Omega$. This set is a Reuter grid. The value `reuter_ker` and `reuter_wav` give the necessary parameter for the point grids (i. e. G in Definition A.1.2). Note that the grids for the low and band pass filter coincide if `reuter_ker = reuter_wav`.
   - `num_scales_ker, scales_ker, num_scales_wav, scales_wav`
     The values `num_scales_ker` and `num_scales_wav` give the number of different scales of Abel–Poisson low and band pass filters, respectively, that are used in the dictionary. The values themselves are stored in `scales_ker` and `scales_wav`, respectively. Note that we combine each scale with the Reuter grid of the centres.
   - `num_caps, sphcaps`
     The value `num_caps` gives the number of spherical caps used for Slepian functions of the dictionary. In particular, this is the number of different values of $\theta = \arccos(c) \in [0, \pi]$ which are stored in `sphcaps`. Note that each value is combined with each combination of Euler angles given in `Ealpha, Ebeta, Egamma`.
   - `num_Ealpha, Ealpha, num_Ebeta, Ebeta, num_Egamma, Egamma`
     The number of different values of the Euler angle $\alpha \in [0, 2\pi[$ used in Slepian functions of the dictionary is given in `num_Ealpha`. The values themselves are stored in `Ealpha`. Analogously, there are variables `num_Ebeta` and `Ebeta` for $\beta \in [0, \pi]$ as well as `num_Egamma` and `Egamma` for $\gamma \in [0, 2\pi[$
   - `bandlimit`
     The band-limit of the Slepian functions in the dictionary.

4. Termination criteria
   - `iterations`
     Gives the maximal number of iterations for a matching pursuit.

- `size_res`
  Gives the minimal size allowed for $\|R^N\|_{\mathbb{R}^\ell} / \|y\|_{\mathbb{R}^\ell}$ in the matching pursuit. Note that, in practice, this termination criterion is the most sensible one. Thus, the number of iterations should be not too small. Further, note that the value should depend on the noise level given in `noise_perc`.

5. Experiment setting

   a) General

   - `regpar`
     Gives a value for $\lambda_0(\delta)$ which defines $\lambda(\delta, y^\delta) = \lambda_0(\delta)\|y\|_{\mathbb{R}^\ell}$ later.
   - `sobolev`
     Gives the value s of the Sobolev space $\mathcal{H}_s(\Omega)$.
   - `sobolev_eval`
     Maximal degree at which the series of the $\mathcal{H}_2(\Omega)$-inner products of the Abel–Poisson low and band pass filters (confer (3.23), (3.24) and (3.26)) are truncated.
   - `non_ortho`
     If set to 1, the (L)RFMP is computed.
   - `ortho`
     If set to 1, the (L)ROFMP is computed.
   - `ortho_it`
     Gives the number K for the iterated ROFMP (see Algorithm 4) as well as for the restart of the LR(O)FMP (see Remark 7.3.5). A restart technique is used in the (L)IPMP algorithms if `ortho_it` is lower than `iterations`. Then the restart takes place after `ortho_it` iterations.
   - `noise_perc` $\in [0, 1]$
     Gives the percentage of the noise.

   b) Learning

   - `splinesize`
     Gives the size ε for the spline in the LROFMP (confer Section 7.3).
   - `smoothing, smooth_sh, smooth_shsl`
     If `smoothing` is set to 1, the additional smoothing mechanism (confer Remark 7.3.8) is used in the LIPMP algorithms. Hence, in the first `smooth_sh` iterations of the LIPMP algorithm, only spherical harmonics are allowed to be chosen. In the next `smooth_shsl` iterations, it is allowed to choose a spherical harmonic or a Slepian function. If `smoothing` is set to 0, the full dictionary $\mathcal{D}^{\mathrm{Inf}}$ is used right from the start in the LIPMP algorithm.
   - `iterated_application`
     If set to 1, the learnt dictionary is applied in an iterated fashion

(confer Remark 7.3.4). If set to 0, the IPMP can choose from the full learnt dictionary in each iteration.

- `opt_routine, local_opt_routine`
  The name `opt_routine` defines the optimization routine from the NLopt library used in the global optimization. Similarly, the name `local_opt_routine` defines the optimization routine in the local optimization.
- `feasibility`
  Gives the tolerance for the feasibility in an optimization process. We fix it to a small negative value (-10^{-8}) such that infeasible values will not be considered as successful solutions.
- `abs_tol_f`
  An optimization process terminates if the value of the objective function at the current and the previous iterate differ less than this value. We fix it to 10^{-8}.
- `abs_tol_x`
  An optimization process terminates if the current and the previous iterate differ component-by-component less than this value. We fix it to 10^{-8}.
- `maxeval`
  Maximal number of function evaluation in an optimization process. From experience, this value should be at least of the size 10^4 for the here considered data.
- `maxtime`
  Maximal time in seconds that an optimization process is allowed to take. We propose to set it to 200.

6. General distinction

   - `raw, training`
     Determines whether an IPMP algorithm (`raw`=1) or an LIPMP algorithm (`training`=1) or both (`raw=training=1`) are run.
   - `do_preproc, do_save`
     If `do_preproc` is set to 1 the preprocessing, of e. g. inner products, is computed. If also `do_save` is set to 1, the preprocessing is saved in the folder "Preproc". If `do_preproc` is set to 0, the respective data is read from the files in the folder "Preproc".

All of these values are the arguments of the function `matchingpursuit`. This function is the spine of the computation process in the sense that it contains most of the declaration of variables and function calls to smaller computation units. Note that, though most variables are declared in this function, their values are set in the smaller units. In particular, these smaller units are:

1. `reuter_size`,
2. determine overall maximal degree of spherical harmonics with respect to the dictionary,
3. `precomputation`,
4. `initializeLookUp`,
5. `computedata`,
6. `computeD`,
7. `computeS`,
8. `rfmp`,
9. `rofmp`,
10. `lrfmp`,
11. `lrofmp`,
12. copy learnt dictionary from structure book to `LookUp`,
13. `computeDTDlearnt`,
14. `computeSlearnt`.

We discuss number 1 – 7 in the next section, Appendix B.1. Number 8 and 9 are explained in Appendix B.2 and number 10 – 14 are considered in Appendix B.3.

B.1. Common preprocessing

We discuss the preprocessing of the (L)IPMP algorithms, i. e. the part of the function `matchingpursuit` that contains the computations up to `computeS` (included).

B.1.1. First computations

We consider simple computations that follow directly from the parameters defined in `exec.c`.

Immediate values We define some further variables:

- `orbit, rel_orbit`
  The value `orbit` gives the distance of the satellite orbit from the centre of the coordinate system. Note that we consider constant satellite height. Hence, the distance is constant as well. We scale the distance of the satellite orbit such that the surface of the Earth is given by the unit sphere. This relative distance is denoted by `rel_orbit`.
- `num_sh_data, size_sh_data`
  The value `num_sh_data` gives the number of spherical harmonics considered in the data. The number of rows in the table of coefficients in `name_file` is given by `size_sh_data`.
- `gridpointsDH`
  Gives the number of grid points of the Driscoll Healy grid.

- `num_sh`
  Gives the number of spherical harmonics in the dictionary.
- `num_sl, full_num_sl`
  The number of Slepian function per localization region is given by `num_sl`. The number of Slepian functions in the dictionary is given by `full_num_sl`.
- `num_dic_ele`
  Gives the size of the dictionary.
- `sobolevseq`
  Stores the values of the Sobolev sequence.
- `l`
  The number of measurements $y_i \in \mathbb{R}$ for $i = 1, ..., \ell$=`l` from Remark 4.1.1 and Remark 4.2.4. This value is computed via `reuter_size(·)` using the data parameter `reuter_data`. This function computes the vector γ from Definition A.1.2 because `l` is the sum of the entries of γ.
- `phi_eta, t_eta, eta`
  Store the Cartesian values of $\eta^{(i)}$ (confer Remark 4.1.1 and Remark 4.2.4) in `eta` as well as the longitude and the latitude of each point in `phi_eta` and `t_eta`, respectively.
- `max_degree_m`
  Gives the maximum of (all) `degree_SH` and `bandlimit`.
- `fnsh_at_data`
  Stores the values of the fully normalized spherical harmonics (2.14) up to degree `max_deg_m` at the data points $\eta^{(i)}$.
- `num_cen_ker, num_cen_wav`
  Similarly as with `l`, we compute the number of centres, i. e. Reuter grid points, for the Abel–Poisson low and band pass filters from `reuter_ker` and `reuter_wav`, respectively.
- `phi_xi, t_xi, xi, phi_zeta, t_zeta, zeta`
  Store the values of the centres of the Abel–Poisson low pass filters in a Cartesian form in `xi` as well as the longitude and the cosine of the latitude of each point in `phi_xi` and `t_xi`, respectively. Analogously, we have `phi_zeta, t_zeta, zeta` for the Abel–Poisson band pass filters.
- `ev_sleps`
  Stores the Fourier coefficients together with the respective set of characteristics $\theta = \arccos(c)$, α, β, γ of the Slepian functions in the dictionary.
- `wignercoeffs_a, wignercoeffs_b, wignercoeffs_c, wignercoeffs_d`
  Store the values of the coefficients $a^n_{k,j}$, $b^n_{k,j}$, $c^n_{k,j}$ and $d^n_{k,j}$ from Definition 3.1.7 up to `bandlimit`. Note that we can often re-use these values throughout the LIPMP algorithms.

After the declaration, we set the values of these variables in

```
precomputation(  radius_earth, satellite_height, reuter_data, l,
                 phi_eta, t_eta, eta, DH_phi, DH_t, degree_data,
                 do_sh, do_apk, do_apw, do_sl, degree_SH,
                 reuter_ker, num_scales_ker, reuter_wav,
                 num_scales_wav, num_caps, sphcaps, num_Ealpha,
                 num_scales_wav, num_caps, sphcaps, num_Ealpha,
                 Ealpha, num_Ebeta, Ebeta, num_Egamma, Egamma,
                 bandlimit, sobolev, sobolev_eval, orbit,
                 rel_orbit, num_sh_data, size_sh_data, gridpointsDH,
                 sobolevseq, num_sh, fnsh_at_data, max_degree_m,
                 num_cen_ker, phi_xi, t_xi, xi, num_cen_wav,
                 phi_zeta, t_zeta, zeta, num_sl, full_num_sl,
                 ev_sleps, wignercoeffs_a, wignercoeffs_b,
                 wignercoeffs_c, wignercoeffs_d, bandlimit,
                 num_dic_ele, do_preproc, do_save ).
```

The function `precomputation` contains the following calculations:

1. `orbit = radius_earth + satellite_height`
2. `rel_orbit = orbit/radius_earth`
3. `num_sh_data = (degree_data+1)`^{2}
4. `size_sh_data = 0.5(degree_data + 2)(degree_data + 1)`
5. `gridpointsDH = (DH_phi+1)(DH_t+1)`
6. $n = 0,...,$`sobolev_eval:  sobolevseq(`n`) =` $(n+0.5)^2$
7. `Reuter(reuter_data, l, phi_eta, t_eta, eta)`
8. `fnsh(max_degree_m, l, phi_eta, t_eta, fnsh_at_data)`
9. `num_sh = (degree_SH+1)`^{2}
10. `Reuter(reuter_ker, num_cen_ker, phi_xi, t_xi, xi)`
11. `Reuter(reuter_wav, num_cen_wav, phi_zeta, t_zeta, zeta)`
12. `num_sl = (bandlimit+1)`^{2}
13. `full_num_sl`
    `= num_sl · num_caps · num_Ealpha · num_Ebeta · num_Egamma`
14. `initializewignercoeffs(bandlimit,`
    `wignercoeffs_a, wignercoeffs_b,`
    `wignercoeffs_c, wignercoeffs_d)`
15. `sleps(bandlimit, num_sl, num_caps, num_Ealpha, num_Ebeta,`
    `num_Egamma, sphcaps, Ealpha, Ebeta, Egamma, ev_sleps,`
    `wignercoeffs_a, wignercoeffs_b, wignercoeffs_c,`
    `wignercoeffs_d)`
16. `num_dic_ele = num_sh · do_sh`
    `+ num_scales_ker · num_cen_ker · do_apk`
    `+ num_scales_wav · num_cen_wav · do_apw`
    `+ full_num_sl · do_sl`

The function `Reuter(G, l, phi, t, cart)` as used in `7.`, `10.` and `11.` implements Definition A.1.2. The longitude is saved in the third argument `phi` and the latitude in the fourth arguments `t`. Using (2.1), the Cartesian values of the grid points are computed and stored in the fifth argument `cart`.
The function `fnsh(N, len_t, phi, t, fnsh_eval)` as used in `8.` computes the values of the fully normalized spherical harmonics up to degree `N` and at the `len_t` gridpoints given via their longitude and latitude in `phi` and `t`, respectively. The values are stored in `fnsh_eval`. The computation can be realized immediately from definition (2.14). The associated Legendre functions can be computed either via Algorithm 9 or Algorithm 11, Algorithm 12 and Algorithm 13 dependent on `max_deg_m`.
In `14.`, the coefficients of the recursion formulas of the Wigner rotation matrices is implemented in `initializewignercoeffs(bandlimit, a, b, c, d)`. They are directly obtained from Definition 3.1.7 and can be re-used in the optimization process.
The computation of the coefficients of the Slepian functions in the dictionary is done in

```
sleps(  bandlimit, num_sl, num_caps, num_Ealpha, num_Ebeta,
        num_Egamma, sphcaps, Ealpha, Ebeta, Egamma, ev_sleps,
        wignercoeffs_a, wignercoeffs_b, wignercoeffs_c,
        wignercoeffs_d).
```

The first arguments up to `Egamma` contain the characteristics of the Slepian functions. The Fourier coefficients will be stored in `ev_sleps`. The Wigner rotation matrices as in Definition 3.1.7 up to `bandlimit` can be computed via a separate call (i. e. `initializewigner`) in this function. Then the implementation of `sleps` contains the following steps. For each size $\theta = \arccos(c)$ stored in `sphcap` of the spherical cap in the dictionary, we compute

- for each order $j = -$`bandlimit`$, \ldots,$`bandlimit`:
  - compute localization matrix M^j as described in Theorem 3.1.11 and thereafter
  - solve eigenvalue problem
    (e. g. with `gsl_eigen_symmv` from the GSL)
  - save real-valued coefficient of the Slepian functions
- for each rotation specified in the dictionary (i. e. each triple $(\alpha,\ \beta,\ \gamma) \in$ `Ealpha` $\times$ `Ebeta` $\times$ `Egamma`:
  - compute Wigner rotation matrices D^n as given in Definition 3.1.7 using the coefficients computed in `initializewignercoeffs` and stored in `wignercoeffs_x` for `x = a, b, c, d`
  - from the real-valued coefficients of the Slepian functions compute the complex valued coefficients (confer (2.15))
  - rotate the complex-valued coefficients of the Slepian functions using the rotation matrices (confer Theorem 3.1.9 and Example 3.1.10)

- from the rotated complex-valued coefficients of the Slepian functions compute the (rotated) real-valued coefficients (confer (2.16))
- save the coefficients together with their characteristics $(c,\ \alpha,\ \beta,\ \gamma)$ in `ev_sleps`

