Fast Radon Transforms and Reconstruction Techniques in Seismology

by Viktor Nikitin

ACADEMIC THESIS

which, by due permissions of the Faculty of Engineering, Centre for Mathematical Sciences of Lund University, will be publicly defended on Friday 19th of August, 2016, at 13.00 in lecture hall MA:01, at the Centre for Mathematical Sciences, Sölvegatan 18, Lund, for the degree of Doctor of Philosophy in Engineering.

Faculty opponent: Prof. Luis Tenorio, Colorado School of Mines, USA
The measurements conducted in tomography and seismology typically yield large multidimensional data sets. This in combination with the fact that the data may have an irregular structure makes it computationally prohibitive to use simple reconstruction methods directly. Hence, for inverse problems in computed tomography and seismology there is a demand for fast computational methods using high-performance computational facilities to find accurate solutions in a reasonable time. We exploit the particular structure of operators involved, investigate their properties and then construct algorithms for fast evaluations. Algorithm implementations are done on CPU and GPU with exploiting Intel and Nvidia facilities for parallel computing.

For computed tomography we develop fast algorithms for evaluating the standard Radon transform and the exponential Radon transform, as well as the corresponding adjoint operators and data inversion schemes. Fast evaluation of the Radon transform is based on using representations in log-polar coordinates, where the operator can be expressed in terms of convolutions and thereby rapidly evaluated by using fast Fourier transforms. Fast evaluation of the exponential Radon transform in turn is based on a generalization of the Fourier slice theorem in the Laplace domain, and here the computations can be made fast by using fast Laplace transforms.

For seismology we construct fast algorithms for data interpolation, compression, denoising, and attenuation of multiple reflections appearing in seismic measurements. Some of these procedures are performed by using sparse representations of seismic data. Sparse representations are for instance obtained with the hyperbolic Radon transform or by decomposing the data with using wave packets. Algorithms for fast evaluation of the hyperbolic Radon transforms are constructed by generalizing the log-polar approach. For the wave-packet decomposition we design fast implementations based on unequally spaced Fourier transforms. We also provide an approach for interpolation of a new type of retrieving seismic data - multicomponent streamer data. The interpolation is formulated in terms of the solution of a partial differential equation that describes how energy is propagated between different parts of the data.

Key words
Radon transform, sparse representation, interpolation, FFT, GPU
Fast Radon Transforms and Reconstruction Techniques in Seismology

Viktor Nikitin
Abstract

The measurements conducted in tomography and seismology typically yield large multidimensional data sets. This in combination with the fact that the data may have an irregular structure makes it computationally prohibitive to use simple reconstruction methods directly. Hence, for inverse problems in computed tomography and seismology there is a demand for fast computational methods using high-performance computational facilities to find accurate solutions in a reasonable time. We exploit the particular structure of operators involved, investigate their properties and then construct algorithms for fast evaluations. Algorithm implementations are done on CPU and GPU with exploiting Intel and Nvidia facilities for parallel computing.

For computed tomography we develop fast algorithms for evaluating the standard Radon transform and the exponential Radon transform, as well as the corresponding adjoint operators and data inversion schemes. Fast evaluation of the Radon transform is based on using representations in log-polar coordinates, where the operator can be expressed in terms of convolutions and thereby rapidly evaluated by using fast Fourier transforms. Fast evaluation of the exponential Radon transform in turn is based on a generalization of the Fourier slice theorem in the Laplace domain, and here the computations can be made fast by using fast Laplace transforms.

For seismology we construct fast algorithms for data interpolation, compression, denoising, and attenuation of multiple reflections appearing in seismic measurements. Some of these procedures are performed by using sparse representations of seismic data. Sparse representations are for instance obtained with the hyperbolic Radon transform or by decomposing the data with using wave packets. Algorithms for fast evaluation of the hyperbolic Radon transforms are constructed by generalizing the log-polar approach. For the wave-packet decomposition we design fast implementations based on unequally spaced Fourier transforms. We also provide an approach for interpolation of a new type of retrieving seismic data - multicomponent streamer data. The interpolation is formulated in terms of the solution of a partial differential equation that describes how energy is propagated between different parts of the data.
Popular science summary

In many situations it is not possible to measure physical properties of some object directly. Instead, indirect measurements that describe the object can be measured. The problem of recovering the desired properties from the indirect measurements is called an inverse problem. Inverse problems of this kind arise for instance in seismology, medical tomography and in x-ray imaging. Modern devices provide high-resolution and high-quality measurements allowing for detailed analysis of the objects. At the same time, the measurements are characterized by multidimensionality, large data sizes, and often an irregular structure. Processing of such data can therefore be rather time-consuming unless good numerical algorithms are used in combination with high-performance computational facilities.

In this study, we develop fast algorithms in computed tomography and seismology for imaging and data representation. Computed tomography refers to a computerized x-ray imaging procedure, in which beams propagate through a rotating object, producing signals that are processed to generate cross-sectional images of the object.

Reflection seismology in turn refers to study the Earth’s subsurface structure. In seismic exploration surveys sources are used for generating seismic waves propagating into the subsurface. Waves reflected from geological boundaries propagate back to the surface where they are recorded by measurement devices. These recorded waves carry information about the locations of the geological boundaries, which is useful for understanding the interior structure of the Earth. The recorded wave field can for instance be used for mineral resources exploration. Seismic data often deals with very large volumes of physical data, often with irregular sampling structures and with a substantial amount of noise presented in the data. Hence, great attention is devoted to processing of the recorded wave field.

In this thesis we develop mathematical models for data processing and for the solution of the inverse problems. We also generalize existing methods for constructing fast computational algorithms to the problems that we consider. High-performance computing also plays an important role in accelerating algorithms, and we utilize modern fast computer processors and video cards to work with large data sets.
Papers and author's contributions

This thesis is based on the publications listed below. The publication are followed by outlines of my contribution.

I Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators
Fredrik Andersson, Marcus Carlsson, Viktor V. Nikitin
I wrote parts of the paper, implemented all the algorithms and ran the simulations.

II Fast Laplace transforms for fast inversion of the exponential Radon transform
Fredrik Andersson, Marcus Carlsson, Viktor V. Nikitin
I wrote parts of the paper, implemented all the algorithms and ran the simulations.

III Fast hyperbolic Radon transform represented as convolutions in log-polar coordinates
Viktor V. Nikitin, Fredrik Andersson, Marcus Carlsson, Anton A. Duchkov
I wrote the main parts of the paper, implemented all the algorithms and ran the simulations.

IV Directional interpolation of multicomponent data
Fredrik Andersson, Adriana Citlali Ramírez Pérez, Torgeir Wiik, Viktor V. Nikitin
I wrote parts of the paper and implemented all the algorithms.

V Parallel algorithm of 3D wave-packet decomposition of seismic data: Implementation and optimization for GPU
Viktor V. Nikitin, Anton A. Duchkov, Fredrik Andersson
I wrote the main parts of the paper, implemented all the algorithms and ran the simulations.
Publications


Acknowledgements

First, I would like to thank my supervisor Fredrik Andersson for the guidance and valuable help during these years of the joint work. Especially, I would like to express my deep gratitude to him for the patience, spending so much time and paying great attention to my scientific and professional development. I am forever grateful for that. All these thesis results would not be reached without Fredrik’s supervising.

I also would like to thank my co-advisors Anton Duchkov, Marcus Carlsson and Carl Olsson. We have many discussions not only about the scientific stuff including mathematics and geology, but also about real life experience and practice. From these discussions I received a lot of useful knowledge and advice for my future. Anton Duchkov was my previous supervisor, and I am thankful for his recommendation to choose Lund University as a place for my PhD study. Marcus Carlsson helped me so much with study of the theoretical framework for this research. Also I am thankful to my previous co-advisor, Alexey Romanenko, for his advice and suggestions about program optimizations.

My sincere thanks to all co-workers at the Centre for Mathematical Sciences. It was very motivating to work in such a great research environment. I am very grateful for the support from Johan Helsing. Special thanks should be also given to our secretaries, Patricia Felix and Lena Lööf, for all the administrative work, and to Carl-Gustav for his hardware support.

Despite the fact that my friends diverted my focus on the research many times, I am still thankful for their belief in me and moral support. Vladimir, Dmitry, Tatiana, Konstantin, Svetlana, Alexander, Viktoria, another Dmitry, and Olga - thank you all. Finally, without my family I would not be the person who I am today, I am forever grateful to my parents and my red brother Sergey.

This work was supported by the Crafoord Foundation and the Swedish Research Council.

Viktor Nikitin,
Lund, Sweden
Contents

1 Introduction ................................................. 2
2 Computed tomography .................................... 3
3 Seismology .................................................. 6
4 Techniques .................................................. 12
  4.1 Fourier transforms and quadratures ................. 12
  4.2 Radon transforms ...................................... 16
  4.3 High-performance computing ......................... 18
5 Main results of the papers ............................... 21
6 References ................................................. 31

Paper I: Fast algorithms and efficient GPU implementations for the
  Radon transform and the back-projection operator represented as convolution operators ............... 33
Paper II: Fast Laplace transforms for fast inversion of the exponential Radon transform ....................... 79
Paper III: Fast hyperbolic Radon transform represented as convolutions in log-polar coordinates ............. 113
Paper IV: Directional interpolation of multicomponent data .... 143
Paper V: Parallel algorithm of 3D wave-packet decomposition of seismic data: Implementation and optimization for GPU .................................................................................. 179
1 Introduction

The problem of deducing from a set of observations the parameters that produced these observations is called an inverse problem. Two important examples where inverse problems arise are computed tomography and seismology, where physical properties of different objects are sought for by using indirect measurements. In the case of computed tomography the properties allow to reconstruct the inner structure of a sample, and in seismology - the subsurface structure of the Earth.

Many direct problems appearing in physics are well-posed, meaning that they have unique solutions and are stable with respect to small perturbations in the data. If one of the properties fails, then the problem is called ill-posed. Additional requirements are typically needed in order to define and find solutions to ill-posed problems \cite{10}. Inverse problems are typically ill-posed and many different techniques to regularize them have been suggested in the literature.

Existence and uniqueness can, for instance, be satisfied by modifying the solution space to seek solutions that approximately satisfy the measurements. Problems of stability typically imply that the obtained solution can be substantially perturbed in the presence of noise or modeling errors. Regularization techniques such as imposing additional conditions on smoothness or that the solution has a sparse representation can be used to control stability of the problems. In many cases one has to rely on iterative reconstruction techniques where the modeling operators are applied several times in order to make stable reconstructions for inverse problems. It then becomes important to have accurate and fast computational algorithms at hand.

The purpose of this thesis is to develop fast computational algorithms that can be used in computed tomography and seismology. Moreover, we will make use of some particular structure that is typically presented in seismic data that can help in the process of recovering information about the subsurface structure.
2 Computed tomography

Computed tomography (CT) refers to a computerized imaging procedure in which beams propagate through a rotating object, producing signals that are processed by a computer to generate cross-sectional images of the object. Measurement data can be obtained from different modalities, where the two most common ones are cone beam CT and parallel (fan) beam CT. The first emits a cone type beam and is detected by a flat panel sensor, whereas the second emits a parallel type beam and is detected by a linear detector array.

The cone beam tomography is commonly used in medicine, where it becomes increasingly important in for instance treatment planning, diagnosis in implant dentistry and interventional radiology. In this study, however, we focus on parallel beam CT, where the measurements can be performed by using x-rays or neutron parallel beams. These two types of beams are currently of significant importance at Lund due to the establishment of the MAX IV Laboratory (synchrotron light source) and the European Spallation Source (neutron source). In the left panel of Figure 1 a setup for collecting the necessary data in the case of neutron CT is shown. X-ray CT with parallel beams can be modeled by a similar setup. Neutron radiography is based on the principal that neutrons interact with the nucleus of the atom, rather than the electrons. Therefore neutrons are absorbed in matter quite differently from x-rays. This means that, contrary to x-rays, neutrons are attenuated by some light materials, such as hydrogen, boron and lithium, but penetrate many heavy materials such as titanium and lead. This leads to a difference between applications of x-ray and neutron radiography. For instance, dark elements in x-ray radiographs could be caused by metal components, whereas dark components in a neutron radiograph could be due to plastic components which in turn are almost transparent to x-rays.

The beam propagation process can be estimated by line integrals over functions representing density or other physical properties of an object along the line. Doing this repeatedly while rotating the sample, it is possible to obtain several such sets of measurements. Each set of estimated line integrals is often called a projection. To describe the process formally, let the function $f(x)$ be the density of the object at the position $x = (x_1, x_2)$. Then for each particular angle $\theta$ and distance $s$ of the line to the origin, the projection is
given by the Radon transform

$$\mathcal{R} f(\theta, s) = \int_{s=x \cdot \theta} f(x) dx = \int_{-\infty}^{\infty} f(s \theta + t \theta^\perp) dt,$$

where $\theta$ denotes a point on the unit circle. By abuse of notation, $\theta$ can also be identified with an angle. The notation $x \cdot \theta = x_1 \cos(\theta) + x_2 \sin(\theta)$ is used to parameterize lines. A schematic illustration of the transform is shown in the right panel of Figure 1. The Radon transform was introduced in 1917 by Johann Radon in his work “On the determination of functions from integrals along certain manifolds” [18], where he proved that it was possible to reconstruct a function from knowledge of its line integrals.

If the sample itself contains emitting sources then the measurements can be modeled using the attenuated Radon transform

$$\mathcal{R}_a f(\theta, s) = \int_{-\infty}^{\infty} f(s \theta + t \theta^\perp) e^{-\int_{\tau_{\theta + t \theta^\perp}}^{\infty} \mu(s \theta + \tau \theta^\perp) d \tau} dt,$$

where the function $\mu$ describes the attenuation. This type of the Radon transform is used in emission computed tomography of quantitating the distribution of gamma emitting radiopharmaceuticals in the body.

Image reconstruction from projections is the process of producing an image of a two-dimensional distribution (structure or physical properties) from
estimates of its line integrals along a finite number of lines of known locations. For transforms $R$ and $R_{dal}$ there exist exact inversion formulas [1, 13, 14, 16]. In some cases iterative reconstruction techniques may be required, for instance for dealing with data containing noise with certain particular distribution [3, 11, 19] or in cases where there is missing data [9, 12, 17]. In the most common setup, the iterative methods rely on applying the Radon transform and the corresponding adjoint operator, called back-projection, several times.

To demonstrate how the reconstruction procedure works in practice we show some examples of real data processing of x-ray tomography data. The reconstructions displayed here are computed using the reconstruction techniques developed in this thesis for fast and accurate evaluation of the standard Radon transform operator $R$ and corresponding back-projection. Figures 2-3 shows examples of 3D reconstructions (set of 2D slices) of insects.