The LookUp-Table A certain look-up table proved to be very convenient. The matrix `LookUp` stores the characteristics of each dictionary element. First, we define its size. If the dictionary contains Slepian functions, we have

$$\texttt{sizeLookUp} = 5 + \texttt{num\_sl} \tag{B.1}$$

in order to ensure that it is large enough to contain the Fourier coefficients of the Slepian functions. If the dictionary does not contain Slepian functions, it suffices to set

$$\texttt{sizeLookUp} = 6. \tag{B.2}$$

Then we initialize `LookUp` in

```
initializeLookUp(  do_sh, do_apk, do_apw, do_sl, degree_SH,
                   num_scales_ker, num_cen_ker, scales_ker, phi_xi,
                   t_xi, num_scales_wav, num_cen_wav, scales_wav,
                   phi_zeta, t_zeta, num_sl, full_num_sl, ev_sleps,
                   LookUp ).
```

Note that, in `LookUp`, we stick to the order of the dictionary (spherical harmonics $\rightarrow$ Abel–Poisson low pass filters $\rightarrow$ Abel–Poisson band pass filters $\rightarrow$ Slepian function) which allows us later to find the respective dictionary element faster. In general, the matrix naturally has different entries dependent on the trial function class. However, the first column is used for categorizing the trial function classes in all cases. We identify spherical harmonics with category 1, Abel–Poisson low pass filters with 2, Abel–Poisson band pass filters with 3 and Slepian functions with 4.
For a spherical harmonic $Y_{n,j} \in \mathcal{D}$, the respective row of `LookUp` is given by

$$\left|\ 1\ \right|\ n\ \left|\ j\ \right|$$

and, thus, stores the identifier, the degree n and the order j.
In the sequel, with respect to the Abel–Poisson low and band pass filters, the elements of the set of centres $x/|x|$ shall be ordered for $i = 1$ to `num_cen_ker` or `num_cen_wav`, respectively. Similarly, we have the value r_j for the jth value of `scales_ker` or `scales_wav` for $j = 1, ..., $`num_scales_ker` or `num_scales_wav`, respectively. Further, we denote φ_x and $t_x = \cos(\theta_x)$ as the related longitude and latitude of x. Then we set the respective row of `LookUp` as

$$\left|\ 2\ \right|\ |x|\ \left|\ \varphi_x\ \right|\ t_x\ \left|\ i\ \right|\ j\ \left|\right.$$

for an Abel–Poisson low pass filter $K(x,\cdot)$ and

$$\left| \; 3 \; \right| \; |x| \; \left| \; \varphi_x \; \right| \; t_x \; \left| \; i \; \right| \; j \; \left|\right.$$

for an Abel–Poisson band pass filter $W(x,\cdot)$. Note that we also save the indices i and j in order to simplify the access to respective rows and columns in the upcoming preprocessing. At last, we set

$$\left| \; 4 \; \right| \; c \; \left| \; \alpha \; \right| \; \beta \; \left| \; \gamma \; \right| \; g_{0,0}^{(k,L)} \; \left| \; g_{1,-1}^{(k,L)} \; \right| \; g_{1,0}^{(k,L)} \; \left| \; \ldots \right.$$

for the Slepian functions where the content of the columns 2 to `sizeLookUp` is inherited from `ev_sleps`. Hence, we see the reason for choosing the number of columns in `LookUp` as given in (B.1) and (B.2), respectively.

B.1.2. Generating the data

Next, we generate necessary data from the solution f (confer Remark 4.2.4). We need the following variables.

- `phi_sol, t_sol, sol`
  Store the values of the gridpoints of the Driscoll Healy grid (for evaluation of the solution and the approximation; confer Definition A.1.1) in a Cartesian form in `sol` as well as the longitude and the cosine of the latitude of each point in `phi_sol` and `t_sol`, respectively.
- `fnsh_at_sol`
  Stores the fully normalized spherical harmonics (2.14) evaluated up to degree and order `max_degree_m` at the Driscoll Healy grid points `sol`.
- `solution`
  Stores the values of the solution f at the Driscoll Healy grid points.
- `data`
  Stores the value of the solution f at the Reuter grid points `eta`. We should store a replica `data_tmp` as well in case we run two algorithms successively.
- `norm_sol, norm_data`
  Gives the Euclidean norm of the solution f evaluated at the Driscoll Healy grid (`norm_sol`) and at the Reuter grid (`norm_data`).

These are used in

```
computedata(  gridpointsDH, DH_phi, phi_sol, DH_t, t_sol, sol,
              max_degree_m, fnsh_at_sol, do_apk, do_apw, do_sl,
              do_data_sol, do_data_inp, orbit, rel_orbit,
              radius_earth, grav_constant, name, name_file, scale_J,
              noise_perc, degree_data, size_sh_data, l, num_sh_data,
              phi_eta, t_eta, solution, data, name_sol, name_dat,
              norm_sol, norm_data, raw, training, do_preproc,
              do_save ).
```

Necessary data for an (L)IPMP algorithm is obtained by the following steps:

1. `DriscollHealyMeshed(DH_phi, phi_sol, DH_t, t_sol, sol)`
2. `fnsh(max_degree_m, gridpointsDH, phi_sol, t_sol, fnsh_at_sol)`
3. `gsl_rng * RNG = gsl_rng_alloc(gsl_rng_taus)`
4. read coefficients from `name_file`
5. compute a representation of f in spherical harmonics for each grid point of the Driscoll Healy grid `sol` and at the surface of the Earth (compute each grid point separately)
6. from these values $\widetilde{y}_i$, $i = 0, \ldots,$ `gridpointsDH-1` compute

$$\texttt{norm\_sol} = \sqrt{\sum_{i=0}^{\texttt{gridpointsDH-1}} \widetilde{y}_i^2}$$

7. compute a representation of f in spherical harmonics for each grid point of the Reuter grid `eta` at `rel_orbit` (compute each grid point separately)
8. for each value $y_i =$ `data(`i`)` on the Reuter grid, add noise (confer (9.1)), e. g. via
   `random = gsl_ran_gaussian(RNG, 1.0)`
   `data(`i`) = data(`i`) · (1.0 + noise_perc · random)`, $i = 0$...,`l-1`
9. from these values `data(`i`)` $= y_i^\delta$, $i = 1, \ldots,$ `l-1` compute

$$\texttt{norm\_data} = \sqrt{\sum_{i=0}^{\texttt{l-1}} \texttt{data}(i)}$$

The function `DriscollHealyMeshed(num_phi, phi, num_t, t, cart)` in `1.` computes the Driscoll Healy grid straight-forwardly from Definition A.1.1. The longitude has `num_phi+1` distinct values and the cosine of the latitude contains `num_t+1` different values. The meshed values are stored in `phi` and `t`, respectively. The corresponding Cartesian values (confer (2.1)) are saved in `cart`.

B.1.3. Preprocessing

At last, we consider the preprocessing as it was described by Telschow (2014).

The matrix D We evaluate the images of the dictionary elements under the operator $\mathcal{T}_\neg$. In our case, this means, we consider the upward continued dictionary elements. These values can be efficiently stored in a matrix. For this, we define the following variables

- `xi_times_eta, zeta_times_eta`
  For all data points $\eta^{(i)}, i = 0, ..., \texttt{l-1}$ in `eta`, the values

$$\frac{x^{(j)}}{\left|x^{(j)}\right|} \cdot \eta^{(i)}$$

  are stored for all centres $x^{(j)}$ with $j = 0, ...,$ `num_cen_ker-1` of Abel–Poisson low pass filters in `xi_times_xi` and $x^{(j)}$ with $j = 0, ...,$ `num_cen_wav-1` of Abel–Poisson band pass filters in `zeta_times_zeta`.
- `matrixD`
  The column $i = 0, ...,$ `num_dic_ele-1` stores the value $\mathcal{T}_\neg d_i \in \mathbb{R}^{\texttt{l}}$ (see (2.27), (7.41), (7.42) and (3.18)) for all dictionary elements d_i.
- `matrixDTD`
  The entry (i, j) for $i,\ j = 0, ...,$ `num_dic_ele-1` of this matrix stores the value $\langle \mathcal{T}_\neg d_i, \mathcal{T}_\neg d_j \rangle_{\mathbb{R}^{\texttt{l}}}$ for all pairs of dictionary elements (d_i, d_j).

and use them in

```
computeD(  do_apk, do_apw, l, eta, num_cen_ker, xi, xi_times_eta,
           num_cen_wav, zeta, zeta_times_eta, LookUp, rel_orbit,
           fnsh_at_data, degree_SH, bandlimit, num_dic_ele, matrixD,
           matrixDTD, do_preproc, do_save, raw, training ).
```

Note that the latter two arguments are only used to specify filenames. In this function, the following steps are taken:

1. compute

$$\frac{x^{(j)}}{\left|x^{(j)}\right|} \cdot \eta^{(i)}$$

   for all centres of
   a) Abel–Poisson low pass filters, i. e. $j = 0, ...,$ `num_cen_ker-1`
   b) Abel–Poisson band pass filters, i. e. $j = 0, ...,$ `num_cen_wav-1`

   Note that, if `num_cen_ker` = `num_cen_wav`, step 1b can be substituted by copying the values of 1a to the respective structures.

2. for all dictionary elements (i. e. i=0,...,num_dic_ele-1)
   dependent on the type of trial function (i. e. LookUp(i,0) = 1, 2, 3 or 4)
   for all Reuter grid points (i. e. j=0,...,l-1)
   compute matrixD(j,i) = $\mathcal{T}d_i(\eta^{(j)})$ via

	(2.27) (LookUp(i,0)=1)	using	rel_orbit, LookUp(i,1), fnsh_at_data,
	(7.41) (LookUp(i,0)=2)	using	rel_orbit, |x|, xi_times_eta,
	(7.42) (LookUp(i,0)=3)	using	rel_orbit, |x|, zeta_times_eta,
and	(3.18) (LookUp(i,0)=4)	using	LookUp(i,:), rel_orbit, fnsh_at_data.

3. compute matrixDTD = matrixD$^{\mathrm{T}}\cdot$ matrixD

Note that, if the fully normalized spherical harmonics are used, we may also normalize the Abel–Poisson low and band pass filters with respect to the $\mathrm{L}^2(\Omega)$-norm in step 2. This also holds for the matrix S we describe next. Further, note that we have to consider this normalization in the processing of candidates in the learning algorithms as well. In the objective function (7.103), we do not have to consider this normalization as it cancels out. At last, note that we could use the symmetry of matrixDTD.

The matrix S We also aim to preprocess the $\mathcal{H}_2(\Omega)$-inner products of all pairs of dictionary elements. For this, we need

- fnsh_at_cen_ker, fnsh_at_cen_wav
  Store the fully normalized spherical harmonics (2.14) evaluated either at the centres of the Abel–Poisson low pass filters (fnsh_at_cen_ker) or at the centres of the Abel–Poisson band pass filters (fnsh_at_cen_wav). Note that we can save runtime if the centres of the Abel–Poisson low and band pass filters coincide.
- xi_times_xi, xi_times_zeta, zeta_times_zeta
  Store the inner products

$$\frac{x^{(i,K)}}{\left|x^{(i,K)}\right|}\cdot\frac{x^{(j,K)}}{\left|x^{(j,K)}\right|}, \qquad i,j=0,\ldots,\texttt{num\_cen\_ker-1},$$

  in xi_times_xi,

$$\frac{x^{(i,K)}}{\left|x^{(i,K)}\right|}\cdot\frac{x^{(j,W)}}{\left|x^{(j,W)}\right|}, \qquad i=0,\ldots,\texttt{num\_cen\_ker-1}, j=0,\ldots,\texttt{num\_cen\_wav-1},$$

in `xi_times_zeta` and

$$\frac{x^{(i,W)}}{\left|x^{(i,W)}\right|} \cdot \frac{x^{(j,W)}}{\left|x^{(j,W)}\right|}, \qquad i,j = 0,...,\texttt{num\_cen\_wav-1}$$

in `zeta_times_zeta` of the centres of the Abel-Poisson low and band pass filters $x^{(\bullet)}$. Note again that we can substitute some of the computation with copying if `num_cen_ker = num_cen_wav`.

- `cn_hh, cn_hb, cn_bb`
  Give the coefficients from (A.5) for $n = 0,...,$`sobolev_eval`. The matrix `cn_hh` is used for all combinations of

$$\tau = \left|x^{(i)}\right| \left|x^{(j)}\right|, \qquad i,j = 0,...,\texttt{num\_cen\_ker-1},$$

  `cn_hb` for all $i = 0,...,$`num_cen_ker-1`, $j = 0,...,$`num_cen_wav-1`, and `cn_bb` for all $i,j = 0,...,$`num_cen_wav-1`.
- `matrixS`
  The entry (i,j) for $i,\ j = 0,...,$ `num_dic_ele-1` of this matrix stores the value $\langle d_i, d_j \rangle_{\mathcal{H}_2(\Omega)}$ for any two dictionary elements d_i and d_j. The first thing to note is the symmetry of the inner product. Next, note that, in the case that we run an LIPMP algorithm, we use `matrixS_small` instead of `matrixS` and copy the former one into the latter one afterwards. This is done in order to plan ahead for an efficient preprocessing of the learnt dictionary. In particular, in this case, we structure `matrixS` in quadrants QI to QIV:

QI	QII
QIII	QIV,

  where QI = `matrixS_small` is of the size `num_dic_ele` $\times$ `num_dic_ele`, QII is of the size `num_dic_ele` $\times$ `iteration_number`, QIII = QII$^{\mathrm{T}}$ and, accordingly, QIV is of the size `iteration_number` $\times$ `iteration_number`. We refer to the quadrants QII and QIV when describing how learnt dictionary elements need to be processed for the next iteration in the LIPMP algorithm.