Figure 2: Measurement data for two slices of an insect (above). Corresponding reconstructed cross-sections (below)
3 Seismology

The process of seismic exploration can be roughly described as follows. Special sources generate seismic waves which propagate into the subsurface; abrupt (wavelength scale) changes in material mechanics act as internal boundaries, causing reflection of the waves. After reflection the waves propagate back to the surface where they are recorded by receivers. The recorded signal can then be used for constructing images of the subsurface boundaries, which in turn can be used for mineral resources exploration. The signal recorded in time by one of the receivers is called a seismic trace. A seismic gather is a collection of seismic traces that share some geometric attribute, for instance that they all correspond to data caused by the same source.

The most important visual characteristic of seismic reflection data is the presence of waves (coherent space-time structures). Wave propagation in the subsurface is described by solutions of the wave equation

\[
\frac{\partial^2 u(x, z, t)}{\partial z^2} + \frac{\partial^2 u(x, z, t)}{\partial x^2} - \frac{1}{v(x, z)^2} \frac{\partial^2 u(x, z, t)}{\partial t^2} = \delta(x - x_s)f(t),
\]

with initial and boundary conditions. Here \(u(x, z, t)\) is the wavefield (pressure field in the acoustic case) displacement function of reflection time \(t\) at any position \((x, z)\) in the subsurface, \(v(x, z)\) is the seismic velocity in the subsurface, and \(\delta(x - x_s)f(t)\) is the point source at the position \(x_s\). Recall that the inverse problem is to find the velocity function \(v(x, z)\) from the data measured at the surface \((d(x, t) \equiv u(x, 0, t))\). In Figure 4 we show an example of layered velocity model (left) and corresponding data computed by solving the
Figure 4: Synthetic velocity model and corresponding solution of wave equation.

wave equation with a finite difference scheme (right); the source was placed at the surface at position 0.5 km. The velocity model contains discontinuities referred to as reflectors. The data contains obvious coherent energy along hyperbolic curves (arriving waves) referred to as reflections.

It should be noted that acquired seismic data is typically characterized by redundancy since it is measured at different positions and during several experiments where the source position $x_s$ is also changed. This type of acquisition is called the reflection seismology with multifold acquisition. Combining the data for all receivers we get the redundant data set $d(x_s; x, t)$ with more coordinates than in the unknown velocity function $v(x, z)$.

This redundancy allows to improve the signal-to-noise ratio (SNR) at the processing stage. In particular, such data can be sorted into the so-called common mid-point (CMP) gathers containing traces with waves reflected from the same common subsurface reflection point. Figure 5, left schematically illustrates the method for a velocity model containing three boundaries. Traces in the CMP gather are parameterized by the distance from the source to the receiver which is called offset. This type of acquisition geometry is commonly used in towed marine exploration.

Let us get back to the structure present in seismic gathers as illustrated in In Figure 4, right. Reflections (reflected waves) arrive to neighbor receivers with a similar waveform but some delay depending on the wave arrival dir-
Seismology

Figure 5: Scheme of the reflection seismology with multifold acquisition (left). Hyperbolic structure of acquired wave signals (right).

rection. Thus locally (for a group of neighboring receivers) one can identify waves arriving at particular *arriving time* and *slope*. One can see even more structure in the data. Reflections from flat boundaries produce arrival impulses forming a curve since it takes longer to travel to the far receivers than the near receivers (cf. Figure 5, left panel). Thus waves with a similar waveform are aligned along *traveltime curves*. Due to the reciprocity principle in the CMP gather the traveltime curves are symmetric with respect to the offset \( x \). For sufficiently small offsets the traveltime curve can be approximated by a hyperbola

\[
t_x^2 = t_0^2 + \frac{x^2}{v^2},
\]

where \( x \) is the offset, \( v \) is the effective summation velocity, \( t_0 \) and \( t_x \) denote travel times at 0 and \( x \) offsets, respectively. The right panel of Figure 5 shows the structure of these hyperbolas. Note that in real seismic data the hyperbola curvature is smaller for the deeper reflections as there is a general trend of increasing velocity with depth (due to rock compaction and pressure increase).

Real seismic gathers usually have more complicated structure. They are contaminated with random noise and coherent noise corresponding to arriving unwanted waves, e.g. surface and multiple waves etc. Traveltime curves deviate from the hyperbolic form at larger offsets in case of complicated sub-
Figure 6: Example of a real CMP gather.

surface structure. Trace sampling can be irregular and contain missing traces due to acquisition problems. Figure 6 shows an example of a real CMP gather. Note that the structure mentioned above is still noticeable in the data but recovering it requires careful processing.

Let us mention the main steps of seismic processing (imaging):

1. Preprocessing
2. Velocity analysis and stacking
3. Migration

**Preprocessing.** Seismic data preprocessing refers to applying such procedures as signal-to-noise enhancement, multiple suppression, data regularization and interpolation. This step is important since otherwise the following processing steps give high errors and in some cases can not be applied. Noise in data can be partially removed by using various filtering operations. Often the noise lies in high-frequency components of the signal, therefore it can be removed by *low-pass filter* determined by some cut-off frequency. Another type of filtering, *F-K filtering*, is related to applying the two-dimensional
Fourier transform in time and offset variables and recovering data by using only particular values of the spectrum. This technique is generally used for removing coherent noise.

Events on a CMP gather that incurred more than one reflection are called multiples, and the reflections that have just been scattered ones are called primaries. Multiples in seismic data continue to be a serious problem for processing. Numerous techniques exist for removing multiples with varying degrees of quality and complexity. Generally, the problem is addressed by using parabolic or hyperbolic Radon transforms since parabolas and hyperbolas are well-suited for approximations of seismic waves at near offsets. The transforms for the function $f$ representing a CMP gather are given by

$$\mathcal{R}_p f(\tau, q) = \int f(\tau + qx^2, x) dx,$$  \hspace{1cm} (2)

$$\mathcal{R}_h f(\tau, q) = \int f(\sqrt{\tau^2 + q^2 x^2}, x) dx.$$  \hspace{1cm} (3)

Here, the parameter $q$ is related to wave velocities; and $\tau$ represents the intercept time at zero offset. Seismic waves in the Radon ($\tau, q$) domain are easier to separate than looking at the CMP gather. Multiple suppression is done by vanishing the corresponding part of the Radon domain.

The parabolic and hyperbolic Radon transforms can be also used for other preprocessing procedures, for instance in data interpolation, since after going back from the Radon to the time-offset domain the structure of waves at missed positions can be recovered.

**Velocity analysis and stacking.** The aim of velocity analysis is to find the velocity that corresponds to a reflection wave. A plurality of sources and receivers are used for enhancing the underlying data structure, which in turn improves the velocity analysis of CMP gathers. The analysis is performed with the assumption that the reflections in the CMP gather correspond to hyperbolas

$$t^2(x) = t_0^2 + \frac{x^2}{v_{st}^2},$$

where $v_{st}$ is a stacking velocity. The first step of the velocity analysis is to find $v_{st}$. The stacking velocity for a chosen time $t_0$ can be easily found since different velocities form different structure of the hyperbolas on the CMP.
Seismology

gather. A velocity spectrum is obtained after summing the energy of the seismic traces along the hyperbolas. Then the velocity that yields the largest energy in the spectrum is chosen to be the stacking velocity $v_{st}$. The hyperbolic Radon transform (3) can be used for the summing procedure. The next step is Normal Moveout (NMO) correction. The reflections are aligned using the correct velocity, such that the event structure is transformed from hyperbolic to horizontal. Having the formula for travel-time curve of reflections, the NMO-correction can be derived and is given by

\[ \Delta t = t_0 - t(x) \quad \text{with} \quad t(x) = \sqrt{t_0^2 + \frac{x^2}{v_{st}^2}}. \]

Finally, stacking the normal moveout corrected traces generates a single trace. Each trace corresponds to a zero-offset trace, that is, the seismic trace that would have been recorded by a receiver that is coincident with the source. The CMP stacking improves continuity of the data since it combines stacking of in-phase signal and out of phase random and coherent noises recorded in traces.

**Migration.** Seismic migration is a technique that creates an image of earth structure from the data recorded by receivers. All of the methods of doing migration are based on solutions to the wave equation that models how waves propagate into the earth. Wave propagation into the surface is described by solutions of the wave equation given in the beginning of this chapter. The migration process in turn concerns to considering the wave equation backward in time. In this case, the boundary condition $u(x, t)$ for the partial differential equation is given by the measured data. Migration can be applied after CMP stacking (poststack migration) so as to unstacked data (prestack migration). The type of migration to choose depends on the quality of the CMP stacking and computational resources.
4 Techniques

In this chapter we describe the main techniques used in this thesis for constructing fast algorithms in computed tomography and seismology.

4.1 Fourier transforms and quadratures

The Fourier transform is a fundamental scientific tool. It is greatly used in computational sciences, and a major reason for that is because the discrete Fourier transform can be rapidly evaluated using the Fast Fourier Transform (FFT).

We will use the following definition of the Fourier transform,

$$\mathcal{F} f(\xi) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i x \xi} \, dx,$$

where the variable $x$ often refers to a spatial coordinate, and the reciprocal variable $\xi$ refers to a frequency coordinate. The Fourier transform is unitary, meaning that the adjoint operator is also the inverse operator: $\mathcal{F}^* = \mathcal{F}^{-1}$, and it is given by

$$\mathcal{F}^{-1} f(x) = \int_{-\infty}^{\infty} f(\xi)e^{2\pi i x \xi} \, d\xi.$$

In many situations, for instance in CT, the operators of interest (e.g. reconstruction formulas) are specified in terms of Fourier transforms. However, the reconstructions need to be computed using sampling. To this end quadrature rules that approximate the continuous Fourier transform are needed, i.e.,

$$\int_{-\infty}^{\infty} f(x)e^{-2\pi i x \xi} \, dx \approx \sum_{j=1}^{N} w_j f(x_j)e^{-2\pi i x_j \xi}.$$

The points $x_j$ are usually referred to as nodes and $w_j$ are referred to as weights. Of particular interest is the case where the points $x_j$ are equally spaced with a sample distance $\Delta_x$, and where the output values $\xi$ lie on a equally spaced lattice with a sample distance of $\Delta_\xi$ such that $N\Delta_\xi \Delta_x = 1$. In this case FFT can be used to rapidly evaluate the quadrature sum above. We will now briefly discuss when such an approximation is possible, and after that we will discuss
how to construct fast algorithms for treating cases where the nodes \( x_j \) are not equally spaced.

The equally spaced sampling is the most simple quadrature rule, and it is commonly referred to as the trapezoidal rule (although that one contains minor adjustments at the endpoints), specifically described by

\[
\int_a^b g(x) dx \approx \sum_{k=0}^{N} \omega_k g(x_k),
\]

where \( \omega_k = 1/2, 1, \ldots, 1, 1/2 \) and \( x_k = a + k \frac{b-a}{N} \). The accuracy of quadrature rules are commonly described in terms of how rapidly the error decreases as the sampling becomes denser. The trapezoidal rule will in general not have good convergence rate, but there are very important exceptions. One such concerns the case where \( f \) is a smooth function such that \( f \) and all of its derivatives vanish outside some interval \([a, b]\). For this case the trapezoidal rule converges super-algebraically, meaning that the error decays faster than \( O(N^{-p}) \) for any \( p \) where \( N \) denotes the total number of sample points. For such functions, the trapezoidal rule will converge very fast for limited values of \( \xi \) as soon as the sampling rate becomes sufficiently high.

Algorithms for fast Fourier transforms FFT decrease computational complexity time complexity from \( O(N^2) \) to \( O(N \log N) \) if we assume that number of samples in \( x \) and \( \xi \) coordinates both have order of \( N \). The main idea of the algorithms is to recursively split the Fourier sum by two parts containing elements with even and with odd indexes. Such scheme works better with \( N = 2^n \). However, there are other fast algorithms (e.g. the Bluestein algorithm) for treating cases where \( N \) is not a power of two.

Let us now turn our attention to the case of rapid evaluation of quadrature rules where the nodes are not equally spaced. Such algorithms are sometimes referred to as algorithms for unequally-spaced Fast Fourier Transforms (USFFT), but other acronyms such as NFFT and NUFFT are also often used in the literature [6, 15].

Let us now assume that nodes \( \{x_j\}_{j=1}^{J} \) are given, and that they (are scaled to) satisfy the condition \( |x_j| < 1/2 \). We see fast algorithms for the evaluation
4 Techniques

of the operations

\[ F = \mathcal{F}_{\mathbb{R} \rightarrow \mathbb{Z}}(f) : \quad F_n = \sum_j f_j e^{-2\pi i nx_j}, \]

\[ f = \mathcal{F}_{\mathbb{Z} \rightarrow \mathbb{R}}(F) : \quad f_j = \sum_n F_n e^{-2\pi i nx_j}, \]

where we use the notations \( \{F_n\}_{n=1}^N \) and \( \{f_j\}_{j=1}^J \) to denote discrete samples. USFFT is based on combining convolution-type operations with FFT to approximate the sums above at arbitrary, but fixed, accuracy. We provide an heuristic description of how it works, for more precise formulation along with error estimates, see [2, 5, 6, 8].

We begin with the distribution representation of the exponents

\[ e^{-2\pi ix_j \xi} = \int_{-\infty}^{\infty} e^{-2\pi i x \xi} \delta(x - x_j) dx. \]

It follows that

\[ F_n = \sum_j f_j e^{-2\pi i nx_j} = \int_{-\infty}^{\infty} \sum_j f_j \delta(x - x_j) e^{-2\pi i nx} dx = \]

\[ \int_{-\infty}^{\infty} F(x) e^{-2\pi i nx} dx, \]

where \( F(x) = \sum_j f_j \delta(x - x_j) \). Let \( \varphi \) be a bump function, e.g. a Gaussian. Other examples are B-spline or Kaiser-Bessel functions; for details see the references mentioned above. By multiplying (6) by \( \mathcal{F} \varphi \) we obtain

\[ \mathcal{F} \varphi(n) F_n = \int_{-\infty}^{\infty} \sum_j f_j \varphi(x - x_j) e^{-2\pi i nx} dx. \]

or

\[ F_n = \frac{1}{\mathcal{F} \varphi(n)} \int_{-\infty}^{\infty} \sum_j f_j \varphi(x - x_j) e^{-2\pi i nx} dx \]

The integral above can be approximated by the trapezoidal rule because the integrable function is smooth enough due to the convolution with the function \( \varphi \). Therefore, accurate results can be obtained by samples of \( x \) given on
a regular grid with some oversampling factor $\nu$. Another important observation for choosing function $\varphi$ is that it has small numerical support. This fact allows to decrease computational times for convolutions. Moreover, the numerical width of $\varphi$ depends on the requested accuracy ($\epsilon$) and thus function $\varphi$ is typically changed by its thresholded version $\varphi^\epsilon$. To this end, the approximation $\mathcal{F}_{R \to Z}(f)$ can be rapidly computed in three steps

1. Convolution-type operation in the spatial domain
2. FFT
3. Division in the frequency domain

with respect to the approximation formula with accuracy $\epsilon$

$$F_n = \sum_j f_j e^{-2\pi i x_j n} \approx \frac{1}{\mathcal{F}\varphi(n)} \frac{1}{\nu N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \left( \sum_j f_j \varphi^\epsilon \left( \frac{k}{\nu N} - x_j \right) \right) e^{-2\pi i \frac{kn}{\nu N}},$$

where the sum over $k$ variable consists of $M \ll N$ non-zero terms. The constant $M$ depends on the chosen function $\varphi$ and requested accuracy $\epsilon$.