We compute and store the inner products in

```
computeS( sobolevseq, sobolev_eval, num_dic_ele, LookUp, matrixS,
          do_apk, do_apw, fnsh_at_cen_ker, max_degree_m,
          num_scales_ker, scales_ker, num_cen_ker, phi_xi, t_xi, xi,
          fnsh_at_cen_wav, num_scales_wav, scales_wav, num_cen_wav,
          phi_zeta, t_zeta, zeta, xi_times_xi, xi_times_zeta,
          zeta_times_zeta, bandlimit, do_preproc, do_save, raw,
          training ).
```

Again, the last two arguments are used for setting filenames if the preprocessing shall be saved. The function computes the following steps:

1. fnsh(max_degree_m, num_cen_ker, phi_xi, t_xi, fnsh_at_cen_ker)
   fnsh(max_degree_m, num_cen_wav, phi_zeta, t_zeta, fnsh_at_cen_wav)
2. compute cn_hh, cn_hb and cn_bb for $n = 0, ...,$ sobolev_eval with sobolevseq and
   $|x^{\bullet}| \in$ scales_ker or
   $|x^{(i)}| \in$ scales_ker, $|x^{(j)}| \in$ scales_wav or
   $|x^{\bullet}| \in$ scales_wav, respectively
3. compute xi_times_xi with $x^{(\bullet)}/|x^{(\bullet)}| \in$ scales_ker
4. compute xi_times_zeta with $x^{(i)}/|x^{(i)}| \in$ scales_ker and $x^{(j)}/|x^{(j)}| \in$ scales_wav
5. compute zeta_times_zeta with $x^{(\bullet)}/|x^{(\bullet)}| \in$ scales_wav
6. for all combinations of dictionary elements (i. e. i, j=0,...,num_dic_ele-1)
   dependent on the types of trial function
   (i. e. LookUp(i,0), LookUp(j,0) $\in \{1,2,3,4\}$)
   compute matrixS(i,j) = $\langle d_i, d_j \rangle_{\mathcal{H}_2(\Omega)}$ via

(3.19)	if	LookUp(i,0) = 1, LookUp(j,0) = 1
	using	sobolevseq, LookUp(i,:), LookUp(j,:)
(3.20)	if	LookUp(i,0) = 1, LookUp(j,0) = 2, and vice versa
	using	sobolevseq, LookUp(i,:), LookUp(j,:), fnsh_at_cen_ker
(3.23)	if	LookUp(i,0) = 2, LookUp(j,0) = 2
	using	LookUp(i,:), LookUp(j,:), xi_times_xi, cn_hh, Algorithm 8
(3.21)	if	LookUp(i,0) = 1, LookUp(j,0) = 3, and vice versa
	using	sobolevseq, LookUp(i,:), LookUp(j,:), fnsh_at_cen_wav
(3.24)	if	LookUp(i,0) = 2, LookUp(j,0) = 3, and vice versa
	using	LookUp(i,:), LookUp(j,:), xi_times_zeta, cn_hb, Algorithm 8
(3.26)	if	LookUp(i,0) = 3, LookUp(j,0) = 3
	using	LookUp(i,:), LookUp(j,:), zeta_times_zeta, cn_bb, Algorithm 8
(3.22)	if	LookUp(i,0) = 1, LookUp(j,0) = 4, and vice versa
	using	sobolevseq, LookUp(i,:), LookUp(j,:)
(3.25)	if	LookUp(i,0) = 2, LookUp(j,0) = 4, and vice versa
	using	sobolevseq, LookUp(i,:), LookUp(j,:), fnsh_at_cen_ker

(3.27) if LookUp(i,0) = 3, LookUp(j,0) = 4, and vice versa
using sobolevseq, LookUp(i,:), LookUp(j,:), fnsh_at_cen_wav

and (3.28) if LookUp(i,0) = 4, LookUp(j,0) = 4
using sobolevseq, LookUp(i,:), LookUp(j,:)

Note that we may have to take into consideration the $L^2(\Omega)$-normalization analogously as in computeD. The remaining interesting computation steps in the function matchingpursuit depend on the respective (L)IPMP algorithm we run. We will explain the separate cases in the next sections. Other than these, the function matchingpursuit takes care of freeing previously declared (structured) variables, like gsl_vector or gsl_matrix, as well as reset data to its initial values by using data_tmp if we run several (L)IPMP algorithms.

B.2. Implementing an IPMP algorithm

For both IPMP algorithms, the first step is to declare a few more variables:

- IP_data_Tupd
  Stores the values of $a_1(d_j)$ for all dictionary elements d_j with $j = 0, ...,$ num_dic_ele-1 (confer (7.7)). Initialized with

$$\left\langle y, \mathcal{T}_{\neg} d_j \right\rangle_{\mathbb{R}^\ell} = \sum_{i=0}^{\texttt{l-1}} \texttt{data}(i) \cdot \texttt{matrixD}(i,j)$$

  for all dictionary elements d_j.
- IP_appr_d
  Stores the values of $a_2(d_j)$ for all dictionary elements d_j with $j = 0, ...,$ num_dic_ele-1 (confer (7.7)). Initialized with zero entries.
- marks
  Is an advanced structure book (confer (4.8)). Each row represents one iteration. The columns are set analogously to LookUp. Additionally, between the first and the second column, it contains an additional column to store the value of the respective coefficients α_N and $\alpha_N^{(N)}$, respectively. Furthermore, it also saves the Cartesian coordinates of the centres of the chosen Abel–Poisson low and band pass filters.
- Appr
  Stores the current approximation f_N and $f_N^{(N)}$, respectively, evaluated on the Driscoll Healy grid sol. Initialized with zero entries, i. e. we set $f_0 \equiv 0$.
- TikhFun
  Stores the value of the Tikhonov functional in each iteration. We set

$$\texttt{TikhFun(0)} = \texttt{norm\_data}^2.$$

- RDE
  Stores the value of the relative data error in each iteration. We set

$$\texttt{RDE(0) = 1.}$$

- RAE
  Stores the value of the relative approximation error in each iteration in the (L)IPMP algorithm. We set

$$\texttt{RAE(0)} = \frac{\sqrt{\frac{\texttt{norm\_sol}^2}{\texttt{gridpointsDH}}}}{\texttt{norm\_sol}}.$$

- lambda0 = regpar · norm_data
- num_trials
  We choose the next basis element from the first num_trials dictionary elements. Initially, we set

$$\texttt{num\_trials = num\_dic\_ele.}$$

- iteration
  Gives the current iteration. Initially, we set

$$\texttt{iteration = 1.}$$

B.2.1. The RFMP algorithm

We explain the next steps in the function

```
rfmp(  l, data, norm_data, LookUp, num_dic_ele, sizeLookUp, num_sl,
       matrixD, matrixDTD, matrixS, regpar, free,
       iterated_application, ortho_it, iterations, size_res,
       gridpointsDH, solution, norm_sol, fnsh_at_sol, sol ).
```

after the declaration of the previously given variables. Note that the variable free is set to 1 if we apply the learnt dictionary (i. e. training=1). If we use a manually chosen dictionary (i. e. raw=1), then free is 0. The algorithm contains the following steps.

1. do
   a) lambda = regpar · norm_data/iteration
   (confer Remark 7.3.7)

$$\text{b) num\_trials} = \begin{cases} \texttt{iteration}, & \texttt{iterated\_application=1} \\ & \texttt{\& iteration < num\_dic\_ele} \\ \texttt{num\_dic\_ele}, & \textit{else} \end{cases}$$

   (confer Remark 7.3.4)

c) $i = 0,...,$ num_trials-1:
objective_function(i)
= (IP_data_Tupd(i) - lambda · IP_appr_d(i))^2
/ (matrixDTD(i,i) + lambda · matrixS(i,i))
(confer Algorithm 3)

d) ν = max_index(objective_function)
(confer Galassi et al. (2019))

e) marks(iteration-1,0) = LookUp(ν,0);
marks(iteration-1,1)
= (IP_data_Tupd(ν) - lambda · IP_appr_d(ν))
/ (matrixDTD(ν,ν) + lambda · matrixS(ν,ν))
(i. e. α, confer Algorithm 3)
$i = 1,...,$ sizeLookUp:
marks(iteration-1,i+1) = LookUp(ν,i)
if(marks(iteration-1,0) = 2 or 3)
marks(iteration-1,7) = $x_1/|x|$
marks(iteration-1,8) = $x_2/|x|$
marks(iteration-1,9) = $x_3/|x|$
(confer (2.1) and using LookUp(ν,2:3))

f) $i = 0,...,$ l-1:
data(i) = data(i) - marks(iteration-1,1) · matrixD(i,ν)
(i. e. R^N, confer Algorithm 3)
RDE(iteration) = $\sqrt{\sum_{i=0}^{\text{l-1}} \text{data}(i)^2}$ / norm_data

g) $i = 0,...,$ num_dic_ele-1:
IP_data_Tupd(i)
= IP_data_Tupd(i) - marks(iteration-1,1) · matrixDTD(ν,i)
(confer step 1f)
IP_appr_d(i)
= IP_appr_d(i) + marks(iteration-1,1) · matrixS(ν,i)
(confer (4.26))

h) $i = 0,...,$ gridpointsDH-1:
Appr(i)
= Appr(i) + marks(iteration-1,1)
$\times \begin{cases} \text{fnsh\_at\_sol(idx,}i\text{)}, & \text{marks(iteration-1,0) = 1} \\ \text{(3.12)}, & \text{marks(iteration-1,0) = 2} \\ \text{(3.15)}, & \text{marks(iteration-1,0) = 3} \\ \text{tmp}, & \text{marks(iteration-1,0) = 4} \end{cases}$
idx = marks(iteration-1,2) · marks(iteration-1,2)
+ marks(iteration-1,2) + marks(iteration-1,3)
tmp = $\sum_{k=0}^{\text{num\_sl-1}}$ marks(iteration-1,k+6) · fnsh_at_sol(k,i)
(by definition of the trial functions; confer Algorithm 3)

i) RAE(iteration)

$$= \frac{\sqrt{\frac{\sum_{i=0}^{\texttt{gridpointsDH-1}} (\texttt{Appr}(i) - \texttt{solution}(i))^2}{\texttt{gridpointsDH}}}}{\texttt{norm\_sol}}$$

j) TikhFun(iteration)
= TikhFun(iteration) - objective_function(ν)
(confer (4.24))

k) if (iteration % ortho_it == 0)
$i = 1, ...,$ num_dic_ele: IP_appr_d(i) = 0
(confer Remark 7.3.5)

l) iteration = iteration+1

2. while(iteration < iteration_number+1
&& RDE(iteration-1) < 2.0
&& RDE(iteration-1) > size_res)
(confer Section 4.5)

3. save Appr, marks, RDE, TikhFun, RAE in files

B.2.2. The ROFMP algorithm

This algorithm needs additionally some orthogonalization variables:

- beta stands for $\beta_n^{(N)}(d)$ from Theorem 4.4.2 for a dictionary element d
- mu
  Counter for the restarts of the iterated ROFMP algorithm. Initially, we have mu = 0.
- matrixB
  Stores the projection coefficients $\beta_n^{(N)}(d_i)$, $d_i \in \mathcal{D}$, $n = 0, ...,$ iteration-1, $N =$iteration. Initially, the matrix is zero.
- ind_cho, check_cho
  Recall that the dictionary is ordered as d_i for $i = 0, ...,$ num_dic_ele-1. The list ind_cho saves the numbers i of the chosen basis elements. The list check_cho memorizes whether a dictionary element was already chosen in the current ROFMP step. Both lists contain only zeros entries at the beginning.
- Sik
  Store the values $a_4(d)$, $b_4(d)$ and $b_5(d)$ for $d \in \mathcal{D}$. They are computed at the beginning of each iteration step. At initialization all values are set to zero.
- Val_d_Appr
  In the (L)RFMP the approximation can be easily computed in each iteration. For the (L)ROFMP, this leads to an increase of the runtime as the coefficients $\alpha_n^{(N)}$, $n = 1, ..., N$, change in every step. Hence, if we want to avoid this, we evaluate the chosen basis element d_N in each iteration N at the grid

points in sol, save the values in this matrix and combine the values with the respective coefficients after the iterations.

- matrixDfresh, matrixDTDfresh
  The former is a copy of matrixD and the latter one stores the main diagonal of matrixDTD after the preprocessing. Both are used in the restart procedure of the iterated ROFMP algorithm.

Then we consider

```
rofmp( l, data, norm_data, LookUp, num_dic_ele, sizeLookUp, num_sl,
       matrixD, matrixDTD, matrixS, regpar, free,
       iterated_application, iterations, ortho_it, size_res,
       gridpointsDH, solution, norm_sol, fnsh_at_sol, sol ).
```

This algorithm can be implemented as follows.

1. do

   a) lambda = regpar · norm_data/iteration

   (confer Remark 7.3.7)

   b) $\texttt{num\_trials} = \begin{cases} \texttt{iteration}, & \texttt{iterated\_application=1} \\ & \texttt{\& iteration < num\_dic\_ele} \\ \texttt{num\_dic\_ele}, & \textit{else} \end{cases}$

   (confer Remark 7.3.4)

   c) $k = 0,\ldots,$num_trials-1
   $i =$ mu,..., iteration-2
   $\texttt{s}(i) = \sum_{j=\texttt{mu}}^{\texttt{iteration-1}} \texttt{matrixB}(j,k) \cdot \texttt{matrixS}(\texttt{ind\_cho}(i),\texttt{ind\_cho}(j))$
   $\texttt{Sik}(0,k) = \sum_{i=0}^{\texttt{iteration-1}} \texttt{marks}(i,1) \cdot \texttt{s}(i)$
   $\texttt{Sik}(1,k) = \sum_{i=\texttt{mu}}^{\texttt{iteration-1}} \texttt{matrixB}(i,k) \cdot \texttt{s}(i)$
   $\texttt{Sik}(2,k) = \sum_{i=\texttt{mu}}^{\texttt{iteration-1}} \texttt{matrixB}(i,k) \cdot \texttt{matrixS}(k,\texttt{ind\_cho}(i))$

   (confer Telschow (2014))

   d) $i = 0,\ldots,$ num_trials-1:
   if (check_cho(i) = 0 && matrixDTD(i,i) > 1e-8)

   (confer Section 4.4, Realization in practice)

   objective_function(i)
   = (IP_data_Tupd(i) + lambda · (Sik(0,i) - IP_appr_d(i)))^2
   / (matrixDTD(i,i) + lambda · (matrixS(i,i) + Sik(1,i)
   - 2.0 · Sik(2,i)))

   (confer Algorithm 4)

   else
   objective_function(i) = 0

   e) ν = max_index(objective_function)

   (confer Galassi et al. (2019))

   f) marks(iteration-1,0) = LookUp(ν,0);
   marks(iteration-1,1)

```
      = (IP_data_Tupd(ν) + lambda · (Sik(0,ν) - IP_appr_d(ν)))
        / (matrixDTD(ν,ν) + lambda · (matrixS(ν,ν) + Sik(1,ν)
                                                - 2.0 · Sik(2,ν)))
```
(i. e. α, confer Algorithm 4)

```
   i = 1,..., sizeLookUp:
   marks(iteration-1,i+1) = LookUp(ν,i)
   if( marks(iteration-1,0) = 2 or 3 )
      marks(iteration-1,7) = x_1/|x|
      marks(iteration-1,8) = x_2/|x|
      marks(iteration-1,9) = x_3/|x|
```
(confer (2.1) and using `LookUp(ν,2:3)`)

```
g) check_cho(ν,1)
h) ind_cho(iteration-1,ν)
i) i = 0,..., l-1
   data(i) = data(i) - marks(iteration-1,1) · matrixD(i,ν)
```
(i. e. R^N, confer Algorithm 4)

```
   RDE(iteration) = sqrt(Σ_{i=0}^{l-1} data(i)^2) / norm_data
j) i = mu,..., iteration-1
   marks(i,1)
      = marks(i,1) - marks(iteration-1,1) · matrixB(i,ν)
```
(update to $\alpha_n^{(N+1)}$, confer Algorithm 4)

```
k) i = 0,..., num_dic_ele-1:
```
(for this step, confer Telschow (2014))

```
   IP_appr_d(i) = Σ_{j=0}^{iteration-1} marks(j,1) · matrixS(ind_cho(j),i)
```
(due to (4.29) and (4.35), respectively)

```
   if (check_cho(i) = 0 )
      beta = Σ_{j=0}^{l-1} matrixD(j,i) · matrixD(j,ν)
      beta = beta / matrixDTD(ν,ν)
```
(confer Theorem 4.4.2)

```
      j = 0,..., l-1:
        matrixD(j,i) = matrixD(j,i) - beta · matrixD(j,ν)
      IP_data_Tupd(i) = Σ_{j=0}^{l-1} data(i) · matrixD(j,i)
      matrixDTD(i,i) = Σ_{j=0}^{l-1} matrixD(i,i)^2
      j = mu ,..., iteration-2
        matrixB(j,i) = matrixB(j,i) - beta · matrixB(j,ν)
      matrixB(iteration-1,i) = beta
   matrixD(:,ν) = 0
   matrixB(:,ν) = 0
   matrixB(iteration-1,ν) = 1
l) i = 0,..., gridpointsDH-1:
   Val_d_Appr(iteration,i)
```

$$= \begin{cases} \texttt{fnsh\_at\_sol(idx,}i\texttt{),} & \texttt{marks(iteration-1,0) = 1} \\ \texttt{(3.12),} & \texttt{marks(iteration-1,0) = 2} \\ \texttt{(3.15),} & \texttt{marks(iteration-1,0) = 3} \\ \texttt{tmp,} & \texttt{marks(iteration-1,0) = 4} \end{cases}$$

idx = marks(iteration-1,2) · marks(iteration-1,2)
+ marks(iteration-1,2) + marks(iteration-1,3)

tmp = $\sum_{k=0}^{\texttt{num\_sl-1}}$ marks(iteration-1,k+6) · fnsh_at_sol(k,i)