The first operation (division) has $O(MN)$ computational cost, the second operation is evaluated by FFT in $O(\nu N \log(N))$ operations, and the third operation costs $O(N)$. The total complexity is thus $O(\nu N \log N + MN)$, and in practice the most time consuming part comes from the $O(MN)$ part.

The algorithm for $\mathcal{F}_{Z \to R}(F)$ can be constructed in a similar manner by applying the operations in reverse order, since after changing $x_j \to -x_j$ the operator $\mathcal{F}_{Z \to R}(F)$ becomes adjoint to $\mathcal{F}_{R \to Z}(f)$.

Fourier transforms are used in all papers presented in this thesis. In all papers we address problems of accurate discretization with constructing low or high order composite quadrature rules depending on the setup. In Paper I and Paper III we show how to effectively compute convolutions in log-polar coordinates in the sense of sampling rates and accuracy. In Paper II we deal with unequally spaced fast Laplace transforms (USFLT), which is a generalization of USFFT. In Paper III we also consider an interpolation technique with cardinal cubic B-splines, which is related to that used for USFFT, in the way that the interpolation is conducted by smearing data in one of the domains, and the compensating for that effect is done in the reciprocal domain.
In **Paper IV** Fourier transforms are used for the quality control of the directional interpolation.

In **Paper V** we use three-dimensional versions of USFFT as a core component for the synthesis and analysis of function using wave-packets.

### 4.2 Radon transforms

Radon transforms are fundamental tools for computed tomography, as well as in seismic data processing. Recall, that in the field of computed tomography the standard Radon transform over straight lines is defined for angles $\theta$ and distances $s$ of lines to the origin by formula (1). In seismic processing the Radon transform are often referred to as the *tau-p transform* or *slant stack*, and it is defined for time interception ($\tau$) and the slope ($p$) of the wave by

$$
\mathcal{R}_s f(\tau, p) = \int_{-\infty}^{\infty} f(\tau + px, x) \, dx.
$$

(7)

In seismology the function $f$ represents data measurements (e.g., CMP gathers) and the Radon transform is used for processing these measurements, whereas in computed tomography the measurements are given by Radon data $\mathcal{R} f$ and the goal is to recover function $f$ from these measurements. Seismic processing also operate with Radon transforms over curves, such as parabolas or hyperbolas; see formulas (2) and (3).

Let us first discuss how to reconstruct tomography data measurements. An important theorem associated with the Radon transform over straight lines is the *Fourier slice theorem*, which relates one-dimensional Fourier transforms of Radon transformed data with the two-dimensional Fourier transform of data. The relation is obtained by the following formal calculation,

$$
\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathcal{R} f(\theta, s) e^{-2\pi i s \sigma} \, ds = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(s \theta + t \theta^\perp) e^{-2\pi i s \sigma} \, dt \, ds =
$$

$$
\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(s \theta + t \theta^\perp) e^{-2\pi i (s \theta + t \theta^\perp, \theta) \sigma} \, dt \, ds = \int_{\mathbb{R}^2} f(x) e^{-2\pi i (x, \theta) \sigma} \, dx.
$$

An inverse Fourier transform now yields

$$
f(x) = \int_{0}^{\pi} \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} \mathcal{R} f(\theta, s) e^{-2\pi i s \sigma} \, ds \right) e^{2\pi i (x, \theta) \sigma} |\sigma| \, d\sigma \, d\theta.
$$
The obtained formula is called filtered back-projection, and $|\sigma|$ plays the role of an inverse filter: multiplying by $|\sigma|$ increases the influence of the Radon data at high frequencies. Therefore the inversion formula is not robust towards to noise typically presented by high-frequency components of the data. In practice, the filter $|\sigma|$ is changed by other filter types to decrease high-frequency components without big adverse effects on reconstruction. There exist many reconstruction filters, Figure 7 illustrates commonly used ones. The ramp filter is a cropped version of the inverse filter $|\sigma|$; Shepp-Logan and Cosine filters multiply the ramp filter by sinc and cosine functions, respectively; Hann filter is obtained by multiplying the ramp filter by the Hann window $\frac{1+\cos(2\pi\sigma)}{2}$.

In the Papers I and II fast algorithms for computed tomography are developed. In Paper I we propose a method for fast evaluation of operators $R, R^*$, and perform accuracy tests with using different types of filters for inversion. We also consider iterative methods for dealing with incomplete measurements or measurements with noise. The iterative methods rely on applying the Radon transform and the back-projection operator several times. In Paper II we show how the filtered back-projection formula can be generalized for the exponential Radon transform.

As was mentioned before, Radon transforms are also used in seismic data processing. The structure of seismic waves can be approximated by hyperbolas in the simplified cases where the earth is modeled by a one-dimensional
4 Techniques

function. However, the hyperbolic structure is still of interest for the models represented with mild deviations from the one-dimensional layers structure. Therefore the processing will be more efficient when working with Radon transforms over hyperbolic curves (3) instead of straight lines as for the tau-p transform (7).

In many applications there is a need for a sparse representation of seismic data using hyperbolic wave events. A sparse representation can be obtained by minimization of

\[ \| R_h^* g - f \|_2^2 + \mu \| g \|_1, \]

where \( R_h^* \) is the adjoint operator to the hyperbolic Radon transform, and \( \mu \) is a sparsity parameter. With the first term in the sum we aim to find a good approximation of the function: \( R_h^* g \approx f \), the second term in turn promotes sparse solutions with a small number of non-zeros elements in \( g \).

In Paper III we propose a method for fast evaluation of the hyperbolic Radon transform and the corresponding adjoint operator. We also apply standard algorithms for solving the minimization problem to perform seismic processing procedures such as multiple suppression, interpolation, and denoising using the developed fast algorithms.

4.3 High-performance computing

High-performance computing (HPC) is the use of parallel processing for running advanced application programs efficiently and fast. HPC is used with a growing number of industrial applications and scientific algorithms, which makes it a key inter-disciplinary tool. Its popularity is caused by large measurement data sizes and high computational complexity of processing algorithms.

Nowadays the term HPC is not only referred to supercomputers but also to a standard desktop computer containing, for instance, a modern Intel i7 processor or Nvidia video card. It is common to use the term Central Processing Unit (CPU) when referring to standard processors and Graphical Processing Unit when referring to processors located on video cards. Architecturally, the CPU is composed of just a few cores (e.g. Intel i7 processor has 4 or 6 cores) with lots of hierarchical cache memory that can handle a few software threads at a time. Besides, most of the new processors support vectorization techniques, where with making use of Single instruction, multiple
data (SIMD) registers one instruction can be carried out for a series of adjacent values. In contrast, a GPU has a simpler cache hierarchy since the cache memory is mostly connected to several cores. GPUs are composed of hundreds of cores that can handle thousands of threads simultaneously. This kind of parallelism is typically referred to as Single instruction, multiple threads (SIMT) parallelism.

Which architecture to use depends on constructed algorithms and software compatibility. GPUs are optimized for taking huge batches of data and executing a sequence of programmed instructions independently by each computing core. The ability of a GPU with 300+ cores to process thousands of threads can accelerate some software by 100x over a CPU alone. At the same time, this could work quite bad if there is high dependence between the data processed by different threads. This means that sometimes the threads have to ‘wait’ for their turn for computing, which sufficiently slows down the execution. In this situation it could be reasonable to use CPUs with a lower number of parallel threads to decrease resource lockout time. Another issue with GPU computing is that the processing data has to be transferred between operative memory and GPU memory, taking additional overhead time. In addition, most parallel programs written for standard processors do not require much additional software to install, whereas scientific GPU programs in most cases need drivers and supplementary libraries.

Despite all of the issues with GPU computing it is more frequently used in such scientific fields as computed tomography and seismology, because many parallel algorithms can be constructed in the way that parts of tomography or seismic data are processed independently.

One of the most commonly used facilities for parallel computing on CPU is OpenMP (Open Multi-Processing) interface that supports shared memory multiprocessing programming in C language. Performance with Intel processors can be also increased by utilizing Intel Math Kernel Library (Intel MKL) which includes highly vectorized and threaded Linear Algebra, Fast Fourier Transforms (FFT), Vector Math and Statistics functions.

Scientific GPU computing is primarily related to Nvidia CUDA (Compute Unified Device Architecture) technology designed for developing GPU programs with programming languages such as C, C++ and Fortran. The CUDA programming model is a heterogeneous model in which both the CPU and GPU are used. In CUDA, the host refers to the CPU and its memory,
while the *device* refers to the GPU and its memory. Code run on the host can manage memory on both the host and device, and also launches *kernels* which are functions executed on the device. These kernels are executed by many GPU threads in parallel.

Scientific algorithms can be also implemented with using standard routines for computer graphics, in particular standard routines from OpenGL (Open Graphics Library) [4]. The library represents a cross-platform interface for interaction with a graphics processing unit (GPU), to achieve hardware accelerated rendering. OpenGL is used, for instance, in modeling the physically-based deformation process or interpolation of vector fields given at unequally spaced samples. These two processes are similar when considering the rendering procedures with OpenGL. The library provides an interface for writing special functions - vertex and fragment shaders executed on GPUs. A scheme of the rendering can be briefly described as follows. First, the whole surface is split by triangles; vertices of the triangles are used as manipulating nodes. Each node has its own coordinates and a corresponding data value. Processes of interpolation and retrieving values between vertices of triangles (triangular interpolation) are accelerated due to the GPU pipeline structure. To apply a surface deformation or interpolation of vector fields, the vertices of triangles only change their coordinates. The values at the nodes keep constant and mostly stored in the texture cache of GPU, decreasing time for their retrieving.

We employ HPC for implementing all the methods proposed in this thesis work. We obtained programs which are several tens of times faster compared to sequential versions. Program implementation of the methods from *Papers I,III,V* were performed on GPU using Nvidia CUDA technology. For *Paper II* we utilized Intel facilities to have a fast and easily distributable program. Implementation of the methods from *Paper IV* is done by making use Nvidia CUDA technology, as well as standard routines from OpenGL for interpolation of vector fields.
5 Main results of the papers

Paper I

In this paper we construct a fast algorithm for the evaluation of the standard Radon transform, i.e.,

\[ R f(\theta, s) = \int f(x) \delta(x \cdot \theta - s) dx. \]

The proposed method is based on switching to log-polar coordinates at which the transform becomes translation invariant. It can thus be represented by means of convolutions. Convolutions can be rapidly evaluated by using FFT, and in this way decrease the total computational cost. A direct evaluation of the Radon transform in terms of line integrals has a \( O(N^3) \) computational complexity if we assume that number of samples for each coordinate axis in spatial and Radon domains are of the order \( O(N) \). In contrast, the proposed log-polar-based method has a time complexity of \( O(N^2 \log N) \).

Other fast methods for evaluating the Radon transforms exist in the literature. The most popular versions are based on using the Fourier slice theorem. Here, an interpolation-like procedure has to be performed in the frequency domain, and since data in the frequency domain is typically rather oscillatory it is required to use high interpolation order for the method to be accurate. The proposed method also requires interpolation, but this interpolation is performed either in the spatial or Radon domains where data are typically less oscillatory and where it is enough to use low-order interpolations.

The idea is to make the change of variables \( x_1 = e^\rho \cos(\theta), x_2 = e^\rho \sin(\theta), s = e^\rho \). With this change of variables the Radon transform can be evaluated by computing convolutions of the form

\[ R f(\theta, e^\rho) = \int_{-\pi}^{\pi} \int_{-\infty}^{\infty} f(\theta', \rho') e^{\rho' \zeta (\theta - \theta', \rho - \rho')} d \rho' d \theta'. \]

By discretization, periodic extension and suitable geometric transformations, the above convolutions can be approximated by using FFT. The transfer function corresponding to the distributions \( \zeta \) can be accurately precomputed. We derive sampling conditions for accurate approximation methods.

Several interpolation steps needed for the proposed method. Here we
propose to make use of cardinal cubic B-splines, which seems to be sufficient accurately while being computationally very efficient to evaluate. Accuracy tests were performed with different types of filters applied to tomography data, where different amounts of high-frequency suppression is applied to suppress noise or other data irregularities. We also consider an iterative scheme for reconstruction of data containing Poisson noise. The scheme is based on applying the Radon transform and corresponding back-projection operator several times.

The algorithms were implemented on both CPU and GPU platforms. In particular, the GPU platform can favorable be used for the proposed method due to the fact that linear interpolation is hard-wired on GPUs, meaning that it has the same computational cost as direct memory access. Moreover, cubic order interpolation schemes can be constructed by combining linear interpolation steps and this provides important computation speedup. The second reason of choosing the GPU platform is the existence of high-performance libraries to apply FFT and to perform matrix-vector computations. Finally, GPUs are well-adapted for iterative schemes, where during all iterations the processing data can be stored in device memory, and unloaded only after the final iteration.

The algorithms were implemented by using Nvidia CUDA C++ technology with MEX interface to MATLAB. The developed software package is publicly available. The implementation is several times faster than those ones of other software packages based on GPU implementations of the Radon transform and the back-projection operator, while typically being amongst the more accurate methods. The accuracy of the Fourier-based methods controlled at user level, but where increased accuracy leads to increased computational cost. For the same accuracy level the CPU version (as well as the GPU version) of the log-polar-based method has a higher gain in computational speed compared to corresponding CPU implementations of the Fourier-based methods.


d Paper II
In this paper we propose a fast algorithm for computing the attenuated Radon transform with constant attenuation parameter. The transform is referred to as the exponential Radon transform. For an attenuation parameter $\mu$ the
transform is given by the integral transform as
\[ \mathcal{R}_\mu f(\theta, s) = \int_{-\infty}^{\infty} f(s \theta + t \theta^\perp) e^{\mu t} \, dt. \]

Standard evaluation of the transform in terms of weighted line integrals has computational complexity of \( O(N^3) \) if we assume that number of samples for each coordinate axis in spatial and Radon domains are of the order \( O(N) \). In practical measurements the transform has to be performed for a large amount of slices, hence it requires extra computational resources and processing time.