(by definition of the trial functions; confer Algorithm 3)

m) TikhFun(iteration)
= TikhFun(iteration) - objective_function(ν)

(confer (4.33))

n) if (iteration % ortho_it == 0)

(for the next steps, confer (Telschow, 2014))

matrixD = matrixDfresh
matrixB(:,:) = 0
$i = 0, \ldots,$ num_dic_ele-1
check_cho(i) = 0
IP_data_Tupd(i) = $\sum_{j=0}^{\texttt{l-1}}$ data(i) · matrixD(j,i)
matrixDTD(i,i) = matrixDTDfresh(i)
mu = iteration+1
IP_appr_d(i) = 0

(for the last reset, confer Remark 7.3.5)

o) iteration = iteration+1

2. while(iteration < iteration_number+1
&& RDE(iteration-1) < 2.0
&& RDE(iteration-1) > size_res)

(confer Section 4.5)

3. $j = 0, \ldots,$iteration-1, $i = 0, \ldots,$ gridpointsDH-1:
Appr(i) = Appr(i) + marks(j,1) · Val_d_Appr(j,i)
4. RAE

$$= \frac{\sqrt{\frac{\sum_{i=1}^{\texttt{gridpointsDH}} (\texttt{Appr}(i) - \texttt{solution}(i))^2}{\texttt{gridpointsDH}}}}{\texttt{norm\_sol}}$$

5. save Appr, marks, RDE, TikhFun, RAE in files

Note that with the ROFMP algorithm, in contrast to the RFMP algorithm, we omit computing the approximation in every step of the iteration process as the coefficients $\alpha_n^{(N)}, n = 0, \ldots, N$, are changing in each iteration N. However, to consider the approximation error along the iterations, the computation of $f_N^{(N)}$ in each iteration is necessary. Thus, we abstain from these considerations due to efficiency reasons.

B.3. Implementing an LIPMP algorithm

Before we run one of the LIPMP algorithms, we declare some variables in addition to, in particular, the ones from the IPMP algorithms. First, we consider those used for an efficient preprocessing of the learnt dictionary.

- `matrixDlearnt`
  In the i-th column, the values $\mathcal{T}_{\neg} d_i$ for the i-th chosen dictionary element d_i are stored. As this matrix is for the use in an IPMP algorithm, in the sequel, we will use `matrixDlearnt` in order to distinguish between the similar matrix of the starting dictionary and of the learnt dictionary.
- `fnsh_at_cen_cho`
  Stores the fully normalized spherical harmonics (2.14) evaluated at the centres of chosen Abel–Poisson low or band pass filters. Initially, the matrix is zero.
- `idx_local_sol`
  Assume the starting dictionary is ordered by $i = 1, ...,$ `num_dic_ele`. Then `idx_local_sol` stores the number i of chosen dictionary elements from the starting dictionary. Initially, it is set to zero. Note that this vector is important in particular with respect to chosen spherical harmonics.
- `size_learnt_dictionary`
  Gives the size of the learnt dictionary after the LIPMP algorithm terminated. Thus, it replaces `num_dic_ele` in future runs of an IPMP algorithm with the learnt dictionary. Initially, it is set to zero.

Further, in an LIPMP algorithm itself, we use the following variables.

- `max_deg_cho`
  Saves the currently maximal chosen degree of spherical harmonics, i. e. the learnt degree ν_0 (see (7.12)). Initially, we set it to zero.
- `max_val`
  Gives the value of the objective function $\mathrm{IPMP}(\cdot; N)$ (see (7.103)) in an optimization process.
- `x_wk, x_sl, grad_wk, grad_sl`
  The vectors `x_wk` and `x_sl` store the values of the iterates for the optimization problems. The vectors `grad_wk` and `grad_sl` analogously store the values of the gradients of the objective function $\mathrm{IPMP}(\cdot; N)$ (see (7.103)). The values `x_wk` and `grad_wk` are related to the Abel–Poisson low and band pass filters. The values `x_sl` and `grad_sl` are related to the Slepian functions.
- `candidate_values`
  List of candidates of the current iteration. In particular, we use the indices s, g and l originating from Algorithm 5 and denoting whether it is a starting, global or local solution. Then, with x or z as the vector of characteristics of

the (locally or globally) optimized solution, we save the following:

IPMP $\left(d_{N+1}^{\mathrm{SH}}; N\right)$		
IPMP $\left(d_{N+1}^{(\mathrm{s,APK})}(x); N\right)$	IPMP $\left(d_{N+1}^{(\mathrm{g,APK})}(x); N\right)$	x
IPMP $\left(d_{N+1}^{(\mathrm{s,APW})}(x); N\right)$	IPMP $\left(d_{N+1}^{(\mathrm{g,APW})}(x); N\right)$	x
IPMP $\left(d_{N+1}^{(\mathrm{s,SL})}(z); N\right)$	IPMP $\left(d_{N+1}^{(\mathrm{g,SL})}(z); N\right)$	z
	IPMP $\left(d_{N+1}^{(\mathrm{l,APK})}(x); N\right)$	x
	IPMP $\left(d_{N+1}^{(\mathrm{l,APW})}(x); N\right)$	x
	IPMP $\left(d_{N+1}^{(\mathrm{l,SL})}(z); N\right)$	z

- `fnsh_tmp, fnsh_at_cen_ker_cho, fnsh_at_cen_wav_cho`
  Give the values of the fully normalized spherical harmonics (2.14) evaluated at the centres of chosen Abel–Poisson low (`fnsh_at_cen_ker_cho`) as well as band (`fnsh_at_cen_wav_cho`) pass filters. The vector `fnsh_tmp` is temporarily used for both cases. Initially, they are all set to zero.
- `TK_at_eta, TW_at_eta, Tg_at_eta, Tg`
  Gives the image of an Abel–Poisson low (`TK_at_eta`) and band (`TW_at_eta`) pass filter as well as a set of Slepian functions (`Tg`) and $\mathcal{T}_{\neg}$. The respective values of one Slepian function is given in `Tg_at_eta`. Initially, they are all set to zero.
- `ev_sleps`
  Gives the set of Slepian Fourier coefficients of the current spherical cap in the optimization process together with its characteristics. See also the function `sleps(·)`. Initially, it is set to zero.

Moreover, particularly for the LROFMP algorithm, we need

- `beta_cur, beta_g`
  The vector `beta_cur` gives the values of the projection coefficient of an optimized trial function. The matrix `beta_g` stores the vectors `beta_cur` for a whole set of Slepian functions. Initially, it is set to zero.
- `PWnTd, PWnTg`
  In the optimization process, it gives the image of the current iterate under $\mathcal{T}_{\neg}$ projected onto the orthogonal complement of the previously chosen basis elements. Initially, it is set to zero.
- Bsaved, Dsaved
  To enable an efficient computation of arbitrary projection coefficients in the

optimization process, it is convenient to store certain values along the iterations. In `Dsaved`, we save the column i of `matrixD` of the N-th iteration if d_i is chosen in this step. Further, `Bsaved` saves the respective column of `matrixB`.

- `slepian_centres_cho, slepian_centre_cur`
  For a chosen Slepian functions, we store the centre of the localization region in addition to the Euler angles (see (7.98)) in `slepian_centres_cho`. The centre of the localization region of a current candidate is stored in `slepian_centres_cur`.

B.3.1. Declaring the optimization problems

In our summary of an implementation of an LIPMP algorithm, we will step into the optimization process by calls of the type `nlopt_optimize((local_)optname, z, max_val)` where `z` stands for `x_wk` and `x_sl`, respectively, as described in Johnson (2019). Note that we use `optname` as a general handler of the optimization problems. In particular, we use `opt_ker` (low pass filters), `opt_wav` (band pass filters) and `opt_sleps` (Slepian functions) for the global optimizations as well as `local_opt_ker`, `local_opt_wav` and `local_opt_sleps` for the local problems.
Before we consider the workflow of the LIPMP algorithm, we explain our definition of a optimization problem. Based on Johnson (2019), we declare an optimization problem as

```
nlopt_opt opt_name = nlopt_create(opt_routine,3)
nlopt_set_lower_bounds(opt_name,lb)
nlopt_set_upper_bounds(opt_name,ub)
nlopt_set_max_objective(opt_name, ipmp_name, data_obj_fun_z[0])
nlopt_add_inequality_constraint(
     opt_name, constraint, NULL, feasibility)
nlopt_set_ftol_abs(opt_name, abs_tol_f)
nlopt_set_xtol_abs1(opt_name, abs_tol_x)
nlopt_set_maxeval(opt_name, maxeval)
nlopt_set_maxtime(opt_name, maxtime).
```

The function `nlopt_create` declares an instance of an optimization problem using the algorithm specified by `opt_routine`. Next, we set the values of the lower and upper bounds of the optimization problem. These bounds are given by `lb` and `ub` (see below). Note that most optimization routines need proper values here even though, in theory, we could do with less boundaries. Then we pass the objective function `ipmp_name` (i. e. `ipmp_ker`, `ipmp_wav` or `ipmp_sleps`) using a certain structured object `data_obj_fun_z` (see below). In particular, by using `nlopt_set_max_objective`, we ensure that the algorithm seeks the maximal value. Hence, we do not have to take care of adjusting the sign of the objective function by ourselves. In the case of the Abel–Poisson low and band pass filters, we define the inequality constraint (see below) by `nlopt_add_inequality_constraint`. The argument `feasibility` sets the size of a neighbourhood in which the con-

straint condition is softened or narrowed. That is, if the theoretical constraint is given by $h(x) \leq 0$, then we consider $h(x) \leq$ `feasibility` in practice. In the case of the Slepian functions, the function call `nlopt_add_inequality_constraint` is omitted. At last, we set the termination criteria `abs_tol_f, abs_tol_x, maxeval` and `maxtime` as described before.

Additional structure We explain `lb` and `ub` as well as `data_obj_fun_z`. The vectors `lb` and `ub` store the lower and upper bounds of the optimization problems, respectively. In particular, for bounds with respect to the Abel–Poisson low and band pass filters, we have

```
lb[3] = { -1, -1, -1 } and ub[3] = { 1, 1, 1 }
```

as this is the smallest cube that contains the unit ball (confer (7.15) to (7.18)). Note that these values are mathematically irrelevant, but the NLopt library demands proper bound values for global optimization routines. With respect to the Slepian functions, we have

```
lb[4] = { 0, 0, 0, 0 } and
ub[4] = { M_PI, 2.0 · M_PI, M_PI, 2.0 · M_PI }
```

(confer (7.13) and (7.14)). The objective function IPMP$(d; N)$ depends on certain other values than only the dictionary element d such as the current residual R^N. These arguments are passed to the implemented objective function `ipmp_name` via a so-called structured object `data_obj_fun_z`. In particular, for the Abel–Poisson low and band pass filters, the objects `data_obj_fun_k` and `data_obj_fun_w` of the type `add_data_wk` contain

```
typedef struct{
   l, iteration, sobolev_eval, max_degree_m, bandlimit, mu,
     num_dic_ele
   lambda, rel_orbit, splinesize, alpha
   res, sobolevseq, beta_cur, PWnTd, Td
   marks, eta, matrixDlearnt, Dsaved, Bsaved, matrixS,
     fnsh_tmp
} add_data_wk
```

with `Td = TK_at_eta` or `Td = TW_at_eta`. Similarly, for the Slepian functions, we use an object `data_obj_fun_sl` of the type `add_data_sl` defined by

```
typedef struct{
   l, iteration, bandlimit, idx_slep, mu, num_dic_ele
   lambda, rel_orbit, splinesize, alpha
   res, sobolevseq, slepian_centre_cur
```

```
    beta, PWnTd, Tg, evsleps, marks, fnsh_at_data,
      fnsh_at_cen_ker_cho, fnsh_at_cen_wav_cho, matrixDlearnt,
      Dsaved, Bsaved, matrixS, slepian_centres_cho,
      wignercoeffs_a, wignercoeffs_b, wignercoeffs_c, wignercoeffs_d
} add_data_sl.
```

Note that with the assignment

```
data_obj_fun_xx-><struct argument> = <variable>,        (B.3)
```

each object is updated in order to contain the current values in each iteration of an LIPMP algorithm.

The objective functions We introduce the implementation of the objective function `ipmp_name`, that is the functions `rfmp_name` and `rofmp_name`, respectively. We do this simultaneously, but highlight the additional computation necessary for `rofmp_name`.

For the filters For the Abel–Poisson low and band pass filters, the objective functions are built as follows.