The proposed fast method (with complexity \( O(N^2 \log N) \)) is based on using the Fourier-Laplace slice theorem which is a generalization of the standard Fourier slice theorem. To see this it is enough to take the one-dimensional Fourier transform of the exponential Radon data with respect to the variable \( s \). It is straightforward to establish
\[ \int_{-\infty}^{\infty} \mathcal{R}_\mu f(\theta, s) e^{-2\pi i s \sigma} \, ds = \int_{\mathbb{R}^2} f(x) e^{-2\pi i \langle x, \theta \rangle \sigma + \mu \langle x, \theta^\perp \rangle} \, dx, \]

and conclude that the obtained integral is nothing but the two-dimensional Laplace transform to the grid \( \sigma \theta + \frac{i \mu}{2\pi} \theta^\perp \).

The exponential Radon transform, as well as the corresponding adjoint operator can be rapidly evaluated by using fast algorithms for unequally spaced Laplace transforms (USFLT). Similar to USFFT, the USFLT algorithms utilize convolution type operations with FFT to achieve approximations of the Laplace sums with arbitrary, but fixed, accuracy. Approximation theorems for computing USFLT deal with modulated Gaussians of the form \( \varphi_a(x) = e^{-(ax - i\beta \alpha)^2} \) for some parameters \( \alpha \) and \( \beta \).

We adapt the Tretiak and Metz inversion formula for our proposed notation related to the Laplace transforms, where the formula has a simple form. In cases where there is not sufficient data for the standard inversion formula to apply, we show how the reconstruction problem can be reformulated in terms of a deconvolution problem. The deconvolution problem can be solved by using iterative methods.

The inversion formula is given by
\[ f = \mathcal{R}_{-\mu}^* \mathcal{W} \mathcal{R}_\mu f, \]
and besides the exponential Radon transform operator, it contains the modified \((\mu \rightarrow -\mu)\) adjoint operator \(\mathcal{R}_{-\mu}^*\) so as the filtering operator applied with respect to \(s\) variable in the frequency domain \(\mathcal{W}\). Accurate approximation of the inversion formula can not be done by using standard quadrature rules due to singularities coming with the filtering step. The problem can be addressed by an oversampled representation of the Radon data in the Laplace domain. The singularities are located in the low-frequency components, therefore higher oversampling is required in this frequency range to deal with the singularities accurately. However, oversampling is not needed for accurate reconstruction of the higher frequencies. High computational resources for the reconstruction algorithm are required when treating the unequally spaced data in the Laplace domain, hence it is desirable to keep the total number of samples as low as possible. We therefore propose to use a dense sampling for low-frequency range, and gradually decrease the number of grid points for higher frequencies. To do this we employ composite quadrature rules applied after splitting the integral by parts. The sampling rate is increased for the small parts containing singularities and keep low for the integrals over long intervals.

We perform accuracy tests for different values of the attenuation parameter \(\mu\). We use the modified Shepp-Logan phantom image for the tests and check reconstruction results by comparing them to the original phantom function. The results demonstrate reasonable accuracy even for high values of \(\mu\). We also show how reconstructions are distorted depending on the incorrect choice of \(\mu\). In the paper we focus on fast and accurate algorithms for the exponential Radon transforms rather than the regularization techniques. Additional regularization methods such as filtering, sparse data representation, or Tikhonov regularization should be considered when dealing with noisy measurements.

The algorithms were implemented by using C++ routines along with MEX interface to MATLAB. For accelerations we used the tools from the Intel Math Kernel Library (MKL), Intel Performance Primitives (IPP), and OpenMP API. The developed software package is publicly available. Performance tests demonstrate speedup more than 100 times compared to the standard discretization of line integrals with cubic interpolation, which was also implemented by using C++ language with Intel high-performance facilities.
5 Main results of the papers

Paper III

This paper deals with the construction of fast methods for evaluation of the hyperbolic Radon transforms, and related applications in reflection seismology. The hyperbolic Radon transform has proven to be a valuable tool for instance in velocity analysis; aliasing and noise removal; trace interpolation; and attenuation of multiple reflections. The hyperbolic Radon transform is defined by

$$\mathcal{R}_h f(\tau, q) = \int_{-\infty}^{\infty} f(\sqrt{\tau^2 + q^2 x^2}, x) dx.$$ 

Here, the function $f$ typically represents seismic common midpoint gathers. The direct approximation of the integral as sums over hyperbolas has a computational complexity of $O(N^3)$, given that the numbers of samples for the variables $t, x, \tau, q$ are $O(N)$. The hyperbolic Radon transform requires additional computations in the time domain, thus it is more challenging to develop a fast evaluation method compared to Radon transforms over straight lines or parabolas.

In this work we propose an algorithm for fast evaluation of the hyperbolic Radon and its adjoint. The approach is similar to the one used for constructing fast algorithms for the standard Radon transform, namely to represent the transform as convolutions in log-polar coordinates and after that rapidly evaluate the convolutions in terms of FFT.

The approach with switching to log-polar coordinates in the case of hyperbolic Radon transform requires some additional steps in comparison to the corresponding approach taken for the standard Radon transform. This is due to the specific coordinate structure and assumptions for the function support. First, specific operations of scaling, rotation and translation have to be performed before convolutions to avoid the origin which is problematic to represent using log-polar coordinates. The obtained geometrical setup is designed with respect to the structure of seismic common midpoint gathers, and for reducing needed sampling rates.

The next issue concerns the structure of hyperbolic coordinates. The convolutions used for computing the log-polar Radon transform can be rapidly evaluated by using FFT if the log-polar samples are given on an equally spaced grid. Since data is assumed to be sampled in the time-offset $(t, x)$ and Radon $(\tau, q)$ domains, a resampling is needed. The resampling is done in terms of
interpolations. We propose to apply interpolation using cardinal B-splines, since this type of interpolation is particularly well-suited for GPU implementations. This technique is similar to that used for USFFT, in the way that the interpolation is conducted by smearing data in one of the domains, and the compensating for that effect is done in the reciprocal domain.

We consider iterative methods for representing CMP gathers by sparse sums of hyperbolic wave events. The sparse representation is used for interpolation, multiple suppression, and for related interpolation and reconstruction techniques. A popular such method is based on using iterative soft thresholding for obtaining sparse representations, which produces solutions $g$ with respect to minimization of

$$ ||\mathcal{R}^*g - f||_2^2 + \mu ||g||_1, $$

where $\mu$ is a parameter that controls the degree of sparsity.

We derive guidelines for how to choose discretization parameters in order to maintain accurate interpolation. The quadratic behavior in the sampling can be fairly well described in terms of the log-polar sampling, as long as the range is not too large. In the case of large ranges, the sampling density has to be increased. We provide an approach with splitting the time-offset and Radon domains into parts and consider the log-polar Radon transform for each of these parts. This allows to sufficiently decrease the number of grid points to apply Fourier transforms. The splitting procedure is not computationally intensive and can be applied several times to achieve needed accuracy.

Accuracy and performance tests were carried out for synthetic CMP gathers, where the error comparisons are done with respect to the direct summation over hyperbolas and with respect to the results from alternative fast methods. The tests show that our method demonstrates the same accuracy level as competing methods, but with lower computational costs. For instance, for large data sizes the developed program on CPU outperforms the fast butterfly algorithm more than 40 times, and more than 1200 times it is faster than the direct summation over hyperbolas. Moreover, the GPU implementation gives additional 14x performance gain compared to the CPU version, yielding a total speed gain of factor 500 compared to the fast butterfly approach, and a factor of 17000 compared to direct evaluation.

Iterative schemes constructed to deal with missed traces, noise and multiple reflections are based on applying the forward and adjoint hyperbolic
Radon transform operators several times. They are well-suited for computing on GPUs due to the almost complete absence of host-device data transfers. The tests show high-quality reconstruction results and demonstrate additional 30% GPU speedup. The algorithms were implemented by using Nvidia CUDA C++ technology with MEX interface to MATLAB. The developed software package is publicly available.

Paper IV

In this paper we focus on the problem of interpolating multicomponent seismic data. The towed multicomponent streamers is a new acquisition technology that allows to measure the crossline and vertical components of water-particle motion in addition to the pressure. We focus on the problem of interpolating towed streamer data where multicomponent measurements exist, and where the crossline direction is sparsely sampled, which is typical for this type of seismic data.

In this work we aim to describe perturbation of sums of plane waves. Assume that data \( u = u(t,x,y) \) is given where \( t \) being the time coordinate, \( x \) being the inline coordinate and \( y \) being the crossline coordinate. In the case where \( u \) locally consists of a sum of less than or equal to two plane waves, the data will satisfy the relation

\[
 u_y + q_t(t,x,y)u_t + q_x(t,x,y)u_x = 0.
\]

In the cases of purely plane waves, the functions \( q_t \) and \( q_x \) will be constant. For typical seismic data there is some (small) curvature present, and this can be captured by having smoothly varying functions \( q_t \) and \( q_x \). The data itself will typically be oscillatory. The main idea here is that instead of interpolating the data directly, one tries to estimate and interpolate the smoothly varying functions \( q_t \) and \( q_x \), and then use these estimates to obtain an interpolation of the actual data by using the partial differential equation above.

The functions \( q_t \) and \( q_x \) can be estimated by using structure tensors. The structure tensor is defined as \( T_α(u)(x) = g_α ∗ (∇u)(∇u)^T(x) \), where \( g_α \) denotes a Gaussian. Owing to the smooth behavior, the structure tensor provides accurate approximation even if only partial information of \( u \) is available. It turns out that this kind of representation is equivalent to a weighted least squares estimation of the gradient. As an alternative approach, we can
try to find \( q_t \) and \( q_x \) by imposing a representation with built-in regularity, i.e., of representations of the form

\[
q_t(t, x, y) = \sum_{n=1}^{N} c_t(n) \phi_n(t, x, y), \quad q_x(t, x, y) = \sum_{n=1}^{N} c_x(n) \phi_n(t, x, y),
\]

for some smooth localized functions \( \phi_n \).

Once estimates for the functions \( q_t \) and \( q_x \) are given, the partial differential equation above (which is a transport equation) can be solved by using the ray tracing method. Considering the ordinary differential equations

\[
\begin{align*}
\frac{\partial \tau}{\partial y}(t, x, y) &= -q_t(\tau(t, x, y), \xi(t, x, y)), \quad \tau(t, x, 0) = t, \\
\frac{\partial \xi}{\partial y}(t, x, y) &= -q_x(\tau(t, x, y), \xi(t, x, y)), \quad \xi(t, x, 0) = x
\end{align*}
\]

we observe that

\[
\frac{\partial u}{\partial y}u(\tau(t, x, y), \xi(t, x, y), y)) = 0,
\]

which means that \( u \) is constant along the rays \((\tau(t, x, y), \xi(t, x, y))\). Using this fact the solution to the partial differential equation above can be computed in two steps: first, the rays \((\tau(t, x, y), \xi(t, x, y))\) are computed by solving the ordinary differential equations, and then the solution \( u \) is computed at an equally spaced grid by interpolating the data along these rays.

Both procedures can be effectively evaluated on GPUs by using CUDA technology and routines from OpenGL. The second procedure is commonly used in computer graphics applications when considering hardware accelerated rendering. The same directionality structure can be used for the interpolation of both the pressure components, and the depth derivative component. This implies that there is no need for measuring the crossline derivate of the depth derivative, which seems to be needed at a first glance. We show and discuss the results of numerical experiments with synthetic and real data sets. The results are quite robust towards measurement noise due to the interpolations with structure tensors and due to diffusion applied for low-frequency components of derivative parts.

The methods were implemented by using Nvidia CUDA Runtime API, OpenGL API with MEX interface to MATLAB.
Paper V

In this paper we consider a parallel algorithm of seismic data decomposition by using redundant basis of wave packets. A main advantage of the decomposition is that functions with wave-like structure, such as seismic CMP gathers, can be well approximated using a linear combination of a small number of these wave packets. Wave-packet representation is a useful tool in seismic data processing, including procedures of data compression, denoising, and interpolation. In this paper we address the problem of developing a fast parallel implementation of 3D wave-packet transform working on GPUs. We consider different optimization techniques for the algorithm structure and for the program implementation on GPU.

A wave-packet representation of a function $f$ has the form

$$f(x) = \sum_{\gamma} c_{\gamma} \phi_{\gamma}(x),$$

where $\phi_{\gamma}(x)$ is a wave packet parameterized by $\gamma$. The parameter $\gamma$ specifies translation, orientation and scaling of a wave packet. The wave packets can be specified by windows functions in the frequency domain and their structure is chosen so that these windows functions form a partition of unity.

Each wave packet has compact (numerical) support in the Fourier domain. The support is contained in a box of prescribed sizes and orientation. By employing Fourier series expansions on each individual box, one can obtain corresponding coefficients $c_{\gamma}$ for data representation. The procedure is done by making use of unequally spaced fast Fourier transforms (USFFT).

The USFFT algorithms are based on smearing data in one of the domains, and the compensating of that effect in the reciprocal domain. There are two kinds of smearing operations when working with the wave-packet decomposition: gathering and scattering. The gathering operation refers to a weighted summation of the values at grid points located not far than a specific distance (radius). The scattering is the adjoint operation to the gathering and refers to a weighted spreading of the value from a specific point to other grid points located not far than the radius.

From the computational point of view, the gathering operations can be computed independently and therefore in parallel for all grid points. However, during the scattering procedure we need to perform writing operations
to the common global grid (associated to the samples of $f(x)$). It is not possible to simultaneously write at the same part of memory, hence the process cannot be parallelized effectively. Instead, we decided to substitute scattering by the gathering operation. This can be done since all rotated and scaled grids for wave packets are still equally spaced when considering independently. The appropriate modification of the algorithm then leads to avoiding the problem of simultaneous writing to memory, which results in an efficient parallel implementation.

We consider optimization details for effective GPU computing. The code was optimized for inner loop operations, mathematical instructions, memory bandwidth, and GPU occupancy. The total speedup of using one GPU Tesla M2010 is approximately equal to 45 when comparing to the sequential program on Intel i7 processor. Moreover, the code was modified for running on several GPUs using OpenMP streams its scalability for large number of GPUs was analyzed. The performance gain for 4 and 8 GPUs is 3.93 and 7.70, respectively. It allows the usage of a large number of GPUs simultaneously for processing large data volumes. The obtained accuracy (5 digits) is acceptable for seismic data analysis.

The program implementation was tested for synthetic 3D seismic data sets in procedures of compression, denoising and interpolation to missed traces. The sparse representation is obtained by storing only a few coefficients of the wave-packet decomposition. The data compression ratio (CR) is defined as a ratio of non-zero wave-packet coefficients to the total number of discrete samples in input data. The performed tests show that the structure of seismic data remains acceptable for analysis when compression ratio is 0.02. Interpolation and denoising procedures are automatically done when recovering data from the coefficients above a chosen thresholding level.

The program was implemented by using Nvidia Cuda C++ technology and tested in Madagascar software package [7] for seismic data processing.
References


Abstract

The Radon transform and its adjoint, the back-projection operator, can both be expressed as convolutions in log-polar coordinates. Hence, fast algorithms for the application of these operators can be constructed by using FFT, if data is resampled at log-polar coordinates. Radon data is typically measured on an equally spaced grid in polar coordinates, and reconstructions are represented (as images) in Cartesian coordinates. Therefore, in addition to FFT, several steps of interpolation have to be conducted in order to apply the Radon transform and the back-projection operator by means of convolutions. However, in comparison to the interpolation conducted in Fourier-based gridding methods, the interpolation performed in the Radon and image domains will typically deal with functions that are substantially less oscillatory. Reasonable reconstruction results can thus be expected using interpolation schemes of moderate order. It also provides better control over the artifacts that can appear due to measurement errors.

Both the interpolation and the FFT operations can be efficiently implemented on Graphical Processor Units (GPUs). For the interpolation, it is possible to make use of the fact that linear interpolation is hard-wired on GPUs, meaning that it has the same computational cost as direct memory access. Cubic order interpolation schemes can be constructed by combining linear interpolation steps and this provides important computation speedup.

We provide details about how the Radon transform and back-projection can be implemented efficiently as convolution operators on GPUs. For large data sizes, these algorithms are several times faster than those of other software packages based on GPU implementations of the Radon transform and the back-projection operator. Moreover, the gain in computational speed is substantially higher when comparing against other CPU-based algorithms.
1 Introduction

The two-dimensional Radon transform is the mapping of functions to their line integrals, i.e., a mapping $\mathcal{R} : \mathbb{R}^2 \rightarrow S^1 \times \mathbb{R}$, where $S^1$ denotes the unit circle, defined by

$$\mathcal{R} f(\theta, s) = \int f(x) \delta(x \cdot \theta - s) \, dx.$$  

(1)

The parameter $\theta$ represents the (normal) direction of the lines, and the parameter $s$ denotes the (signed) distance of the line to the origin. It is customary to use $\theta$ both as a point on the unit sphere and as an angle; i.e., the notation $x \cdot \theta$ is used to parameterize lines in $\mathbb{R}^2$ by the relation $x \cdot \theta = x_1 \cos(\theta) + x_2 \sin(\theta)$. Note that each line is defined twice in this definition, since $s$ can take both positive and negative values, and since $\theta \in S^1$.

A schematic illustration of the Radon transform is given in Figure 1, where beams are propagating through an object and after absorption are measured by receivers. The Radon transform appears, for instance, in computational tomography (CT). The tomographic inversion problem lies in recovering an unknown function $f$ given knowledge of $\mathcal{R} f$. For more details about CT, see [18, 21, 26, 30].

One of the most popular methods of inverting the usual Radon transform is by means of the filtered back-projection (FBP) method [30]. It uses the inversion formula

$$f = \mathcal{R}^\# \mathcal{W} \mathcal{R} f.$$  

(2)

where $\mathcal{W}$ is a convolution operator acting only on the $s$-variable, and where $\mathcal{R}^\# : S^1 \times \mathbb{R} \rightarrow \mathbb{R}^2$ denotes the back-projection operator, which integrates over all lines through a point, i.e.,

$$\mathcal{R}^\# g(x) = \int_{S^1} g(\theta, x \cdot \theta).$$

The back-projection operator is adjoint to the Radon transform. The convolution operator $\mathcal{W}$ can be described either as a Hilbert transform followed by a derivation, both with respect to the variable $s$, or as a convolution operator with a transfer function being a suitably scaled version of $|\sigma|$, where $\sigma$ denotes the conjugate variable of $s$. 
Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators

Figure 1: Scheme of computing projections for a given angle $\theta$.

A direct implementation of the FBP has a time complexity of $N^3$, if we assume that reconstructions are made on an $N \times N$ lattice and that the numbers of samples in $s$ and $\theta$ are both $O(N)$. This is because for each point $x$, integration has to be made over all lines ($N$ directions) passing through that point. However, there are fast methods for computing the back-projection. For instance, the method of fast hierarchical filtered back-projection (FHBP) introduced by Bresler and colleagues [7, 22] is based on a hierarchical decomposition of the Radon transform. In this way the total time can be reduced, but at a cost of decreased accuracy; cf. [7]. The technique proposed in [12, 13] is also based on a hierarchical scheme. A speedup is obtained in this case by viewing the back-projection as a line integration in the Radon domain, where longer line integrals are recursively formed from shorter ones.

Another class of fast methods for inversion of Radon data goes via the Fourier-slice theorem. These methods are based on the Fourier slice theorem which relates one-dimensional Fourier transform of the Radon data with a two-dimensional Fourier transform of the reconstruction image. The sampling
Figure 2: The (modified) Shepp-Logan phantom (a) and its Radon transform (b). The panel (d) shows the two-dimensional Fourier transform of the phantom. The regions indicated by the squares in (b) and (d) are shown in higher resolution in (c).

of the Radon data gives rise to a nonstandard two-dimensional sampling in the frequency domain that cannot be computed directly by using standard FFT. To circumvent this problem, algorithms for unequally spaced FFT (USFFT, also known under the acronyms NFFT and NUFFT) can be used [8, 9, 20, 40]. In the standard parallel beam case, the Radon data can be mapped onto data on a polar grid. As an alternative, pseudopolar sampling can be used to construct fast Fourier reconstructions by making use of chirp-Z transforms; see, for instance, [3, 4, 5]. The data in the frequency domain will typically be oscillatory, as seen in Figure 2d) and (at in increased resolution) in the lower
Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators

right panel of Figure 2c). Hence, in order to interpolate data of this type high interpolation order is required. In comparison, the data in the Radon domain will not be particularly oscillatory. This is illustrated in Figure 2b) and the upper left panel of Figure 2c). This fact implies that interpolation methods of moderate order can be expected to produce reasonable results. This means in turn that less time needs to be spent on conducting interpolation. It also gives more control over the interpolation errors, as local errors will be kept local in the Radon domain (and hence more easily distinguishable as artifacts in the reconstruction).

In this paper, we discuss how to design fast algorithms for the application of the Radon transform and the back-projection operator by using the fact that they can be expressed in terms of convolutions when represented in log-polar coordinates [2, 17, 25, 43]. In particular, we follow the approach suggested in [2]. This formulation turns out to be particularly well-suited for implementation on GPUs. A major advantage with using GPUs is that the routines for linear interpolation are fast. In fact, the cost for computing linear interpolation is the same as for reading directly from memory [23]. This feature can be utilized for constructing fast interpolation schemes. In particular, in this paper we will work with cubic interpolation on GPU [35, 39].

For the sake of comparison, we will provide performance and accuracy tests of the proposed method along with comparisons against other software packages for tomographic computations. We also conduct a performance comparison between the different methods as utilized in iterative reconstruction techniques. The iterative methods rely on applying the Radon transform and the back-projection operator several times. An advantage of keeping all computations on the GPU is that the needed time for CPU-GPU memory transfer can be reduced.
2 The Radon transform and the back-projection expressed as convolutions

We recapitulate some of the main ideas of the method described in [2]. A key part there is the usage of log-polar coordinates, i.e.,

\[
\begin{cases}
  x_1 = e^\rho \cos(\theta), \\
  x_2 = e^\rho \sin(\theta),
\end{cases}
\]

where \(-\pi < \theta < \pi\). To simplify the presentation, we identify \(f(\theta, \rho)\) with \(f(x_1, x_2)\) if \((\theta, \rho)\) in the log-polar coordinate system corresponds to the point \(x = (x_1, x_2)\) of the Cartesian coordinate system, and similarly for other coordinate transformations.

By representing the distance between lines and the origin \(s = e^\rho\), and by a change of variables in (1) from Cartesian to log-polar coordinates the log-polar Radon transform can be expressed as

\[
\mathcal{R}_{lp} f(\theta, \rho) = \int_{-\pi}^{\pi} \int_{-\infty}^{\infty} f(\theta', \rho') e^\rho \mathcal{D} \left( \cos(\theta - \theta') - e^{\rho - \rho'} \right) d\rho' d\theta'
\]

\[
= \int_{-\pi}^{\pi} \int_{-\infty}^{\infty} f(\theta', \rho') e^\rho \zeta(\theta - \theta', \rho - \rho') d\rho' d\theta',
\]

where \(\zeta(\theta, \rho) = \mathcal{D}(\cos(\theta) - e^\rho)\). In particular, \(\mathcal{R} f(\theta, s) = \mathcal{R}_{lp} f(\theta, \log(s))\) for \(s > 0\) (which is sufficient since the information for \(s < 0\) is redundant).

We briefly mention how to put this formula into a theoretical framework. Set \(S = (-\pi, \pi) \times \mathbb{R}\) and note that for a compactly supported smooth function \(h\) on \(S\) we have

\[
\int_{-\pi}^{\pi} \int_{-\infty}^{\infty} h(\theta, \rho) \zeta(\theta, \rho) d\rho d\theta = \int_{-\pi/2}^{\pi/2} h(\theta, \log(\cos(\theta))) d\theta,
\]

which can be written as \(\int_S h d\mu\) where \(\mu\) is an infinite measure on \(S\). Hence, the formula extends by continuity to, e.g., all continuous compactly supported functions \(h\) in \(S\). It follows that (3) is well-defined whenever \(f\) is a continuous compactly supported function which is zero in a neighborhood of 0.
As the Radon transform in the coordinate system \((\theta, \rho)\) is essentially a convolution between \(f\) and the distribution \(\zeta(\theta, \rho)\), it can be rapidly computed by means of Fourier transforms. Special care has to be taken with the distribution \(\zeta\), an issue we will return to in what follows. Ignoring possible difficulties with the distribution \(\zeta\), let us discuss how (3) can be realized by using FFTs. It is natural to assume that the function \(f\) has compact support (the object that is measured has to fit in the device that is measuring it). The compact support also implies that the Radon transform of \(f\) will have compact support in the \(s\)-variable. However, this is not true in the log-polar setting, since \(\rho \to -\infty\) as \(s \to 0\).

Note also that for any point \(x\) in the plane there is a direction for which there is a line passing through \(x\) and the origin. This implies that it is not possible to approximate the values of the Radon transform by using a finite convolution in log-polar coordinates if it is to be computed for all possible line directions. However, by restricting the values of \(\theta\), and by making a translation so that the support of \(f\) is moved away from the origin, it is in fact possible to describe the partial Radon transform as a finite convolution, and then recover the full Radon transform by adding the contributions from various partial Radon transforms. The setup is illustrated in Figure 3.

Figure 3a) schematically shows the parameters used by the proposed method. Assume that the function we are considering has support in the unit circle. Let \(\beta\) be a fixed angle and translate and scale \(f\) according to Figure 3a). The new function is now supported in a circle with radius

\[
a_R = \frac{\sin\left(\frac{\beta}{2}\right)}{1 + \sin\left(\frac{\beta}{2}\right)}.
\]

Moreover, the tangent lines \(L_1\) and \(L_2\) are chosen to be perpendicular to the straight lines determining the (symmetric) angle span. Let \(a_r\) be the shortest distance from the center \(O_0\) (in the translated coordinates) to the tangent lines. From the geometry, it is readily verified that

\[
a_r = \frac{\cos\left(\frac{\beta}{2}\right) - \sin\left(\frac{\beta}{2}\right)}{1 + \sin\left(\frac{\beta}{2}\right)}.
\]

Assume for the moment that \(f\) has support in the gray circle indicated in Figure 3a). In log-polar coordinates \((\theta, \rho)\) it then has support inside of
Figure 3: (a) Tangent lines to the circle to determine the support of the log-polar Radon transform function; (b) three angle spans to compute partial Radon transforms.

If we restrict our attention to values of $\mathcal{R}_{lp}(f)$ in the same sector $\theta \in \left[-\frac{\beta}{2}, \frac{\beta}{2}\right]$, then the only nonzero values of $\mathcal{R}_{lp}(f)$ will be for $\rho$ in the interval $[\log a_r, 0]$. This means that $\mathcal{R}_{lp}(f)(\theta, \rho)$ values can be computed by the finite convolution

$$
\int_{-\beta/2}^{\beta/2} \int_{\log(1-2a_R)}^{0} f(\theta', \rho')e^{\rho' \zeta} (\theta - \theta', \rho - \rho') d\theta' d\rho', 
$$

(4)

where $(\theta, \rho) \in \left[-\frac{\beta}{2}, \frac{\beta}{2}\right] \times [\log a_r, 0]$. We now replace the integral (4) by the periodic convolution

$$
\mathcal{R}_{lp}^P f(\theta, \rho) = \int_{-\beta}^{\beta} \int_{\log(a_r)}^{0} f(\theta', \rho')e^{\rho' \zeta_{\text{per}}} (\theta - \theta', \rho - \rho') d\theta' d\rho',
$$

(5)

where $\zeta_{\text{per}}$ is the periodic extension of $\zeta$ defined on $[-\beta, \beta] \times [\log(a_r), 0]$. It is readily verified that for $(\theta, \rho) \in \left[-\frac{\beta}{2}, \frac{\beta}{2}\right] \times [\log a_r, 0]$, it thus holds that

$$
\mathcal{R}_{lp}^P f(\theta, \rho) = \mathcal{R} f(\theta, e^\rho).
$$

42
Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators

![Figure 4](image_url)

Figure 4: The effect on the support due to convolutions: a) Radon transform; b) back-projection.

We refer to $R_{lp}$ as the partial log-polar Radon transform.

Note that, in analogy with the argument following (3), the formula (4) can be written as a convolution between $f(\theta', \rho')e^{\rho'}$ and a finite measure, whereas (5) can be written as a convolution with a locally finite periodic measure. The above formulas are thus well defined as long as $f$ is continuous (or even piecewise continuous) in the log-polar coordinates. We refer to [46, Chapter 11], for basic results about convolution between functions and periodic distributions.

The convolution setup for the Radon transform is depicted in Figure 4a). The rightmost black solid curve $C$ shows $\rho = \log(\cos(\theta))$, $(-\beta \leq \theta \leq \beta)$, which is the support of $\zeta$ in $[-\beta, \beta] \times [\log a_r, 0]$. Let $D$ denote the circle,

$$D = \{(x_1, x_2) : (x_1 - 1 + a_R)^2 + x_2^2 < a_R^2\}.$$ 

The black dots show the perimeter of $D$ (in log-polar coordinates), and the gray curves indicate translation of the curve $C$ associated with the black dots.
Paper I on the circle, within the interval \([-\beta, \beta]\). There is a difference in grayscale for the points with \(\theta\) inside the range \([-\frac{\beta}{2}, \frac{\beta}{2}]\), as only these values are of interest to us. Note that the smallest \(\rho\)-value of the contributing part in this interval is \(\log(a_r)\). Moreover, the red lines indicate parts of the translations of \(C\) outside \([-\beta, \beta] \times [\log(a_r), 0]\), and the blue lines show how these curves are wrapped back into the domain \([-\beta, \beta] \times [\log(a_r), 0]\) by the periodic extension of \(\zeta\). We see that these effects do not have any influence on the domain \([-\frac{\beta}{2}, \frac{\beta}{2}] \times [\log(a_r), 0]\).