1. read data from `data_obj_fun_z`
2. `if rofmp_name:`
   compute `spline` as the product given in (7.100)
   execute the following computations `3.` to `10.` only if `spline` $\neq 0$
3. compute `b2` via (3.23) and (3.26), respectively, using `sobolev_seq`
4. $i = 0, ...,$ `l-1:`
   compute `x_eta = x` $\cdot$ `eta(i,:)`
   `Td_at_eta` via (7.41) and (7.42), respectively, using `rel_orbit` and `eta`
   `DTd_at_eta` via (7.44) and (7.46), respectively, using `rel_orbit` and `eta`
   `if rofmp_name:`
   `beta_cur` via Theorem 4.4.3, respectively, using `Td_at_eta, Dsaved` and `Bsaved`
   `Dbeta_cur` via (7.39) and (7.40), respectively, using `DTd_at_eta, Dsaved` and `Bsaved`
   `PWnTd` via (7.37) using `Td_at_eta, beta_cur` and `matrixD`
   `DPWnTd` via (7.38) using `DTd_at_eta, Dbeta_cur` and `matrixD`

```
5. if rfmp_name:
      compute a1                                   via (7.47) or (7.55)
                                                 using Td_at_eta and data
               b1                                  via (7.51) or (7.59)
                                                     using Td_at_eta
   if rofmp_name:
      compute a3                                   via (7.49) or (7.57)
                                                 using PWnTd and data
               b3                                  via (7.53) or (7.61)
                                                     using PWnTd
6. compute fnsh_tmp          as the evaluation of (2.14) at the current iterate
                                                     using fnsh(·)
7. if rofmp_name
      compute a4, b5                                   via (7.73)
                                          using marks, beta_cur and matrixS
8. i = mu,..., iteration-1:
      compute a2                          via (7.74) and (7.75), respectively,
                                          using marks, sobolev_seq, fnsh_tmp
                                                 and the Clenshaw algorithm
      if rofmp_name:
               b4                         via (7.74) and (7.75), respectively,
                                 using beta_cur, marks, sobolev_seq, fnsh_tmp
                                                 and the Clenshaw algorithm
9. set in data_obj_fun_z:
      value (= max_val)                   via (7.7) and (7.9), respectively,
      alpha                               as in Algorithm 3 and Algorithm 4,
      Td, fnsh_tmp
      if rofmp_name:  beta_cur, PWnTd
10. if grad:
      execute aLfDerivPlus(·)                    implements Algorithm 15
                                                 using Algorithm 14
               fnshQUOT(·)      implements (2.14) but with the Algorithm 14
                                                 instead of e. g. Algorithm 9
      j = 1,2,3
         compute gradb2                          via (7.96) and (7.97), respectively,
                                                 using sobolev_seq
                  if rfmp_name:
                     grada1                      via (7.48) or (7.56)
                                                 using DTd_at_eta and data
                     gradb1                      via (7.52) or (7.60)
                                                     using DTd_at_eta
```

`if rofmp_name:`
`grada3` via (7.50) or (7.58)
using `DPWnTd` and `data`
`gradb3` via(7.54) or (7.62)
using `DPWnTd`
`grada4, gradb5` via (7.78)
using `marks, beta_cur, Dbeta_cur`
and `matrixS`
`nabla_x_SH` via (7.82)
using `fnsh_tmp` and the results from
`aLfDerivPlus(·)` and `fnshQUOT(·)`
`nabla_x_Pn` via (7.89)
using `sobolev_eval` and `marks`
as well as implementations of (A.1) and (A.2)
`grada2` via (7.92) and (7.93), respectively,
using `marks, sobolev_seq,`
`nabla_h_SH` and `nabla_h_Pn`
`if rofmp_name:`
`gradb4` via (7.92) and (7.93), respectively,
using `beta_cur, Dbeta_cur, matrixS, marks,`
`sobolev_seq, nabla_h_SH` and `nabla_h_Pn`
`splinegrad` via (7.102)
set `grad[`j`]` via (7.20) as well as (7.21) and (7.101), respectively

For the Slepian functions Similarly, we have for the function `ipmp_sl`

1. read data from `data_obj_fun_sl`
2. compute `slepsgrad(·)` implements a similar routine as `sleps`
but additionally with Theorem 3.1.13, (7.33), (7.34), (7.35), (7.36)
and the rotation of the gradient vectors via (7.25), (7.26), (7.27),(7.28), (7.29), (7.30), (7.31);
it stores the coefficients in `evsleps`
and the gradients in `dz_evsleps` for `dz = dc, da, db, dg`
3. `if rofmp_sl:`
compute `spline` as the product given in (7.100) and `splinegrad` via (7.102)
execute the following computations only if `spline` $\neq 0$
4. $m = 0, ...,$ `bandlimit`, $k = -m, ..., m$
a) compute `b2` via (3.28)
using `sobolev_seq` and `evsleps`
`gradb2` via (7.95)
using `sobolev_seq` and `dz_evsleps`

b) if rfmp_sl:
  compute a2 via (7.76)
  using marks, sobolev_seq, evsleps,
  fnsh_at_cen_ker and fnsh_at_cen_wav
  grada2 via (7.94)
  using marks, sobolev_seq, dz_evsleps,
  fnsh_at_cen_ker and fnsh_at_cen_wav

c) $i = 0, ...,$ l-1:
  compute Tg(:, $m^2 + m + k$) via (3.18)
  using evsleps, rel_orbit and fnsh_at_data
  DTg(:, $m^2 + m + k$) via (7.22)
  using dz_evsleps, rel_orbit and fnsh_at_data
  if rofmp_sl:
    compute beta_cur via Theorem 4.4.3, respectively,
    using Tg, Dsaved and Bsaved
    Dbeta_cur via (7.39) and (7.40), respectively,
    using DTg, Dsaved and Bsaved
    PWnTd via (7.37)
    using Tg, beta_cur and matrixD
    DPWnTd via (7.38)
    using DTg, Dbeta_cur and matrixD

d) if rfmp_sl:
  compute a1 via (7.63)
  using Tg and data
  b1 via (7.67)
  using Tg
  grada1 via (7.64)
  using DTg and data
  gradb1 via (7.68)
  using DTg
  if rofmp_sl:
  compute a3 via (7.65)
  using PWnTd and data
  b3 via(7.69)
  using PWnTd
  a4, b5 via (7.73)
  using marks, beta_cur and matrixS
  a2, b4, via (7.76)
  using marks, beta_cur sobolev_seq, evsleps,
  fnsh_at_cen_ker and fnsh_at_cen_wav
  grada3 via (7.66)
  using DPWnTd and data

gradb3 via (7.70)
using DPWnTd
grada4, gradb5 via (7.78)
using marks, beta_cur, Dbeta_cur and matrixS
grada2, gradb4 via (7.94)
using marks, sobolev_seq, evsleps, dz_evsleps,
fnsh_at_cen_ker and fnsh_at_cen_wav

e) set objective_value_sleps(m^2+m+k)
via (7.7) and (7.9), respectively,
alpha_sleps(m^2+m+k) as α in Algorithm 3 and Algorithm 4,
gradgk(m^2+m+k,:)
via (7.20) as well as (7.21) and (7.101), respectively

5. idx=max_index(objective_value_sleps)
(confer (Galassi et al., 2019))

6. set in data_obj_fun_sl:
value = objective_value_sleps(idx),
alpha = alpha_sleps(idx),
Tg, evsleps
if rofmp_sl: beta_cur, PWnTd, slepian_centre_cur

Note that, in 4., we compute the value of the objective function ipmp_sl for all Slepian functions of the current localization cap. These values are stored in the vector objective_value_sleps. In 5., we determine the particular function which produces the maximal value in objective_value_sleps.

The constraint In the case of the Abel–Poisson low and band pass filters, we need to implement the inequality constraint $x \in \mathring{\mathbb{B}}$, see (7.15) to (7.18). This is straightforward done by

```
constraint(n, x, grad, data){
   if(grad){
     grad[0] = 2.0 · x[0]
     grad[1] = 2.0 · x[1]
     grad[2] = 2.0 · x[2]
   }    return (x[0] · x[0] + x[1] · x[1] + x[2] · x[2] - 1.0)
}
```

Note that we adhere to the definition of a constraint as given in Johnson (2019): in the NLopt library, an inequality constraint $h(x)$ is implemented such that it holds $h(x) \leq 0$. The function depends on the dimension of the optimization problem n. In our case, we have $n = 3$. Furthermore, the general definition enables the use of a structured object for additional input (as we used it with the objective functions). In our case, this is not necessary and we pass a NULL pointer

to it when calling `nlopt_add_inequality_constraint`. Obviously, we need the current point `x` $\in \mathring{\mathbb{B}}$ and a memory for the gradient values `grad`. If the gradient values are needed in the chosen optimization routine, we set them accordingly to $\nabla \|x\|^2_{\mathbb{R}^3} = (2x_1, 2x_2, 2x_3)^T$. At last, we return the value $\|x\|^2_{\mathbb{R}^3} - 1$.

B.3.2. The iterations of the LIPMP algorithms

We explain the structure of the functions

```
lipmp(  sobolevseq, sobolev_eval, l, data, norm_data, LookUp,
        num_dic_ele, sizeLookUp, num_sh, max_degree_m,
        num_scales_ker, num_cen_ker, num_scales_wav, num_cen_wav,
        full_num_sl, matrixD, matrixDTD, matrixS, regpar,
        iterations, size_res, ortho_it, smoothing,
        smooth_sh, smooth_shsl, do_sh, do_apk, do_apw, do_sl,


  gridpointsDH, sol, solution, norm_sol, fnsh_at_sol,
  opt_routine, local_opt_routine, opt_routine_sleps,
  local_opt_routine_sleps, feasibility, abs_tol_f, abs_tol_x,
  maxeval, maxtime, rel_orbit, eta, bandlimit, num_sl,
  fnsh_at_data, wignercoeffs_a, wignercoeffs_b,
  wignercoeffs_c, wignercoeffs_d, xi, zeta, fnsh_at_cen_ker,
  fnsh_at_cen_wav, marks, matrixDlearnt,
  size_learnt_dictionary, fnsh_at_cen_cho, idx_local_sol )
```

We introduce the structure for both algorithms simultaneously and mark those parts that are only needed in the LROFMP algorithm. Note that we abstain from describing the smoothing mechanism (Remark 7.3.8) in detail (see `1b`). This is done in order to avoid repetition. If `smoothing = 1`, then we choose only from spherical harmonics and / or Slepian functions in the first iterations. Hence, in this case, the remaining steps of `1.` are executed but only with respect to the fewer trial function classes. The interested reader will be able to transfer the steps.

`1. do`

`a) lambda = lambda0/iteration`
(confer Remark 7.3.7)

`b) if (smoothing) ... else`
(confer Remark 7.3.8)

`c) if LROFMP`
compute `Sik` as explained in the ROFMP algorithm
(Appendix B.2.2, step `1c`)

`d) for xxx` $\in$ `{ SH, SL, APK, APW }`
`if LRFMP`

compute step 1c of the RFMP algorithm (Appendix B.2.1)
and save the values of the trial function class xxx
in objective_function_xxx
if LROFMP
compute step 1d of the ROFMP algorithm (Appendix B.2.2)
and save the values of the trial function class xxx
in objective_function_xxx
ν_{xxx} = max_index(objective_function_xxx)
(confer Galassi et al. (2019))
candidate_values(LookUp(ν_{xxx},0)-1,0)
= objective_function_xxx(ν_{xxx})
e) for i = 2,3,4
switch(i)
case 2:
set data_obj_fun_k via (B.3)
x_wk via (2.1)
using LookUp(ν_{APK},2:3)
compute
check_opt = nlopt_optimize(optker, x_wk, max_val)
tmp = $\sum_{i=0}^{2}$ x_wk(i)^2
if (check_opt > 0 && tmp < 1)
candidate_values(1,1) = max_val
candidate_values(1,2:4) = x_wk
else
candidate_values(1,1:4) = 0
compute
check_opt
= nlopt_optimize(local_optker, x_wk, max_val)
tmp = $\sum_{i=0}^{2}$ x_wk(i)^2
if (check_opt > 0 && tmp < 1)
candidate_values(4,1) = max_val
candidate_values(4,2:4) = x_wk
else
candidate_values(4,1:4) = 0

case 3:
Same as case 2, but with (i.o. = instead of)

data_obj_fun_w	i.o.	data_obj_fun_k
LookUp(ν_{APW},2:3)	i.o.	LookUp(ν_{APK},2:3)
(local_)optwav	i.o.	(local_)optker
candidate_values(2,:)	i.o.	candidate_values(1,:)
candidate_values(5,:)	i.o.	candidate_values(4,:)

```
          case 4:
            set data_obj_fun_sl                                  via (B.3)
                x_sl                                  via LookUp(ν_SL,1:4)
            compute
                 check_opt
                   = nlopt_optimize(optsleps, x_sl, max_val)
            if (check_opt > 0 && x_sl feasible)
              candidate_values(3,1) = max_val
              candidate_values(3,2:5) = x_sl
            else
              candidate_values(3,1:5) = 0
            compute
                 check_opt
                   = nlopt_optimize(local_optsleps, x_sl, max_val)
            if (check_opt > 0 && x_sl feasible)
              candidate_values(6,1) = max_val
              candidate_values(6,2:5) = x_sl
            else
              candidate_values(6,1:5) = 0
f) (imax, jmax) = max_index(candidate_values)
g) if (jmax = 0)
       processing a chosen candidate, case 1, Appendix B.3.3
   else
       idx_local_sol(iteration-1,-1)
       if (imax = 1 | | imax == 4)
         processing a chosen candidate, case 2, Appendix B.3.3
       if (imax = 2 | | imax == 5)
         processing a chosen candidate, case 2, Appendix B.3.3
       if (imax = 3 | | imax == 6)
         processing a chosen candidate, case 3, Appendix B.3.3

h) if LRFMP
     possibly compute step 1k of the RFMP algorithm (see Appendix B.2.1)
   and additionally set
         mu = iteration+1
   if LROFMP
     compute step 1n of the ROFMP algorithm (see Appendix B.2.2) and
   possibly additionally set
         Bsaved(:,:)  = 0
         Dsaved(:,:)  = 0
i) iteration = iteration+1
```

2. while(iteration < iteration_number+1
   && RDE(iteration-1) < 2.0
   && RDE(iteration-1) > size_res)

3. if LROFMP
   $j = 0, \ldots,$ iteration-1
   $i = 0, \ldots,$ gridpointsDH-1:
   Appr(i) = Appr(i) + marks(j-1,1) · Val_d_Appr(j-1,i)

$$\texttt{RAE} = \frac{\sqrt{\frac{\sum_{i=0}^{\texttt{gridpointsDH-1}} (\texttt{Appr}(i) - \texttt{solution}(i))^2}{\texttt{gridpointsDH}}}}{\texttt{norm\_sol}}$$

4. size_learnt_dictionary = iteration-1
5. save Appr, marks, RDE, TikhFun, RAE in files

B.3.3. Processing a chosen candidate

There are 3 necessary and different routines for processing a chosen candidate. They are distinguished in: a candidate from the starting dictionary (in particular, a spherical harmonic) is chosen, an optimized Abel–Poisson low or band pass filter is chosen or an optimized Slepian function is chosen. In general, the processing of a chosen candidate is similar as it was described in Telschow (2014). However, note that, if we choose an optimized trial function d, certain $\mathcal{H}_2(\Omega)$-inner products need to be computed from scratch, e. g. for the updated term $\langle f_N, d\rangle_{\mathcal{H}_2(\Omega)}$. Further, we additionally include aspects that process the learnt dictionary for future runs of an IPMP algorithm.
Note that, in the sequel, if we compare steps to the RFMP or the ROFMP algorithm, we refer to Appendix B.2.1 or Appendix B.2.2, respectively.

Choose a starting solution The steps are very similar to the RFMP and ROFMP algorithm.

1. set $\nu = \nu_{\texttt{xxx}}$
   marks(iteration-1,:)
   as in 1e of the RFMP algorithm
   or 1f of the ROFMP algorithm
2. if (marks(iteration-1, 0) = 2 or 3):
   set the column iteration-1 of
   fnsh_at_cen_ker_cho and fnsh_at_cen_cho
   using fnsh_at_cen_ker(:,marks(iteration-1,5))
   or fnsh_at_cen_wav_cho and fnsh_at_cen_cho
   using fnsh_at_cen_wav(:,marks(iteration-1,5))
3. idx_local_sol(iteration-1) = ν

4. if (marks(iteration-1, 0) = 1)
   max_deg_cho = marks(iteration-1, 2)
5. if LRFMP:
   a) store the ν-th column of matrixD of the chosen dictionary element in the column iteration-1 of matrixDlearnt
   b) compute 1f of the RFMP algorithm
   c) compute 1g to 1j of the RFMP algorithm

   if LROFMP:
   a) if (marks(iteration-1, 0) = 4):
      compute slepian_centres_cho(iteration-1,:)
   b) compute 1i of the ROFMP algorithm
   c) store the ν-th column of
      matrixDfresh in the column iteration-1 of matrixDlearnt
      matrixD in the column iteration-1 of Dsaved
   d) update coefficients of the approximation as in 1j of the ROFMP algorithm
   e) update QIV:
      $i = 0, \ldots,$ iteration-1:
      matrixS(num_dic_ele+i,num_dic_ele+iteration-1)

$$= \begin{cases} \text{matrixS(idx\_local\_sol}(i),\nu), & \text{idx\_local\_sol}(i) >= 0, \\ & i < \text{iteration-1} \\ \text{matrixS}(\nu,\text{num\_dic\_ele}+i)), & \text{idx\_local\_sol}(i) < 0, \\ & i < \text{iteration-1} \\ \text{matrixS}(\nu,\nu), & i = \text{iteration-1} \end{cases}$$

      and similarly below the main diagonal
   f) compute 1k to 1m of the ROFMP algorithm

Choose an optimized filter In the second case, we execute the following steps.

1. set x_wk = candidate_values(imax,2:4)
   and run ipmp_name(3, x_wk, grad_wk, data_obj_fun_z) to ensure that all values in data_obj_fun_z are related to the chosen candidate
2. read from data_obj_fun_z: Td_at_eta, fnsh_tmp, alpha
   if LROFMP: also read beta_cur, PWnTd
3. set the column iteration-1 of
   fnsh_at_cen_ker_cho and fnsh_at_cen_cho using fnsh_tmp
   or fnsh_at_cen_wav_cho and fnsh_at_cen_cho using fnsh_tmp
4. set marks(iteration-1,:) as in 1e of the RFMP algorithm
   or 1f of the ROFMP algorithm
5. if LRFMP:
   a) store Td_at_eta in the column iteration-1 of matrixDlearnt
   b) compute 1f of the RFMP algorithm
   c) update IP_data_Tupd as in 1g of the RFMP algorithm

but using `Td_at_eta` and `matrixD` instead of `matrixDTD`

d) compute $\langle d_{N+1}, d_i \rangle_{\mathcal{H}_2(\Omega)}$ for all dictionary elements $d_i \in \mathcal{D}^{\mathrm{s}}$ analogously to `computeS(·)` and update `IP_appr_d` and `matrixS` (i. e. QII and QIII)

e) compute `1h` to `1j` of the RFMP algorithm

`if LROFMP:`

a) store
`Td_at_eta` in the column `iteration-1` of `matrixDlearnt`
`PWnTd` in the column `iteration-1` of `Dsaved`
`beta_cur` in the column `iteration-1` and rows `mu` to `iteration-1` of `Bsaved`

b) compute `1i` of the ROFMP algorithm

c) update coefficients of the approximation as in `1j` of the ROFMP algorithm

d) compute $\langle d_{N+1}, d_i \rangle_{\mathcal{H}_2(\Omega)}$ analogously to `computeS(·)`
   i. for all dictionary elements $d_i \in \mathcal{D}^{\mathrm{s}}$ and update `IP_appr_d`, QII, QIII and QIV of `matrixS`
   ii. and for all previously chosen basis elements $d_i \in \mathcal{D}^{\mathrm{Inf}}$ and update QIV of `matrixS`

e) compute `1k` to `1m` of the ROFMP algorithm

Choose an optimized Slepian function At last, a Slepian function is processed in the following way.

1. set `x_sl = candidate_values(imax,2:5)` and run `ipmp_sleps(3, x_sl, grad_sl, data_obj_fun_sl)` to ensure that all values in `data_obj_fun_sl` are related to the chosen candidate
2. read from `data_obj_fun_sl`: `idx, Tg, evsleps, alpha`
   `if LROFMP:` `beta_g, PWnTg, slepian_centre_cur`
3. choose `beta_cur` from `beta`, `Tg` from `Tg_at_eta` and `PWnTd` from `PWnTg`
4. set `marks(iteration-1,:)` as in `1e` of the RFMP algorithm or `1f` of the ROFMP algorithm
5. if `LRFMP:`
   a) store `Tg` in the column `iteration-1` of `matrixDlearnt`
   b) compute `1f` of the RFMP algorithm
   c) update `IP_data_Tupd` as in `1g` of the RFMP algorithm but using `Tg` and `matrixD` instead of `matrixDTD`
   d) compute $\langle d_{N+1}, d_i \rangle_{\mathcal{H}_2(\Omega)}$ for all dictionary elements $d_i \in \mathcal{D}^{\mathrm{s}}$ analogously to `computeS(·)` and update `IP_appr_d` and `matrixS` (i. e. QII and QIII)
   e) compute `1h` to `1j` of the RFMP algorithm

   if `LROFMP:`
   a) store

Tg in the column iteration-1 of matrixDlearnt
PWnTd in the column iteration-1 of Dsaved
beta_cur in the column iteration-1 and rows mu to iteration-1 of Bsaved

b) compute 1i of the ROFMP algorithm
c) update coefficients of the approximation as in 1j of the ROFMP algorithm
d) compute $\langle d_{N+1}, d_i \rangle_{\mathcal{H}_2(\Omega)}$ analogously to computeS(·)
   i. for all dictionary elements $d_i \in \mathcal{D}^{\mathrm{s}}$ and update IP_appr_d, QII, QIII and QIV of matrixS
   ii. and for all previously chosen basis elements $d_i \in \mathcal{D}^{\mathrm{Inf}}$ and update QIV of matrixS
e) compute 1k to 1m of the ROFMP algorithm

B.3.4. Preprocessing of the learnt dictionary

Recall that the main aspects of the preprocessing introduced in Appendix B.1 and Telschow (2014) are to compute the matrices LookUp, matrixD, matrixDTD and matrixS. However, we constructed the LIPMP algorithms such the we obtain the respective matrixD (= matrixDlearnt) of the learnt dictionary on the fly.

- LookUp
  The matrix LookUp contains the elements of the matrix marks except for the coefficients α_i, $i = 1, ..., \texttt{iteration}$. That means, we copy LookUp(:,1) = marks(:,1) and LookUp(:,2:end) = marks(:,3:end).
- matrixDTD
  With matrixD, we again obtain matrixDTD via $\texttt{matrixD}^{\mathrm{T}} \cdot \texttt{matrixD}$.
- matrixSlearnt
  The computation of matrixSlearnt, i. e. matrixS with respect to the learnt dictionary, depends on whether we run the LRFMP or the LROFMP. In the former case, we use the function computeSlearnt (see below). In the latter case, we can copy the respective elements from the respective matrixS (see QIV) which we update during the LROFMP (see Appendix B.3.3).

At last, we explain the function

```
computeSlearnt(  matrixS, idx_local_sol, sobolevseq,
                 sobolev_eval, size_learnt_dictionary, marks,
                 matrixSlearnt, fnsh_at_cen_cho, num_dic_ele,
                 bandlimit )
```

for matrixS as used and modified in the LRFMP algorithm. The computations in computeSlearnt depend on the value of idx_local_sol(i) and idx_local_sol(j)

of the ith and jth learnt dictionary element. In the case that at least one of them is non-negative, i. e. either d_i or d_j are chosen from the starting dictionary, we read the value of $\langle d_i, d_j \rangle_{\mathcal{H}_2(\Omega)}$ from the updated `matrixS`. If both are negative, i. e. both d_i and d_j are optimized basis elements, we compute the inner product analogously to `computeS` but using `marks(:,0)` instead of `LookUp(:,0)` to distinguish the different cases.
We summarize this appendix in Figure B.1 and Figure B.2. The former one summarizes major functions. We start in the red box with the declaration of parameters in the file `exec.c`. Then we step into the function `matchingpursuit` which contains the preprocessing (boxes `precomputation(·)` to `computeS(·)`) as well as calls to the various (L)IPMP algorithms and the preprocessing of the learnt dictionary (boxes "compute `LookUp` from `marks`" to `computeSlearnt(·)`) such that a direct use in an IPMP algorithm is enabled. Note that, technically, we could also end the process after any step between "`lrfmp(·)` or `lrofmp(·)`" and "`rfmp(·)` or `rofmp(·)`" if we are less interested in the preprocessing of the learnt dictionary. The workflow of an LIPMP algorithm is summarized in more detail in Figure B.2 where the structure of the optimization processes are highlighted in light blue.

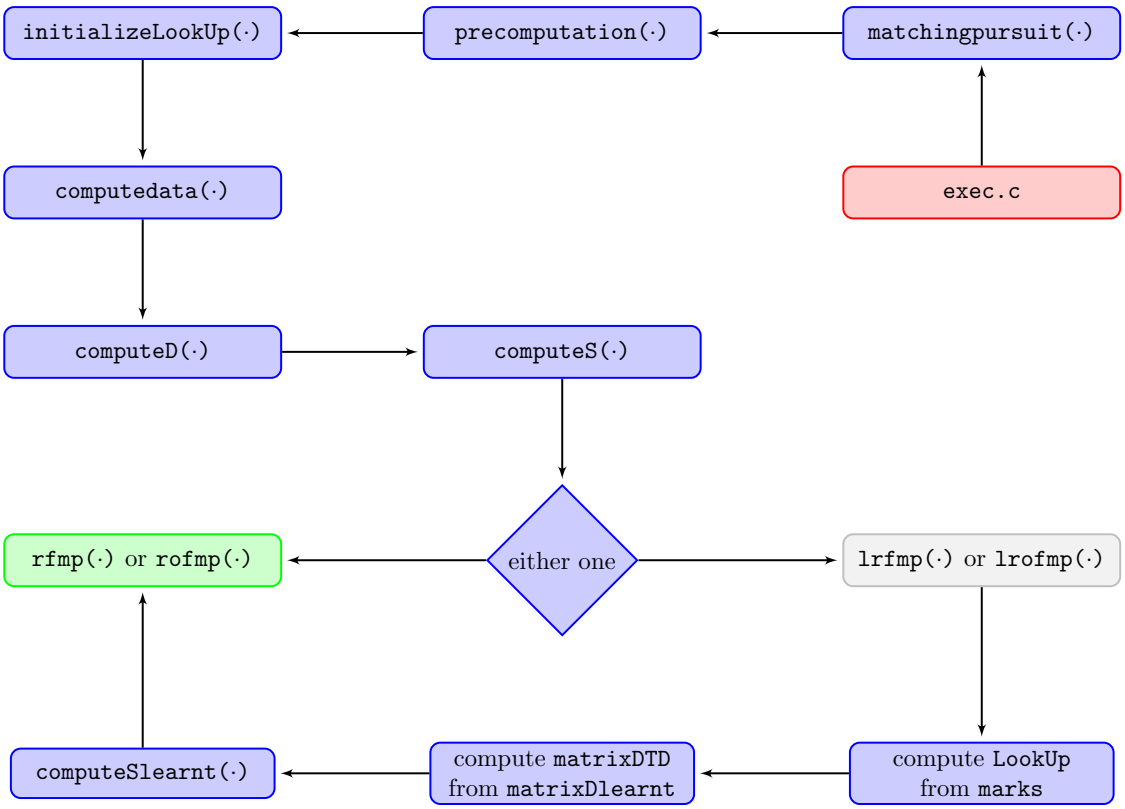


Figure B.1.: Schematic representation of an implementation of the preprocessing and the (L)IPMP algorithms.

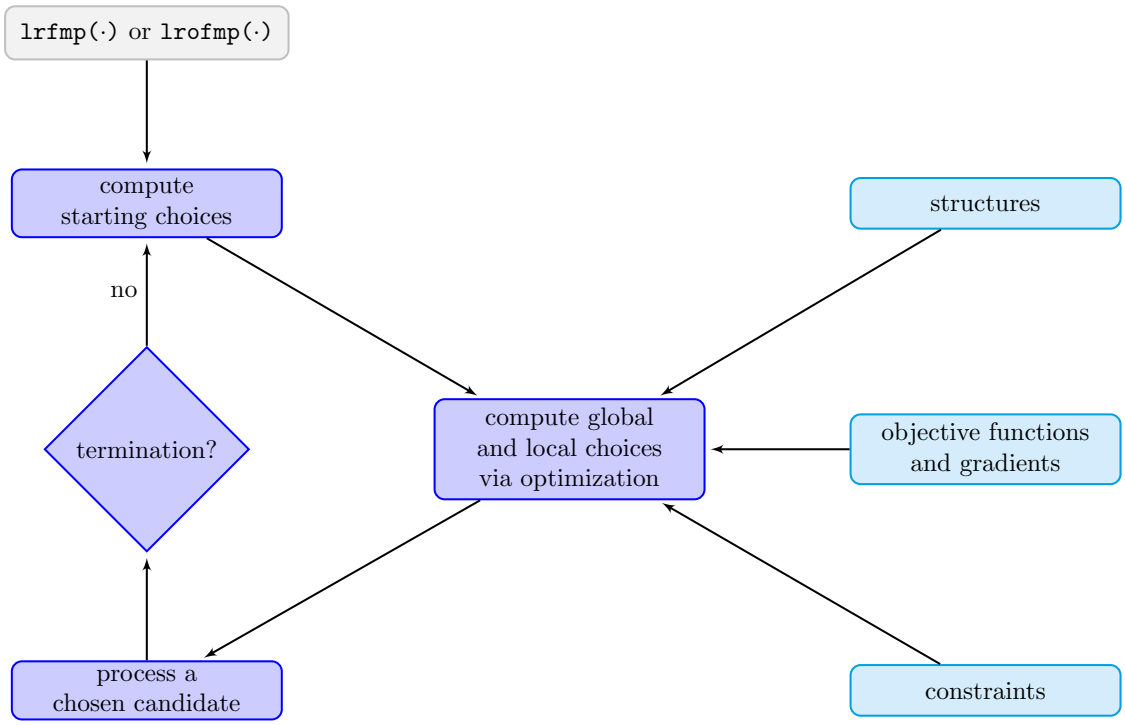


Figure B.2.: Schematic representation of an implementation of an LIPMP algorithm.

Bibliography

Abramowitz, M. and Stegun, I. A. (1972). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, Inc., New York, 10th edition.

Aharon, M., Elad, M., and Bruckstein, A. (2006). K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322.

Akram, M. and Michel, V. (2010). Regularisation of the Helmholtz decomposition and its application to geomagnetic field modelling. *GEM – International Journal on Geomathematics*, 1(1):101–120.

Albertella, A., Sansó, F., and Sneeuw, N. (1999). Band-limited functions on a bounded spherical domain: the Slepian problem on the sphere. *Journal of Geodesy*, 73(9):436–447.

Antoni, M. (2012). *Nichtlineare Optimierung regionaler Gravitationsfeldmodelle aus SST-Daten*. PhD thesis, University of Stuttgart. `https://d-nb.info/1020832185/34`, last accessed 3 March 2020.

Barnes, D., Factor, J. K., Holmes, S. A., Ingalls, S., Presicci, M. R., Beale, J., and Fecher, T. (2015). Earth Gravitational Model 2020. *AGU Fall Meeting Abstracts*, pages G34A–03. Provided by the SAO/NASA Astrophysics Data System.

Barthelmes, F. (1986). Untersuchungen zur Approximation des äußeren Schwerefeldes der Erde durch Punktmassen mit optimierten Positionen. *Veröffentlichung des Zentralinstituts der Physik der Erde Nr. 92, Potsdam*.