We now describe how \(\mathcal{R}_{lp}^{\beta}\) can be used to recover \(\mathcal{R}f\) for a function \(f\) with support in the unit circle. We split the angular variable into \(M\) different parts, and choose

\[
\beta = \frac{\pi}{M}.
\] (6)

For \(m = 0, 1, \ldots, M - 1\) let \(T_m : \mathbb{R}^2 \to \mathbb{R}^2\) denote the change of coordinates

\[
T_m(x) = a_R \begin{pmatrix}
\cos(m\beta) & \sin(m\beta) \\
-\sin(m\beta) & \cos(m\beta)
\end{pmatrix} \begin{pmatrix}
x_1 \\
x_2
\end{pmatrix} + \begin{pmatrix}
1 - a_R \\
0
\end{pmatrix},
\]

and let \(T_m f = f(T_m^{-1} x)\). Note that

\[
\mathcal{R} f(\theta, s) = a_R^{-1} \mathcal{R} (T_m f) \left( \theta - m\beta, \log(a_R s + (1-a_R)\cos(\theta - m\beta)) \right). \tag{7}
\]

The Radon transform can thus be computed for arbitrary \(\theta\) and \(0 < s < 1\), by using the relation

\[
\mathcal{R} f(\theta, s) = a_R^{-1} \mathcal{R}_{lp}^{\beta} (T_m f) \left( \theta - m\beta, \log(a_R s + (1-a_R)\cos(\theta - m\beta)) \right), \tag{8}
\]

where \(m = \lfloor \theta / \beta \rfloor\) and \([x]\) denotes the rounding operator to the closest integer to \(x\), and where \(\mod\) denotes the modulus operator.

We denote the change of coordinates above by

\[
S_m(\theta, s) = \left( \theta - m\beta, \log(a_R s + (1-a_R)\cos(\theta - m\beta)) \right),
\]

and we then have that

\[
\mathcal{R} f(\theta, s) = a_R^{-1} \mathcal{R}_{lp}^{\beta} (T_m f) (S_m(\theta, s)).
\]
Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators

We remark that for fixed $\theta$, the connection (7) between the Radon data in the different domains has an affine dependence on $s$. Since the filter operator $\mathcal{W}$ acts as a convolution operator with regard to $s$, its action will in principle be the same regardless of whether the coordinate transformation $T_m$ is used or not. We use the notation $\mathcal{W}_{lp}$ to denote the action of the filter operator in log-polar coordinates.

The adjoint operator (back-projection) associated with the Radon transform (1) can be written as

$$\mathcal{R}^#g(x) = \int_{-\infty}^{\infty} \int_{S^1} g(\theta, s) \delta(x \cdot \theta - s) \, d\theta \, ds,$$

cf. [30]. It is a weighted integral of $g$ over lines passing through the point $x$, and just as for the Radon transform it can be expressed as a convolution in log-polar coordinates. We define

$$\mathcal{R}_{lp}^#g(\theta, \rho) = \int_{-\infty}^{\infty} \int_{-\pi}^{\pi} g(\theta', \rho') \delta(e^{\rho - \rho'} \cos(\theta - \theta') - 1) \, d\theta' \, d\rho'
= \int_{-\infty}^{\infty} \int_{-\pi}^{\pi} g(\theta', \rho') \zeta^#(\rho - \rho', \theta - \theta') \, d\theta' \, d\rho'.$$

It is shown in [2] that $\mathcal{R}^#f(x) = 2\mathcal{R}_{lp}^#f(\theta, \rho)$, where the factor two comes from the fact that the corresponding integration in the polar representation $(\theta, s)$ is only done in the half-plane $s > 0$. The log-polar back-projection operator has the same problem as the log-polar Radon transform in dealing with $s = 0$, and in a similar fashion we make use of partial back-projections in order to avoid this problem. Because of the relation (8) and the fact that the filter operator $\mathcal{W}$ can be applied to each partial Radon data individually, it will be enough to consider partial back-projections for Radon data with $\theta \in \left[-\frac{\beta}{2} + m\beta, \frac{\beta}{2} + m\beta\right]$ according to the setup of Figure 3. By applying $T_m^{-1}$ to each of the partial back-projections, and summing up the results, we will recover the original function. For detailed calculations, we refer to [2].

The idea is thus to split the Radon data into $M$ parts, where each part is transformed according to Figure 3a). For each part, the filtered data is back-
projected according to
\[ \int_{-\infty}^{\infty} \int_{-\frac{\beta}{2}}^{\frac{\beta}{2}} g(\theta', \rho') \zeta^#(\rho - \rho', \theta - \theta') \, d\theta' \, d\rho'. \]

Since we assumed that our original function had support inside the unit circle, we are interested only in the contributions inside the disc \( D \). Since only lines with \( \rho \in [\log(a_r), 0] \) go through this circle, the integration in the \( \rho \) variable above can be limited to \( \rho \in [\log(a_r), 0] \). As for the Radon transform, we now want to write this (finite) convolution as a periodic convolution. Figure 4b) illustrates how this can be achieved. The black solid lines show translations of the curve \( \rho = -\log(\cos(\theta)) \) representing the back-projection integral in the log-polar coordinates. The black dots now show the perimeter of a support of the Radon data, indicated by dark gray in the left illustration of Figure 4. Here, the dark gray curves of the back-projection illustration show the translations of \( \rho = -\log(\cos(\theta)) \) that will give a contribution inside the disc \( D \). The light gray curves illustrate contributions that fall outside the support of \( D \). The red curves show contributions that will fall outside the range \( [-\beta, \beta] \times [\log(a_r), 0] \), and the blue curves show the effect when these lines are wrapped back into the domain \( [-\beta, \beta] \times [\log(a_r), 0] \). We note that the blue curves do not intersect the circle. Hence, we define the partial log-polar back-projection operator as

\[ \mathcal{R}^p \# g(\theta, \rho) = \int_{\log(a_r)}^{0} \int_{-\frac{\beta}{2}}^{\frac{\beta}{2}} g(\theta', \rho') \zeta^#_{\text{per}}(\rho - \rho', \theta - \theta') \, d\theta' \, d\rho'. \] (9)

where \( \zeta^#_{\text{per}} \) is the periodic extension of \( \zeta^# \) defined on \( [-\beta, \beta] \times [\log(a_r), 0] \), and note that for \((\theta, \rho)\) corresponding to points inside the domain \( D \), it holds that

\[ \mathcal{R}^p \# g(\theta, \rho) = \mathcal{R}^p g(\theta, \rho). \]

Formula (2) can now be recast as

\[ 2 \sum_{m=0}^{M-1} T_m^{-1} \mathcal{R}^p \# \mathcal{W}^p \mathcal{R}^p T_m f(x) = f(x) \]

for all \( x \) in the unit disc.
3 Fast evaluation of the log-polar Radon transform and the log-polar back-projection

Let $h$ be a continuous function on some rectangle $R$ in $\mathbb{R}^2$ with area $A_R$, and let $\mu$ be a finite measure on $R$, and let $\mu_{\text{per}}$ be its periodic extension. Along the same lines as [46, Theorem 11.6-3], it is easy to see that the (normalized) Fourier coefficients of their periodic convolution satisfy

$$\hat{h} \ast \mu_{\text{per}} = A_R \hat{h} \hat{\mu},$$

where $\hat{h} \hat{\mu}$ is the pointwise multiplication of the respective Fourier coefficients with respect to $R$. We will use this formula and FFT to fast evaluate $\mathcal{R}_{\text{lp}}$ (5) and $\mathcal{R}_{\text{lp}}^\#$ (9). Note that the area of $[-\beta, \beta] \times [\log a, 0]$ is $-2\beta \log a_r$.

For $(\theta, \rho)$ in this rectangle we then have

$$f(\theta, \rho) = \sum_{k, \theta, k, \rho} \hat{f}_{k, \theta, k, \rho} e^{2\pi i \left( \frac{\theta k}{\beta} + \frac{\rho}{\log a_r} \right)},$$

$$g(\theta, \rho) = \sum_{k, \theta, k, \rho} \hat{g}_{k, \theta, k, \rho} e^{2\pi i \left( \frac{\theta k}{\beta} + \frac{\rho}{\log a_r} \right)},$$

$$\zeta(\theta, \rho) = \sum_{k, \theta, k, \rho} \hat{\zeta}_{k, \theta, k, \rho} e^{2\pi i \left( \frac{\theta k}{\beta} + \frac{\rho}{\log a_r} \right)},$$

$$\zeta^\#(\theta, \rho) = \sum_{k, \theta, k, \rho} \hat{\zeta}_{k, \theta, k, \rho} e^{2\pi i \left( \frac{\theta k}{\beta} + \frac{\rho}{\log a_r} \right)}.$$

The Fourier coefficients for the two distributions $\zeta$ and $\zeta^\#$ are given by

$$\zeta_{k, \theta, k, \rho} = \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} \int_{0}^{\log(a_r)} \delta(\cos(\theta) - e^\rho) e^{-2\pi i \frac{\theta k}{\beta}} e^{-2\pi i \frac{\rho}{\log a_r}} d\rho d\theta =$$

$$= \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} \int_{0}^{\log(a_r)} \delta(\cos(\theta) - e^\rho) e^{-2\pi i \frac{\theta k}{\beta}} (e^\rho)^{-2\pi i \frac{k}{\log a_r}} d\rho d\theta =$$

$$= \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} e^{-2\pi i \frac{\theta k}{\beta}} (\cos(\theta))^{-2\pi i \frac{k}{\log a_r}} d\theta,$$  \hspace{1cm} (12)
\[ \zeta_{k_\theta,k_\rho} = \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} \int_{0}^{-\log(a_r)} \delta(e^\rho \cos(\theta) - 1) e^{-2\pi i \frac{\theta}{k_\theta}} e^{-2\pi i \frac{\rho}{\log(a_r)}} d\rho d\theta = \]
\[ \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} \int_{0}^{-\log(a_r)} \delta(e^\rho \cos(\theta) - 1) e^{-2\pi i \frac{\theta}{k_\theta}} (e^\rho)^{-2\pi i \frac{\rho}{\log(a_r)}} d\rho d\theta = \]
\[ \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} e^{-2\pi i \frac{\theta}{k_\theta}} (\cos(\theta))^{-2\pi i \frac{\rho}{\log(a_r)}} d\theta. \] (13)

Both the integrals on the right hand sides of (12) and (13) are of the form

\[ P(\mu, \alpha, \beta) = \int_{-\beta}^{\beta} e^{i \mu \theta} \cos(\alpha) d\theta, \] (14)

where \( \mu = -2\pi i \frac{k_\theta}{2\beta}, \alpha = -2\pi i \frac{k_\rho}{\log(a_r)} - 1 \) for (12) and \( \alpha = -2\pi i \frac{k_\rho}{\log(a_r)} \) for (13).

We briefly describe how to evaluate (14) numerically. The integral is well suited for evaluation by FFT, since for a fixed value of \( k_\rho \) (fixed \( \alpha \)) we can obtain \( \zeta_{k_\theta,k_\rho} \) and \( \zeta_{k_\theta,k_\rho}^\# \) for all integers \( k_\theta \) in a given range by evaluating the integral of (14) by the trapezoidal rule by means of FFT. However, for this procedure to be accurate, we need to oversample the integral and make use of end-point corrections. The function \( \cos(\alpha) \) will be oscillatory, but the oscillation is determined by the fixed parameter \( \alpha \). Neglecting boundary effects, we can therefore expect the trapezoidal rule to be efficient, provided that sufficient oversampling is used.

For the boundary effects, we can make use of end-point correction schemes [1]. For the computations used in this paper, we have used an oversampling factor of 8 and an eight-order end-point correction with weights

\[ 1 + \frac{1}{120960} \left[ -23681, 55688, -66109, 57024, -31523, 9976, -1375 \right]. \]

Suppose next that we only know values of \( f \) and \( g \) in (5) and (9), respectively, on an equally spaced sampling covering \([-\beta, \beta] \times [\log(a_r),0]\); i.e., \( f \)
and $g$ are known on the lattice

\[
\left\{ 2\pi \frac{j_\theta}{N_\theta}, \log(a_r) \left(1 - \frac{j_\rho}{N_\rho}\right) \right\}, \quad -\frac{N_\theta}{2M} \leq j_\theta < \frac{N_\theta}{2M}, \quad 0 \leq j_\rho < N_\rho,
\]

(recall that $\beta = \pi/M$ by (6)). We denote these values by $f_{j_\theta,j_\rho}$ and $g_{j_\theta,j_\rho}$, respectively. In order for (5) and (9) to be meaningful, we need to have continuous representations of $f$ and $g$. This is particularly important as $\zeta$ and $\zeta^#$ are distributions. A natural way to do this, is to define $\hat{f}_{k_\theta,k_\rho}$ and $\hat{g}_{k_\theta,k_\rho}$ by the discrete Fourier transform of $f_{j_\theta,j_\rho}$ and $g_{j_\theta,j_\rho}$; i.e., let

\[
\hat{f}_{k_\theta,k_\rho} = \begin{cases} \frac{M}{N_\rho N_\theta} \sum_{j_\theta = -\frac{N_\theta}{2M}}^{N_\theta-1} \sum_{j_\rho = 0}^{N_\rho-1} f_{j_\theta,j_\rho} e^{-2\pi i \left(\frac{j_\theta k_\theta}{N_\theta} + \frac{j_\rho k_\rho}{N_\rho}\right)}, & \text{if } -\frac{N_\theta}{2M} \leq k_\theta < \frac{N_\theta}{2M} \quad 0 \leq k_\rho < N_\rho, \\ 0, & \text{otherwise}, \end{cases}
\]

and

\[
\hat{g}_{k_\theta,k_\rho} = \begin{cases} \frac{M}{N_\rho N_\theta} \sum_{j_\theta = -\frac{N_\theta}{2M}}^{N_\theta-1} \sum_{j_\rho = 0}^{N_\rho-1} g_{j_\theta,j_\rho} e^{-2\pi i \left(\frac{j_\theta k_\theta}{N_\theta} + \frac{j_\rho k_\rho}{N_\rho}\right)}, & \text{if } -\frac{N_\theta}{2M} \leq k_\theta < \frac{N_\theta}{2M} \quad 0 \leq k_\rho < N_\rho, \\ 0, & \text{otherwise}. \end{cases}
\]

We can then use (10) and (11) to define continuous representations of $f$ and $g$. Using this approach, the values of (5) and (9) are also well defined, namely

\[
\mathcal{R}_\theta^p f(\theta, \rho) = -2\beta \log(a_r) \sum_{k_\theta,k_\rho} \hat{f}_{k_\theta,k_\rho} \bar{\zeta}_{k_\theta,k_\rho} e^{2\pi i \left(\frac{\rho k_\theta}{2M} + \frac{\rho k_\rho}{N_\rho}\right)},
\]

and

\[
\mathcal{R}_\theta^g g(\theta, \rho) = -2\beta \log(a_r) \sum_{k_\theta,k_\rho} \hat{g}_{k_\theta,k_\rho} \bar{\zeta}^#_{k_\theta,k_\rho} e^{2\pi i \left(\frac{\rho k_\theta}{2M} + \frac{\rho k_\rho}{N_\rho}\right)}.
\]

In the case where the two transforms are to be evaluated at $(\theta, \rho)$ on the lattice (15), the corresponding sums above can be rapidly evaluated by using FFT.
4 Sampling rates

There are three different cases for which it is natural to use equally spaced discretization: For the representation of $f$ in Cartesian coordinates $(x_1, x_2)$ covering the unit circle $S^1$, for the polar representation $(\theta, s)$ of the sinograms $R(f)$, and for the log-polar coordinates $(\theta, \rho)$ for evaluation of $R_{lp}$ and $R_{lp}^\dagger$.