Barthelmes, F. (1989). Local gravity field approximation by point masses with optimized positions. *Gravity field variations, Veröffentlichungen des Zentralinstituts der Physik der Erde*, (102):157–167.

Barthelmes, F. (2014). Global models. In Grafarend, E., editor, *Encyclopedia of Geodesy*, pages 1–9. Springer, Cham. Revised version `http://icgem.gfz-potsdam.de/GlobalModelsEncyclopedia.pdf`, last accessed 31 March 2020.

Barthelmes, F., Dietrich, R., and Lehmann, R. (1991). Representation of the global gravity field by point masses on optimized positions based on recent spherical

harmonics expansions. Poster presented at the XX. General Assembly of the International Union of Geodesy and Geophysics, Vienna.

Bauer, F., Gutting, M., and Lukas, M. A. (2015). Evaluation of parameter choice methods for the regularization of ill-posed problems in geomathematics. In Freeden, W., Nashed, M. Z., and Sonar, T., editors, *Handbook of Geomathematics*, pages 1713–1774. Springer, Berlin, Heidelberg, 2nd edition.

Bauer, F. and Lukas, M. A. (2011). Comparing parameter choice methods for regularization of ill-posed problems. *Mathematics and Computers in Simulation*, 81(9):1795–1841.

Baur, O. (2014). Gravity field of planetary bodies. In Grafarend, E., editor, *Encyclopedia of Geodesy*, pages 1–6. Springer International Publishing Switzerland AG, Cham.

Berkel, P., Fischer, D., and Michel, V. (2011). Spline multiresolution and numerical results for joint gravitation and normal mode inversion with an outlook on sparse regularisation. *GEM – International Journal on Geomathematics*, 1(2):167–204.

Bettadpur, S. (2012). GRACE 327-742, UTCSR Level-2 Processing Standards Document (Revd 4.0 May 29, 2012) (For Level-2 Product Release 0005). Technical report, Center for Space Research, The University of Texas, Austin.

Blackett, M. (2014). Early analysis of Landsat-8 thermal infrared sensor imagery of volcanic activity. *Remote Sensing*, 6(3):2282–2295.

Bröcker, T. and tom Dieck, T. (1985). *Representations of Compact Lie Groups. Graduate Texts in Mathematics vol. 98*. Springer, Berlin.

Bruckstein, A. M., Donoho, D. L., and Elad, M. (2009). From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81.

Casagli, N., Catani, F., Del Ventisette, C., and Luzi, G. (2010). Monitoring, prediction, and early warning using ground-based radar interferometry. *Landslides*, 7(3):291–301.

Chambolle, A. and Lions, P.-L. (1997). Image recovery via total variation minimization and related problems. *Numerische Mathematik*, 76(2):167–188.

Choi, C. H., Ivanic, J., Gordon, M. S., and Ruedenberg, K. (1999). Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion. *Journal of Chemical Physics*, 111(19):8825–8831.

Claessens, S. J., Featherstone, W. E., and Barthelmes, F. (2001). Experiences with point-mass gravity field modelling in the Perth region, Western Australia. *Geomatics Research Australasia*, 75:53–86.

Cucker, F. and Smale, S. (2002). On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1):1–49.

Dahle, C., Flechtner, F., Gruber, C., König, D., König, R., Michalak, G., and Neumayer, K.-H. (2013). GFZ GRACE Level-2 Processing Standards Document for Level-2 Product Release 0005. Technical report, GFZ German Research Centre for Geosciences, Potsdam. Revised edition.

Dahlen, F. and Tromp, J. (1998). *Theoretical Global Seismology*. Princeton University Press, Princeton.

Daubechies, I., Defrise, M., and De Mol, C. (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457.

Davis, J. L., Tamisiea, M. E., Elósegui, P., Mitrovica, J. X., and Hill, E. M. (2008). A statistical filtering approach for Gravity Recovery and Climate Experiment (GRACE) gravity data. *Journal of Geophysical Research, Solid Earth*, 113(B4).

De Mol, C., De Vito, E., and Rosasco, L. (2009). Elastic-net regularization in learning theory. *Journal of Complexity*, 25(2):201–230.

Defrise, M., Vanhove, C., and Liu, X. (2011). An algorithm for total variation regularization in high-dimensional linear problems. *Inverse Problems*, 27(6):065002 (16pp).

Demtröder, W. (2006). *Experimentalphysik 1 – Mechanik und Wärme*. Springer Berlin, 3rd edition.

Deuflhard, P. (1976). On algorithms for the summation of certain special functions. *Computing*, 17(1):37–48.

Devaraju, B. and Sneeuw, N. (2017). The polar form of the spherical harmonic spectrum: implications for filtering GRACE data. *Journal of Geodesy*, 91(12):1475–1489.

DiMatteo, I., Genovese, C. R., and Kass, R. E. (2001). Bayesian curve-fitting with free-knot splines. *Biometrika*, 88(4):1055–1071.

Driscoll, J. R. and Healy, D. M. (1994). Computing Fourier transforms and convolutions on the 2-sphere. *Advances in Applied Mathematics*, 15(2):202–250.

Edmonds, A. R. (1996). *Angular Momentum in Quantum Mechanics*. Princeton University Press, Princeton, 4th edition.

Eicker, A., Schall, J., and Kusche, J. (2014). Regional gravity modelling from spaceborne data: case studies with GOCE. *Geophysical Journal International*, 196(3):1431–1440.

Engan, K., Aase, S. O., and Husøy, J. H. (1999a). Method of optimal directions for frame design. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, number 5, pages 2443–2446.

Engan, K., Rao, B. D., and Kreutz-Delgado, K. (1999b). Frame design using FOCUSS with method of optimal directions (MOD). In *Proceedings of the Norwegian Signal Processing Symposium*, 65–69.

Engl, H. W., Hanke, M., and Neubauer, A. (1996). *Regularization of Inverse Problems*. Mathematics and Its Applications. Kluwer Academic Publishers, Dordrecht.

Fengler, M. J. (2005). *Vector Spherical Harmonic and Vector Wavelet Based Non-Linear Galerkin Schemes for Solving the Incompressible Navier-Stokes Equation on the Sphere*. PhD thesis, University of Kaiserslautern, Geomathematics Group, Shaker-Verlag, Aachen.

Fengler, M. J., Freeden, W., Kohlhaas, A., Michel, V., and Peters, T. (2007). Wavelet modeling of regional and temporal variations of the Earth's gravitational potential observed by GRACE. *Journal of Geodesy*, 81(1):5–15.

Fischer, D. (2011). *Sparse Regularization of a Joint Inversion of Gravitational Data and Normal Mode Anomalies*. PhD thesis, University of Siegen, Geomathematics Group, Verlag Dr. Hut, Munich. `http://dokumentix.ub.uni-siegen.de/opus/volltexte/2012/544/index.html`, last accessed 4 March 2020.

Fischer, D. and Michel, V. (2012). Sparse regularization of inverse gravimetry – case study: spatial and temporal mass variations in South America. *Inverse Problems*, 28(6):065012 (34 pp.).

Fischer, D. and Michel, V. (2013a). Automatic best-basis selection for geophysical tomographic inverse problems. *Geophysical Journal International*, 193(3):1291–1299.

Fischer, D. and Michel, V. (2013b). Inverting GRACE gravity data for local climate effects. *Journal of Geodetic Science*, 3(3):151–162.

Fischer, G. (2010). *Lineare Algebra*. Vieweg+Teubner, Wiesbaden, 17th edition.

Flechtner, F., Morton, P., Watkins, M., and Webb, F. (2014a). Status of the GRACE follow-on mission. In Marti, U., editor, *Gravity, Geoid and Height Systems. International Association of Geodesy Symposia*, volume 141, pages 117–121. Springer, Cham.

Flechtner, F., Sneeuw, N., and Schuh, W.-D., editors (2014b). *Observation of the System Earth from Space – CHAMP, GRACE, GOCE and future missions*. Springer, Berlin.

Fletcher, R. (1987). *Practical Methods of Optimization. 2nd edition*. Wiley, Chichester.

Freeden, W. and Gerhards, C. (2013). *Geomathematically Oriented Potential Theory*. Taylor & Francis Group, Boca Raton.

Freeden, W., Gervens, T., and Schreiner, M. (1998). *Constructive Approximation on the Sphere – with Applications to Geomathematics*. Oxford University Press, Oxford.

Freeden, W. and Gutting, M. (2013). *Special Functions of Mathematical (Geo-)Physics*. Springer, Basel.

Freeden, W. and Michel, V. (2004a). *Multiscale Potential Theory with Applications to Geoscience*. Birkhäuser, Boston.

Freeden, W. and Michel, V. (2004b). Orthogonal zonal, tesseral and sectorial wavelets on the sphere for the analysis of satellite data. *Advances in Computational Mathematics*, 21(1–2):181–217.

Freeden, W., Michel, V., and Nutz, H. (2002). Satellite-to-satellite tracking and satellite gravity gradiometry (advanced techniques for high-resolution geopotential field determination). *Journal of Engineering Mathematics*, 43(1):19–56.

Freeden, W. and Schneider, F. (1998). Wavelet approximations on closed surfaces and their application to boundary-value problems of potential theory. *Mathematical Methods in the Applied Sciences*, 21(2):129–163.

Freeden, W. and Schreiner, M. (1998). Orthogonal and non-orthogonal multiresolution analysis, scale discrete and exact fully discrete wavelet transform on the sphere. *Constructive Approximation*, 14(4):493–515.

Freeden, W. and Schreiner, M. (2009). *Spherical Functions of Mathematical Geosciences – A Scalar, Vectorial, and Tensorial Setup*. Springer, Berlin.

Freeden, W. and Windheuser, U. (1996). Spherical wavelet transform and its discretization. *Advances in Computational Mathematics*, 5(1):51–94.

Friedman, J. H. and Stuetzle, W. (1981). Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823.

Friis-Christensen, E., Lühr, H., and Hulot, G. (2006). Swarm: A constellation to study the Earth's magnetic field. *Earth, Planets and Space*, 58(4):351–358.

Fukushima, T. (2012). Numerical computation of spherical harmonics of arbitrary degree and order by extending exponent of floating point numbers. *Journal of Geodesy*, 86(4):271–285.

Galassi, M., Davies, J., Gough, B., Jungman, G., Alken, P., Booth, M., Rossi, F., and Ulerich, R. (2019). *GNU Scientific Library Reference Manual (3rd Ed.)*. `https://www.gnu.org/software/gsl/`, last accessed 3 March 2020.

Garey, M. R. and Johnson, D. S. (2009). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York.

Geiger, C. and Kanzow, C. (2002). *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer, Berlin.

GFZ Potsdam (2000). Grace payload. `http://op.gfz-potsdam.de/grace/payload/payload.html`, last accessed 2 April 2020.

Ghahramani, Z. (2004). Unsupervised learning. In Bousquet, O., von Luxburg, U., and Rätsch, G., editors, *Advanced Lectures on Machine Learning*. Springer, Berlin.

Glabonsky, J. (1998). An implementation of the DIRECT algorithm. *Technical Report CRSC-TR98-29, North Carolina State University, Center for Research in Scientific Computation*, pages 1–28.

Glabonsky, J. M. and Kelley, C. T. (2001). A locally-biased form of the DIRECT algorithm. *Journal of Global Optimization*, 21(1):27–37.

Grafarend, E. W. and Kühnel, W. (2011). A minimal atlas for the rotation group $SO(3)$. *GEM - International Journal on Geomathematics*, 2(1):113–122.

Grünbaum, F. A., Longhi, L., and Perlstadt, M. (1982). Differential operators commuting with finite convolution integral operators: some non-Abelian examples. *SIAM Journal on Numerical Analysis*, 42(5):941–955.

Gutting, M. (2007). *Fast Multipole Methods for Oblique Derivative Problems*. PhD thesis, University of Kaiserslautern, Geomathematics Group, Shaker Verlag, Aachen.

Gutting, M., Kretz, B., Michel, V., and Telschow, R. (2017). Study on parameter choice methods for the RFMP with respect to downward continuation. *Frontiers in Applied Mathematics and Statistics*, 3. Article 10.

Halliday, D., Resnick, R., and Walker, J. (2007). *Physik. Bachelor Edition*. Wiley-VCH, Weinheim. Fundamental of Physics, extended 6th edition, 2001, John Wiley & Sons.

Hanke-Bourgeois, M. (2009). *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Vieweg + Teuber, Wiesbaden, 3rd edition.

Heuser, H. (2001). *Lehrbuch der Analysis Teil 1*. B. G. Teubner, Wiesbaden, 14th edition.

Heuser, H. (2004). *Lehrbuch der Analysis Teil 2*. B. G. Teubner, Wiesbaden, 13th edition.

Heuser, H. (2006). *Funktionalanalysis*. B. G. Teubner, Wiesbaden, 4th edition.

Hofmann, B. (1999). *Mathematik inverser Probleme*. B. G. Teubner, Stuttgart.

HSL (2018). A collection of Fortran codes for large scale scientific computation. `http://www.hsl.rl.ac.uk/`, last accessed 11 December 2018.

Huber, P. J. (1985). Projection pursuit. *The Annals of Statistics*, 13(2):435–475.

IPCC (2014). *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. IPCC, Geneva, Switzerland.

Jekeli, C. (2005). Spline representation of functions on a sphere for geopotential modeling. Technical Report 475, The Ohio State University, Columbus.

Johnson, S. G. (2019). *The NLopt nonlinear-optimization package*. `http://github.com/stevengj/nlopt` and `https://nlopt.readthedocs.io/en/latest/`, both last accessed 2 April 2020.

Jones, D. R. (2001). DIRECT global optimization algorithm. In Floudas, C. A. and Pardalos, P. M., editors, *Encyclopedia of Optimization*. Springer, Boston.

Jones, D. R., Perttunen, C. D., and Stuckman, B. E. (1993). Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Application*, 79(1):157–181.

Jones, L. K. (1987). On a conjecture of Huber concerning the convergence of projection pursuit regression. *The Annals of Statistics*, 15(2):880–882.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

Kellogg, O. D. (1967). *Foundations of Potential Theory*. Springer, Berlin. Reprint from the first edition of 1929.

Khalid, Z., Kennedy, R. A., and McEwen, J. D. (2016). Slepian spatio-spectral concentration on the ball. *Applied and Computational Harmonic Analysis*, 40(3):470–504.

Kirsch, A. (1996). *An Introduction to the Mathematical Theory of Inverse Problems*. Springer, New York.

Klees, R., Revtova, E. A., Gunter, B. C., Ditmar, P., Oudman, E., Winsemius, H. C., and Savenjie, H. H. G. (2008). The design of an optimal filter for monthly GRACE gravity models. *Geophysical Journal International*, 175(2):417–432.

Königsberger, K. (2004a). *Analysis 1*. Springer, Berlin, 6th edition.

Königsberger, K. (2004b). *Analysis 2*. Springer, Berlin, 5th edition.

Kontak, M. (2018). *Novel Algorithms of Greedy-Type for Probability Density Estimation as well as Linear and Nonlinear Inverse Problems*. PhD thesis, University of Siegen, Geomathematics Group, Verlag Dr. Hut, Munich. `http://dokumentix.ub.uni-siegen.de/opus/volltexte/2018/1316/index.html`, last accessed 4 March 2020.