We will use a rectangular grid in all three coordinate systems and interpolate data between them as we work with the different domains.

In this section we derive guidelines for how to choose discretization parameters. These will be based on the assumption that $\hat{f}$ is “essentially supported” in a disc with radius $N/2$, in the sense that contributions from the complement of this disc can be ignored without affecting the computational accuracy. Due to the uncertainty principle, $\hat{f}$ cannot have its support included in this disc, since $f$ itself is supported in a disc with radius $1/2$, but it may work quite well in practice and is, therefore, still convenient to use for deriving sampling rates.\footnote{More strict results can be obtained by introducing notions such as numerical support. Recall that any compactly supported $L^1$ function has compact numerical support in both $x$ and $\xi$ by the Riemann-Lebesgue lemma. For instance, Gaussians have (small) compact numerical support in both the spatial and the frequency domains. The presentation will quickly become substantially more technical and we therefore follow the signal processing practice and use Nyquist sample rates despite not having infinitely long samples.}

We also base our arguments on refinements of the Nyquist sampling rate, more precisely the Paley-Wiener-Levinson theorem.

In the spatial variables $(x_1, x_2)$, the Nyquist sampling rate (corresponding to the assumption on the support of $\hat{f}$) is $1/N$. This leads us to cover the unit circle with a grid of size $N \times N$. We use

$$X = \left\{ \frac{j_1}{N}, \frac{j_2}{N} \right\}, \quad -\frac{N}{2} \leq j_1, j_2 < \frac{N}{2}. \quad (18)$$

To derive recommendations for the sampling in the polar coordinates $(\theta, s)$ (for the entire $R f$), recall the Fourier slice theorem, which states that

$$\int_{-\infty}^{\infty} R f(s, \theta) e^{-2\pi i s \sigma} \, ds = \int_{-\infty}^{\infty} f(s \theta + t \theta^\perp) e^{-2\pi i s \sigma} \, ds \, dt =$$

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x) e^{-2\pi i (\sigma \theta \cdot x)} \, ds \, dt = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x) e^{-2\pi i (\sigma \theta \cdot x)} \, ds \, dt = \hat{f}(\sigma \theta).$$

\footnote{More strict results can be obtained by introducing notions such as numerical support.}
Thus $\mathcal{R} f(\theta, s) = \mathcal{F}^{-1}_{\sigma \rightarrow _s}(\hat{f}(\sigma \theta))$, and hence the Nyquist sampling rate for $s$ is $\Delta s = \frac{1}{N}$, yielding that $N_s = N$. Let $\Delta \theta_p$ denote the angular sampling rate in the polar coordinate system. Since $f$ is supported in the unit disc, the Nyquist sampling rate in the frequency domain $\xi$ is 1 (on a rectangular infinite grid). However, the multidimensional version of the Paley-Wiener-Levinson theorem roughly says that it is sufficient to consider an irregular set of sample points whose maximal internal distance between neighbors is 1.

Since we have assumed that the values of $\hat{f}(\xi)$ for $|\xi| > \frac{N}{2}$ are negligible, this leads us to choose $\Delta \theta_p$ and $\Delta \sigma$ (the latter will not be used), so that the polar grid points inside this circle has a maximum distance of 1. It follows that

$$\Delta \theta_p = \frac{2}{N}.$$ 

Since we want to cover an angle span of $[0, \pi]$, this leads to $N_\theta \approx \frac{\pi}{2} N$. We denote the polar grid by

$$\Sigma = \{ j_\theta \Delta \theta_p, j_s \Delta s \}, \quad 0 \leq j_\theta < N_\theta, \quad -\frac{N}{2} \leq j_s < \frac{N}{2}. \quad (19)$$

Practical tomographic measurements can therefore typically have a ratio $\frac{N_\theta}{N_s} \approx 1.5$ between the sampling rates in the $\theta$- and $s$-variables. We refer to [26, 30] for more details, and proceed to discuss the sampling in the log-polar coordinates.

To distinguish between the coordinate sampling parameters we use $\Delta \theta_{lp}$ for angular sampling rate in the log-polar coordinates. Recall that we wish to accurately represent functions of the form $T_m f$, which is a rotation, dilation (with factor $2aR$) and translation of $f$. The support of $T_m f$ lies in the gray circle of Figure 3 and its essential frequency support is then inside the disc of radius $\frac{N}{4aR}$.

Figure 5 illustrates the setup where the black lines indicate equally spaced samples of $\theta_{lp}$; the blue curves indicate equally spaced sampling in $s_{lp}$, and the red dashed curves indicate equally spaced sampling in $\rho$. Note that the maximum distance between points in the $x_1$ direction occurs when $s = 1$ or $\rho = 0$. It is thus clear that $\Delta s_{lp}$ should equal $\Delta x_1 = \frac{2aR}{N}$, whereas $\Delta \rho$ is
Figure 5: (a) Log-polar grid. Samples in the $\rho$-variable are chosen in order not to lose the accuracy of measuring data. (b) The bandwidth of a partial back-projection function.

determined by

$$\max_{\rho \in [\log(a_r), 0]} e^\rho - e^{\rho - \Delta \rho} = \max_{\rho \in [\log(a_r), 0]} e^\rho \left(1 - e^{-\Delta \rho}\right) \leq \Delta x_1.$$  

As the largest distance occurs when $\rho = 0$, it follows that

$$\Delta \rho \leq -\log \left(1 - \frac{2a_R}{N}\right),$$

and consequently, since the total distance that is to be covered is $-\log(a_r)$, that

$$N_\rho \geq \left\lceil \frac{\log(a_r)}{\log(1 - \frac{2a_R}{N})} \right\rceil. \tag{20}$$

where the notation $\lceil x \rceil$ denotes the nearest integer greater than or equal to $x$.

For the determination of sample rate in the angular variable for the representation of $T_m f$ in the disc $D$, we have that

$$\Delta \theta_{lp} \approx \sin(\Delta \theta_{lp}) \leq \frac{2a_R}{N},$$
Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators (see Figure 5). Hence, we introduce

$$\Omega_{lp} = \left\{ j_\theta \Delta \theta_{lp}, j_\rho \Delta \rho \right\}, \quad -\frac{\beta}{2 \Delta \theta_{lp}} \leq j_\theta < \frac{\beta}{2 \Delta \theta_{lp}}, \quad -N_\rho < j_\rho \leq 0,$$

(21)

for the representation of \( T_m f \) in \( D \). The representation of the partial Radon transform of \( T_m f \) needs a reduced sample rate compared to (21), since the original Radon transform is sampled at \( \Delta \theta_p \). As the partial Radon transform is applied as a convolution on a log-polar grid (i.e. by a Fourier multiplier), the higher frequencies in the \( \theta \)-direction will not be needed. Hence, we can apply a low-pass filter in the \( \theta \)-direction and apply the FFT operation on the grid

$$\Omega_p = \left\{ j_\theta \Delta \theta_p, j_\rho \Delta \rho \right\}, \quad -\frac{N_\theta}{2M} \leq j_\theta < \frac{N_\theta}{2M}, \quad N_\rho < j_\rho \leq 0,$$

(22)

for the computation of the partial Radon transform of \( T_m f \). The grid \( \Omega_p \) will also be useful when resampling Radon data in log-polar coordinates.

5 Interpolation

As mentioned in the previous section it will be natural to use equally spaced sampling in the Cartesian, polar and log-polar coordinate systems. We typically want to reconstruct data on an equally spaced Cartesian grid; the tomographic data is sampled in equally spaced polar coordinates; and both the Radon transform and the back-projection can be rapidly evaluated by FFT when sampled on an equally spaced log-polar grid. There are several ways to interpolate between these coordinate systems. In this work we will make use of cubic (cardinal) B-spline interpolation. We will discuss how to incorporate some of the interpolation steps in the FFT operations and also discuss how the B-spline interpolation can be efficiently implemented on graphical processor units (GPUs).

The cubic cardinal B-spline is defined as

\[
B(x) = \begin{cases} 
(3|x|^3 - 6|x|^2 + 4)/6, & 0 \leq |x| < 1, \\
(-|x|^3 + 6|x|^2 - 12x + 8)/6, & 1 \leq |x| < 2, \\
0, & |x| \geq 2.
\end{cases}
\]
This function is designed so that it is nonnegative, piecewise smooth, and \( C^2 \) at \(|x| = 0, 1, 2\). Note that at integer points it holds that

\[
B(j) = \begin{cases} 
\frac{2}{3} & \text{if } j = 0, \\
\frac{1}{6} & \text{if } |j| = 1, \\
0 & \text{otherwise.}
\end{cases}
\]  

(23)

When used as a filter, this will smooth out information and is thus acting as a low-pass filter. Consequently, it cannot be used directly for interpolation.

The Fourier series given by the coefficients in (23), which we denote by \( \hat{B} \), is given by

\[
\hat{B}(\xi) = \sum_j B(j) e^{-2\pi i j \xi} = \frac{1}{6} (e^{2\pi i x \xi} + 4 + e^{-2\pi i x \xi}) = \frac{2}{3} + \frac{1}{3} \cos(2\pi x \xi). \]  

(24)

Suppose that equally spaced samples \( f_k \) of a function \( f \) (in one variable) are available. We want to recover values of \( f \) at arbitrary points \( x \) using

\[
f(x) = \sum_k (Qf)_k B \left( x - \frac{k}{N} \right),
\]  

(25)

where the operator \( Q \) is a prefilter operation, which is compensating for the fact that convolution with \( B \) suppresses high frequencies. Since \( B \) has short support, only values of \( (Qf)_k \) for \( k \approx xN \) will contribute in this sum. The prefilter operation, which is boosting high frequencies in the samples \( f_k \), can be computed in different ways. Perhaps the most direct way is to define \( Q \) in the Fourier domain (by the discrete Fourier transform), where it essentially becomes division by \( \hat{B} \) (upon scaling and sampling). In this case, it is easy to see that the convolution with \( B \) and the prefilter operation will cancel each other at points \( x = \frac{j}{N} \); i.e., the original function is recovered at the sample points which is a requisite for any interpolation scheme.

As we will compute the Radon transform and the back-projection by means of FFT in log-polar coordinates, we can in some steps incorporate the prefilter step \( 1/\hat{B} \) in the Fourier domain, at virtually no additional cost. However, not all of the prefilter operations can be incorporated in this way. While the prefilter easily can be applied by separate FFT operations, we want
Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators

to limit the total number of FFT operations, as these will be the most time-consuming part in the implementations we propose.

As an alternative to applying the prefilter in the Fourier domain, it can be applied by recursive filters. In [36] these operations are derived by using the Z-transform. It turns out that if we define an auxiliary sequence \((g_k)\) by

\[
g_k = 6f_+(\sqrt{3} - 2)g_{k-1},
\]

then

\[
(Qf)_k = (\sqrt{3} - 2)((Qf)_{k+1} - g_k); \tag{26}
\]

cf. [36, equations (12),(13)], where the boundary condition on \((Qf)_k\) and \((g_k)\) are given in [36, equations (14),(15)]. These filters can be efficiently implemented on GPUs.

Two-dimensional prefiltration can be done in two steps, one in each dimension. We will use the notation \(Qf\) for the prefiltering also in this case. On a GPU this implies doing operations on rows and columns separately. However, there are highly optimized routines for transposing data, which means that the prefiltration can be made to act only on column data. This is done to improve the so-called memory coalescing and GPU cache performance [44]. Memory coalescing refers to combining multiple memory accesses into a single transaction. In this way, the GPU threads run simultaneously and substantially increased cache hit ratios are obtained.

Let us now turn our focus to the convolution step in (25). Let

\[
k = \lfloor Nx \rfloor, \\
\alpha = x - \lfloor x \rfloor.
\]

The sum (25) then reduces to

\[
f(x) = w_0(\alpha)(Qf)_{k-1} + w_1(\alpha)(Qf)_k + w_2(\alpha)(Qf)_{k+1} + w_3(\alpha)(Qf)_{k+2},
\]

where

\[
w_0(\alpha) = B(\alpha - 2), \quad w_1(\alpha) = B(\alpha - 1), \quad w_2(\alpha) = B(\alpha), \quad w_3(\alpha) = B(\alpha + 1). \tag{27}
\]

We now discuss how this sum can be evaluated using linear interpolators. As mentioned previously, linear interpolation is executed fast on GPUs. In [39] it is shown how this can be utilized to conduct efficient cubic interpolation. The cubic interpolation is expressed as two weighted linear interpolations,
instead of four weighted nearest neighbor look-ups, yielding $2^d$ operations instead of $4^d$ for conducting cubic interpolation in $d$ dimensions.

We briefly recapitulate the approach taken in [35, 39]. Given coefficients $(Qf)_k$, let $Qf_{\text{lin}}$ be the linear interpolator

$$Qf_{\text{lin}}(x) = (1 - (\alpha))(Qf)_k + \alpha(Qf)_{k+1} = (1 - (x - \lfloor x \rfloor))(Qf)_{\lfloor Nx \rfloor} + (x - \lfloor x \rfloor)(Qf)_{\lfloor Nx \rfloor + 1}.$$ 

The sum (27) can then be written as

$$f(x) = (w_0(\alpha) + w_1(\alpha))Qf_{\text{lin}}\left(k - 1 + \frac{w_1(\alpha)}{w_0(\alpha) + w_1(\alpha)}\right) + (w_2(\alpha) + w_3(\alpha))Qf_{\text{lin}}\left(k + 1 + \frac{w_3(\alpha)}{w_2(\alpha) + w_3(\alpha)}\right). \tag{28}$$

The evaluation of the function $Qf_{\text{lin}}(x)$ can be performed by hard-wired linear interpolation on GPU. In modern GPU architecture, the so-called texture memory (cached on a chip) provides effective bandwidth by reducing memory requests to the off-chip DRAM. The two most useful features of this kind of memory with regards to conducting B-spline interpolation are:

1. The texture cache is optimized for the two-dimensional spatial locality, giving the best performance to GPU threads that read texture addresses that are close together.