Kontak, M. and Michel, V. (2018). A greedy algorithm for nonlinear inverse problems with an application to nonlinear inverse gravimetry. *GEM – International Journal on Geomathematics*, 9(2):167–198.

Kontak, M. and Michel, V. (2019). The regularized weak functional matching pursuit for linear inverse problems. *Journal of Inverse and Ill-Posed Problems*, 27(3):317–340.

Kraft, D. (1988). A software package for sequential quadratic programming. Techreport DFVLR-FB 88-28, Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, Institut für Dynamik der Flugsysteme, Oberpfaffenhofen.

Kraft, D. (1994). Algorithm 733: TOMP-Fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software*, 20(3):262–281.

Kusche, J. (2007). Approximate decorrelation and non-isotropic smoothing of time-variable GRACE-type gravity field models. *Journal of Geodesy*, 81(11):733–749.

Kusche, J. (2015). Time-variable gravity field and global deformation of the Earth. In Freeden, W., Nashed, M. Z., and Sonar, T., editors, *Handbook of Geomathematics*, pages 321–338. Springer, Berlin, Heidelberg, 2nd edition.

Lehmann, R. (1993). The method of free-positioned point masses – geoid studies on the Gulf of Bothnia. *Bulletin Géodésique*, 67(31):31–40.

Leweke, S. (2018). *The Inverse Magneto-electroencephalography Problem for the Spherical Multiple-shell Model: Theoretical Investigations and Numerical Aspects*. PhD thesis, University of Siegen, Geomathematics Group. `http://dokumentix.ub.uni-siegen.de/opus/volltexte/2019/1396/`, last accessed 4 March 2020.

Leweke, S., Michel, V., and Schneider, N. (2018a). Vectorial Slepian functions on the ball. *Numerical Functional Analysis and Optimization*, 39(11):1120–1152.

Leweke, S., Michel, V., and Telschow, R. (2018b). On the non-uniqueness of gravitational and magnetic field data inversion (survey article). In Freeden, W. and Nashed, M. Z., editors, *Handbook of Mathematical Geodesy*, pages 883–919. Birkhäuser, Basel.

Li, C., Yin, W., Jiang, H., and Zhang, Y. (2013). An efficient augmented Lagrangian method with applications to total variation minimization. *Computational Optimization and Applications*, 56(3):507–530.

Lin, M. (2016). *Regional gravity field recovery using the point mass method*. PhD thesis, University of Hannover.

Lin, M., Denker, H., and Müller, J. (2014). Regional gravity field modelling using free-positioned point masses. *Studia Geophysica et Geodaetica*, 58(2):207–226.

Lin, Y., Yu, J., Cai, J., Sneeuw, N., and Li, F. (2018). Spatio-temporal analysis of wetland changes using a kernel extreme learning machine approach. *Remote Sensing*, 10(7):1129.

Louis, A. K. (1989). *Inverse und schlecht gestellte Probleme*. Teubner, Stuttgart.

Magnus, W., Oberhettinger, F., and Soni, R. (1966). *Formulas and Theorems for the Special Functions of Mathematical Physics*. Die Grundlehren der Mathematischen Wissenschaften 52. Springer-Verlag, Berlin, 3rd edition.

Mallat, S. G. and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Tansactions on Signal Processing*, 41(12):3397–3415.

Mandea, M. and Dormy, E. (2003). Asymmetric behavior of magnetic dip poles. *Earth, Planets and Space*, 55(3):153–157.

Michel, V. (2005). Regularized wavelet-based multiresolution recovery of the harmonic mass density distribution from data of the Earth's gravitational field at satellite height. *Inverse Problems*, 21(3):997–1025.

Michel, V. (2013). *Lectures on Constructive Approximation – Fourier, Spline, and Wavelet Methods on the Real Line, the Sphere, and the Ball*. Birkhäuser Verlag, New York.

Michel, V. (2015a). RFMP – An iterative best basis algorithm for inverse problems in the geosciences. In Freeden, W., Nashed, M. Z., and Sonar, T., editors, *Handbook of Geomathematics*, pages 2121–2147. Springer, Berlin, Heidelberg, 2nd edition.

Michel, V. (2015b). Tomography – problems and multiscale solutions. In Freeden, W., Nashed, M. Z., and Sonar, T., editors, *Handbook of Geomathematics*, pages 2087–2119. Springer, Berlin, Heidelberg, 2nd edition.

Michel, V. (2020+). *Geomathematics*. upcoming publication.

Michel, V. and Fokas, A. S. (2008). A unified approach to various techniques for the non-uniqueness of the inverse gravimetric problem and wavelet-based methods. *Inverse Problems*, 24(4):045019.

Michel, V. and Orzlowski, S. (2017). On the convergence theorem for the Regularized Functional Matching Pursuit (RFMP) algorithm. *GEM – International Journal on Geomathematics*, 8(2):183–190.

Michel, V. and Schneider, N. (2020). A first approach to learning a best basis for gravitational field modelling. *GEM - International Journal on Geomathematics*. `https://doi.org/10.1007/s13137-020-0143-5`, last accessed 3 March 2020.

Michel, V. and Telschow, R. (2014). A non-linear approximation method on the sphere. *GEM – International Journal on Geomathematics*, 5(2):195–224.

Michel, V. and Telschow, R. (2016). The regularized orthogonal functional matching pursuit for ill-posed inverse problems. *SIAM Journal on Numerical Analysis*, 54(1):262–287.

Moritz, H. (2010). Classical physical geodesy. In Freeden, W., Nashed, M. Z., and Sonar, T., editors, *Handbook of Geomathematics*, pages 253–289. Springer, Berlin, Heidelberg, 2nd edition.

Muir, T. (1882). *A Treatise on the Theory of Determinants*. Macmillian and Co., London.

Müller, C. (1966). *Spherical Harmonics*. Springer, Berlin.

Narcowich, F. J. and Ward, J. D. (1996). Nonstationary wavelets on the m-sphere for scattered data. *Applied and Computational Harmonic Analysis*, 3(4):324–336.

NASA (2020). Global Climate Change: Scientific Consensus. `https://climate.nasa.gov/scientific-consensus/`, last accessed 3 March 2020.

NASA Jet Propulsion Laboratory (2020). GRACE Tellus. `https://grace.jpl.nasa.gov/`, last accessed 2 April 2020.

National Geospatial-Intelligence Agency, Office of Geomatics (SN), EGM Development Team (2008). Earth Gravitational Model 2008. `http://earth-info.nga.mil/GandG/wgs84/gravitymod/egm2008/`, last accessed 3 March 2020.

Newton, I. (1687). Philosophiae naturalis principia mathematica. *J. Societatis Regiae ac Typis J. Streater*.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, 2nd edition.

Olsen, N. and Mandea, M. (2007). Will the magnetic North Pole move to Siberia? *Eos Earth & Space Science News*, 88(29):293–293.

Olson, P. and Amit, H. (2006). Changes in Earth's dipole. *Naturwissenschaften*, 93(11):519–542.

Pati, Y. C., Rezaiifar, R., and Krishnaprasad, P. S. (1993). Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 40–44.

Pavlis, N. K., Holmes, S. A., Kenyon, S. C., and Factor, J. K. (2012). The development and evaluation of the Earth Gravitational Model 2008 (EGM2008). *Journal of Geophysical Research: Solid Earth*, 117(B4). Correction in Volume 118, Issue 5.

Plattner, A. and Simons, F. J. (2014). Spatiospectral concentration of vector fields on a sphere. *Applied and Computational Harmonic Analysis*, 36(1):1–22.

Poggio, T. and Shelton, C. R. (1999). Machine learning, machine vision, and the brain. *American Association for Artificial Intelligence, AI Magazine*, 20(3):37–56.

Poggio, T. and Smale, S. (2003). The mathematics of learning: dealing with data. *Notices of the American Mathematical Society*, 50(5):537–544.

Prünte, L. (2008). *Learning: Wavelet-Dictionaries and Continuous Dictionaries*. PhD thesis, University of Bremen. `https://elib.suub.uni-bremen.de/diss/docs/00011034.pdf`, last accessed 3 March 2020.

Reuter, R. (1982). *Über Integralformen der Einheitssphäre und harmonische Splinefunktionen*. PhD thesis, RWTH Aachen, Geomathematics Group, Veröffentlichung des Geodätisches Institut der RWTH Aachen, vol. 33.

Rieder, A. (2003). *Keine Probleme mit inversen Problemen. Eine Einführung in ihre stabile Lösung*. Vieweg, Wiesbaden.

Rinnooy Kan, A. H. G. and Timmer, G. T. (1989). Chapter IX Global optimization. In Nemhauser, G. L., Rinnooy Kan, A. H. G., and Todd, M. J., editors, *Optimization*, pages 631–662. Elsevier Science Publishers B.V., North-Holland.

Rubinstein, R., Bruckstein, A. M., and Elad, M. (2010). Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057.

Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1–4):259–268.

Runarsson, T. P. and Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294.

Runarsson, T. P. and Yao, X. (2005). Search biases in constrained evolutionary optimization. *IEEE Transactions on System, Man, and Cybernetics: Part C*, 35(2):233–243.

Russell, S. J. and Norvig, P. (2010). *Artificial Intelligence. A Modern Approach*. Pearson Education, Inc., Upper Saddle River, 3rd edition.

Sakumura, C., Bettadpur, S., and Bruinsma, S. (2014). Ensemble prediction and intercomparison analysis of GRACE time-variable gravity field models. *Geophysical Research Letters*, 41(5):1389–1397.

Schall, J. (2019). *Optimization of point grids in regional satellite gravity analysis using a Bayesian approach*. PhD thesis, University of Bonn. Accepted.

Schall, J., Eicker, A., and Kusche, J. (2014). The ITG-Goce02 gravity field model from GOCE orbit and gradiometer data based on the short arc approach. *Journal of Geodesy*, 88(4):403–409.

Schall, J., Mayer-Gürr, T., Eicker, A., and Kusche, J. (2011). A global gravitational field model from GOCE gradiometer observations. In *Proceedings of the 4th International GOCE User Workshop, Munich, Germany*. ESA SP-696.

Schmidt, R., Flechtner, F., Meyer, U., Neumayer, K. H., Dahle, C., König, R., and Kusche, J. (2008). Hydrological signals observed by the GRACE satellites. *Surveys in Geophysics*, 29(4–5):319–334.

Schreiner, M. (1996). A pyramid scheme for spherical wavelets. *AGTM Report*, (170). Geomathematics Group, University of Kaiserslautern.

Schwarz, H. R. and Köckler, N. (2011). *Numerische Mathematik*. Vieweg + Teubner, Wiesbaden, 8th edition.

Seibert, K. (2018). *Spin-Weighted Spherical Harmonics and Their Application for the Construction of Tensor Slepian Functions on the Spherical Cap*. PhD thesis, University of Siegen, Geomathematics Group, universi – Universitätsverlag Siegen, Siegen.

Senyukov, S. L. (2013). Monitoring and prediction of volcanic activity in Kamchatka from seismological data: 2000-2010. *Journal of Volcanology and Seismology*, 7(1):86–97.

Simons, F. J. (2010). Slepian functions and their use in signal estimation and spectral analysis. In Freeden, W., Nashed, M. Z., and Sonar, T., editors, *Handbook of Geomathematics*, pages 891–923. Springer, Heidelberg.

Simons, F. J. and Dahlen, F. A. (2006). Spherical Slepian functions and the polar gap in geodesy. *Geophysical Journal International*, 166(3):1039–1061.

Simons, F. J., Dahlen, F. A., and Wieczorek, M. A. (2006). Spatiospectral concentration on a sphere. *SIAM Review*, 48(3):504–536.

Sneeuw, N. and Saemian, P. (May 2019). Next-generation gravity missions for drought monitoring. *ESA Living Planet Symposium, Milan, Italy*.

Sparks, R. S. J. and Aspinall, W. P. (2004). Volcanic activity: Frontiers and challenges in forecasting, prediction and risk assessment. In Sparks, R. S. J. and Hawkesworth, C. J., editors, *The State of the Planet: Frontiers and challenges in geophysics, Geophysical Monograph, Volume 150*, pages 359–373. American Geophysical Union.

Stein, O. (2018). *Grundzüge der Globalen Optimierung*. Springer-Verlag, Berlin.

Stoer, J. and Bulirsch, R. (1973). *Numerische Mathematik II*. Springer, Berlin.

Stummel, F. and Hainer, K. (1971). *Praktische Mathematik*. B. G. Teubner, Stuttgart.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction.* The MIT Press, Cambridge, 2nd edition.

Szegö, G. (1975). *Orthogonal Polynomials*. American Mathematical Society Colloquium Publications Vol. XXIII. American Mathematical Society. Providence.

Tapley, B. D., Bettadpur, S., Watkins, M., and Reigber, C. (2004). The gravity recovery and climate experiment: mission overview and early results. *Geophysical Research Letters*, 31(9):L09607. doi:10.1029/2004GL019920, last accessed 3 March 2020.

Telschow, R. (2014). *An Orthogonal Matching Pursuit for the Regularization of Spherical Inverse Problems*. PhD thesis, University of Siegen, Geomathematics Group, Verlag Dr. Hut, Munich.

Temlyakov, V. N. (2000). Weak greedy algorithms. *Advances in Computational Mathematics*, 12(2–3):213–227.

Temlyakov, V. N. (2011). *Greedy Approximation*. Cambridge University Press, Cambridge.

The University of Texas at Austin, Centre for Space Research (2020). Grace gravity recovery and climate experiment. `http://www2.csr.utexas.edu/grace/`, last accessed 3 March 2020.

Vigerske, S., Wächter, A., Kawajir, Y., and Laird, C. (2016). Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT. `https://projects.coin-or.org/Ipopt/browser/stable/3.11/Ipopt/doc/documentation.pdf?format=raw`, last accessed 11 December 2018.

Vikulin, A. V., Akmanova, D. R., Vikulina, S. A., and Dolgaya, A. A. (2012). Migration of seismic and volcanic activity as display of wave geodynamic process. *Geodynamics & Techtonophysics*, 3(1):1–18.

Vincent, P. and Bengio, Y. (2002). Kernel matching pursuit. *Machine Learning*, 48(1–3):165–187.

Wächter, A. (2002). *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University, Pittsburgh, USA.

Wächter, A. and Biegler, L. T. (2005a). Line search filter methods for nonlinear programming: Local convergence. *SIAM Journal on Optimization*, 16(1):1–31.

Wächter, A. and Biegler, L. T. (2005b). Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization*, 16(1):32–48.

Wächter, A. and Biegler, L. T. (2006). On the implementation of a primal-dual interior point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

Watkins, M. M. and Yuan, D.-N. (2012). GRACE, JPL Level-2 Processing Standards Document, For Level-2 Product Release 05. Technical report, Jet Propulsion Laboratory, NASA, Pasadena.

Windheuser, U. (1995). *Sphärische Wavelets: Theorie und Anwendung in der Physikalischen Geodäsie*. PhD thesis, University of Kaiserslautern, Geomathematics Group.

Yosida, K. (1995). *Functional Analysis*. Springer, Berlin, Heidelberg, 6th edition. Reprint of the 1980 edition.

Zörnig, P. (2014). *Nonlinear Programming. An introduction*. Walter De Gruyter, Berlin.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2):301–320.

Index