2. Linear interpolation of neighboring values can be performed directly in the GPU’s texture hardware, meaning that the cost for computing the interpolation is the same as reading data directly from memory.

This implies that the cost for memory access in (28) will be two instead of four as only two function calls of $Qf_{\text{lin}}$ are made in (28), and in two dimensions the corresponding reduction from 16 memory access operations to 4 will give significant improvement in computational speed.

6 Algorithms

We now have the necessary ingredients to present detailed descriptions of how to rapidly evaluate the Radon transform and the back-projection operator by FFT in log-polar coordinates. In the algorithms below, we let $\hat{B}_{k_\theta, k_\rho}$
denote the values of the two-dimensional counterpart of (24), scaled to represent the sampling on \((\theta, \rho) \in [-\beta, \beta] \times [\log a_r, 0]\).

**Algorithm 1** Fast Radon transform

1. Given \(f\) sampled at \(X\) (18) compute \(Qf\) by (26)
2. for \(m = 0, \ldots, M - 1\) do
3. Resample \(T_m f\) at \(\Omega_{lp}\) (21) by (25)
4. Downsample from \(\Omega_{lp}\) (21) to \(\Omega_p\) (22)
5. Multiply result by \(e^\rho\)
6. Apply the log-polar Radon transform with prefiltering incorporated, i.e., compute \(\hat{f}_{k_\theta, k_\rho}\) from (16), and evaluate

\[
\sum_{k_\theta, k_\rho} \hat{f}_{k_\theta, k_\rho} \frac{\zeta_{k_\theta, k_\rho}}{B_{k_\theta, k_\rho}} e^{2\pi i (j_\theta k_\theta \Delta \theta / 2N_\theta + j_\rho k_\rho \Delta \rho / N_\rho)}
\]

by using FFT.
7. Resample from \(S_{-1}^m \Omega_p\) (22) to \(\Sigma\) (19) by (25)
8. end for

A few remarks on time complexity. The most time-consuming part of both algorithms are the convolutions that are implemented by FFT. In total, \(2M\) FFT operations need to be computed (including forward and backward FFTs), and each operation will be done on a grid of size \(\frac{2N_\theta}{M} \times N_\rho\). For \(M = 3\), it follows from (20) that

\[
N_\rho \approx -\frac{N \log (a_r)}{2a_R} \approx 2.1N,
\]

and \(N_\rho\) is monotonically decreasing for increasing \(M\) (since \(a_r\) and \(a_R\) depend on \(\beta\) (recall Figure 3) which equals \(2\pi / M\) by (6)). For \(M = 3\) we thus need roughly twice as many samples in \(\rho\) compared to those originally used for the sampling of \(s\) (\(N_s = N\)), and in the angular variable we also need to sample with twice as many parameters in order to avoid aliasing effects. In total, we need to use an oversampling of about 4 in the FFT operations. Note that
Algorithm 2 Fast back-projection

1: Given \( g \) sampled at \( \Sigma \) (19) compute \( Qg \) by (26)
2: for \( m = 0, \ldots, M - 1 \) do
3:     Resample \( g(\Sigma_m) \) at \( \Omega_p \) (22) by (25)
4:     Apply the log-polar back-projection with prefiltering incorporated, i.e., compute \( \hat{g}_{k_\theta, k_\rho} \) from (17) and evaluate
5:     Resample from \( T_m^{-1}\Omega_p \) (22) to \( X \) (18) by (25) to obtain partially back-projected data
6: end for
7: Sum up the \( M \) partial back-projections

this is the same oversampling that is generally needed for computing the convolution of two functions if aliasing is to be avoided. The total time spent on FFT is thus \( M \) times the time it takes to do an FFT of size \( \frac{2N_\theta}{M} N_\rho \) which is comparable to the time it takes to do one FFT operation of size \( 2N_\theta N_\rho \). The FFT cost for applying the Radon transform and the back-projection operator is thus about the same as would be expected for a generic convolution. In addition to the FFT time, interpolation also has to be conducted. For the back-projection case, the obtained result after computing a partial back-projection needs to be interpolated onto a \( N \times N \) lattice, yielding an interpolation complexity of \( \mathcal{O}(MN^2) \). A similar result holds for the (forward) Radon transform. This gives a total computational cost that has \( \mathcal{O}(N^2 \log N + MN^2) \) cost. In practice the parameter \( M \) should be chosen small and such that the data sizes that FFT is applied to are powers of two. In our simulations we use \( M = 3 \) which fits well for the typical setup where \( N \) is a power of two and \( N_\theta = 3N/2 \) directions are used.
7 Numerical experiments

Let us start by briefly discussing the filters used in the filtered back-projection (2). This discussion is included in order to better interpret the errors obtained when comparing different methods. For theoretically perfect reconstruction (with infinitely dense sampling), the filter \( \mathcal{W} \) in (2) is given by

\[
\hat{w}_{\text{ramp}}(\sigma) = |\sigma|.
\]  

This filter is sometimes referred to as the ramp filter. If the sampling rate is insufficient in relation to the frequency content of \( f \) (or the object upon which the measurements is conducted), it can be desirable to suppress the highest frequencies in order to localize the effects of the insufficient sampling. There is a relation between one-dimensional filtering in the \( s \)-direction of Radon data, and a two-dimensional convolution in the spatial domain; see the computation following (18) or \([30, \text{p. 102}]\). It can be explained by the Fourier slice theorem, which describes how Radon data can be converted to a polar sampling in the frequency domain by taking one-dimensional Fourier transforms in the \( s \)-direction. The application of the one-dimensional filter will then correspond to a change in amplitude (and possible phase) along the lines (indicated by dots with the same angles) in Figure 5b). As there is no information outside the circle with radius \( N/2 \), the action of the ramp filter will be the equivalent of applying a two-dimensional convolution to the original
function \( f \) using a two-dimensional filter with Fourier transform

\[
\hat{W}_{\text{ramp}}(\xi) = \begin{cases} 
1 & \text{if } |\xi| \leq \frac{N}{2}, \\
0 & \text{otherwise}.
\end{cases}
\tag{30}
\]

We see that if \( f \) contains more high-frequency information than that prescribed by the sampling rate \( N \), then the sharp cutoff in (30) can yield artifacts, and in the presence of noise in the Radon sampling the high-frequency boosting of (29) will boost the noise. Replacing the ramp filter (29) with a filter that goes smoothly to zero at the highest (discrete) frequencies will thus yield an image that is slightly smoother but, on the other hand, an image with suppressed high-frequency noise and artifacts due to incomplete sampling. Sometimes, the ramp filter is modified so that it does not reach zero but only reduces the high-frequency amplitudes. Two common choices of filters are the cosine and the Shepp-Logan filters, defined by

\[
\hat{w}_{\text{cos}}(\sigma) = \begin{cases} 
|\sigma| \cos\left(\frac{\pi\sigma}{N}\right) & \text{if } |\sigma| \leq \frac{N}{2}, \\
0 & \text{otherwise}.
\end{cases}
\]

\[
\hat{w}_{\text{SL}}(\sigma) = \begin{cases} 
|\sigma| \text{sinc}\left(\frac{\sigma}{N}\right) & \text{if } |\sigma| \leq \frac{N}{2}, \\
0 & \text{otherwise}.
\end{cases}
\]

The cutoff is made above the sampling bandwidth, as it will illustrate the practical effect that the filters have on measured data. The three filters are illustrated in the frequency domain in Figure 6. The two-dimensional filters associated with these filters have Fourier representations

\[
\hat{W}_{\text{cos}}(\xi) = \begin{cases} 
\cos\left(\frac{\pi|\xi|}{N}\right) & \text{if } |\xi| \leq \frac{N}{2}, \\
0 & \text{otherwise},
\end{cases}
\]

and

\[
\hat{W}_{\text{SL}}(\xi) = \begin{cases} 
\text{sinc}\left(\frac{|\xi|}{N}\right) & \text{if } |\xi| \leq \frac{N}{2}, \\
0 & \text{otherwise},
\end{cases}
\]

respectively. As the \( \hat{w}_{\text{cos}} \) goes to zero at the highest sampling rate, we can expect smaller errors when using this filter compared to the others. On the
other hand, the highest frequencies are suppressed, and the resulting reconstructions will look slightly less sharp.

To illustrate the accuracy of the suggested implementation, we conduct some examples on the Shepp-Logan\cite{shepp-logan} phantom. We use the modified version introduced in \cite[Appendix B.2]{modified-version}. The function used ($f$) is illustrated in Figure 2a). The phantom consists of linear combinations of characteristic functions of ellipses, and its support is inscribed in the unit circle. Since the Radon transform is a linear operator, $\mathcal{R}f$ is a linear combination of Radon transforms of characteristic functions of ellipses. Since the Radon transform of the characteristic function of a circle can be computed analytically, analytic expressions are available for the Radon transform of the Shepp-Logan phantom by applying transform properties of shifting, scaling, and rotation. This Radon transform is depicted in Figure 2b). For the sake of quality comparison, we use the results of the back-projection from the following packages:

- MATLAB Image Processing Toolbox
- ASTRA Tomography Toolbox
- NiftyRec Tomography Toolbox
- SPIRAL back-projection package
- Image Reconstruction Toolbox (IRT)
- NFFT3

The presence of a high-frequency discontinuity caused by the filter can cause artifacts, but also the discontinuity of the derivative at $\sigma = 0$. To avoid artifacts from this part, we apply end-point trapezoidal corrections. The effect of this correction can be seen around $\sigma = 0$ in Figure 6. Our aim is to eliminate as much of the error as possible to distinguish the errors that the resampling between the different coordinate systems used in the proposed methods introduces.

In Figure 7 we show some reconstruction results from the filtered back-projection using the log-polar-based method. For these tests we use exact expressions for Shepp-Logan phantom and its Radon data, when applying the ramp, Shepp-Logan and cosine filters. The tests are performed for Radon data of sizes $(N_\theta \times N_s) = (\frac{3}{2} N \times N)$, where $(N \times N)$ is the size of the reconstructed
Figure 7: Computational errors of the filtered back-projection for the log-polar-based method, with the ramp, Shepp-Logan and cosine filters.

image. For the log-polar reconstruction, we used $M = 3$ partial reconstructions, and $N_\rho = 2N$. The reconstructed panels have inscribed normalized $\ell^2$-errors against the filtered versions of the phantom. The tests are performed for $N = 512$.

In the reconstruction results that follow, we use the Shepp-Logan filter for accuracy comparisons. The different packages have different interfaces and different sets of parameters, and therefore some code modifications have to be done for making suitable comparisons. We have tried to keep the code modifications at a minimum and measure time only for the computation of the back-projection. The times for the precomputational steps (initialization
Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators

of grids, computing basis functions, etc.) are excluded from the total time as these can be computed off-line. All the packages contain examples on how to perform back-projection or reconstruction of different types.

MATLAB. The MATLAB functions `radon` and `iradon` are used as reference methods for comparing reconstruction accuracy of the back-projection performed with linear-, cubic-, and spline-based interpolation. As suggested during the review, the computational times for the MATLAB reconstructions are not included in the speed comparison tables.

ASTRA. The ASTRA Toolbox [31, 42] is a MATLAB toolbox of GPU primitives for two- and three-dimensional tomography. In the simulations presented in this section, ASTRA v1.6 is used, and results are based on the routine `s014_FBP`. Iterative tests in the next section are based on the script `s003_gpu_reconstruction` where the parameter `EM_CUDA` is used for reconstruction.

NiftyRec. The NiftyRec Toolbox [33, 32] comes with both GPU and CPU implementations of Radon transforms. The results presented here uses NiftyRec v2.0 and make use of the back-projection part of the demo `tt_demo_mlem_parallel` for parallel beam transmission tomography with zero attenuation.

SPIRAL. The SPIRAL back-projection package implements the fast hierarchical back-projection method (FHBP) [7, 34]. The code was compiled using the Intel compiler (icc version 15.0.3) with the flag `-fast`. Two parameters are used to control the hierarchical scales: the parameter $B$ is the base case size, at which the program stops recursion; the parameter $D$ is the size of the image at which the program starts (angularly) to down-sample the filtered sinograms. Varying these parameters one can control accuracy and computational time. Following the setup in [34, Figure 3], the simulation results presented here use $B = \frac{N}{8}, D = \frac{N}{4}$ (hierarchical approach), and $B = N, D = N$ (which reduces the algorithm to the classic FBP algorithm).

IRT. The Image Reconstruction Toolbox (IRT)[19] includes a parallel implementation of the FBP (using OpenMP), as well as the Fourier-based approach. In this implementation the USFFT algorithms use Kaiser-Bessel bell functions with an additional min-max criterion, cf. [20]. The simulations conducted are based on the examples `fbp/fbp2_example, systems/tests/Gtomo_nufft_test`. The USFFT tests are based on the MATLAB script.
NFFT3. The C library NFFT 3.0 \cite{27} provides algorithms for unequally spaced FFTs. The library also contains routines for computing Radon transforms either using the Fourier slice theorem or using the pseudopolar Fourier transform. For our tests, we adapt the MATLAB script \texttt{radon/radon} to use only one back-projection step (it originally uses an iterative method).

For the pseudopolar grids, it is assumed that the data is sampled in a non-standard fashion, i.e., not equally spaced in the $\theta$ and $s$-variables. This is an effect of the fact that the directions of the lines, as well as the corresponding spacing in the radial variable, have to be adapted so that they agree with the pseudopolar frequency lattice. As the pseudopolar grids are not orientation invariant, the Shepp-Logan filter should be slightly modified to include angular variations. The effect seems negligible, and we use the standard filter in what follows. The simulations make use of the fact that there are exact formulas for the Radon data of the Shepp-Logan phantom in order produce samples that agree with the pseudopolar format, and we therefore sample the data at these locations. In practice, an interpolation step would be needed here.

For the tests a standard desktop computer with an Intel Core i7-3820 processor and an NVIDIA GeForce GTX 770 graphic card was used. The packages have been tested with MATLAB 2015b via MEX interfaces or via execution of binary files. All packages were tested in single precision. Figure 8 displays accuracy results. The reconstructions as before were made on a $512 \times 512$ grid using 768 samples in the angular direction.

The top three panels of Figure 8 show the Fourier-based methods, using high-order interpolation in the frequency domain. The next four panels (the second row and the left panel in the third row) show results using third-order interpolation in the Radon domain, and the subsequent four panels show the results using linear interpolation in the Radon domain. The last (bottom right) panel shows the hierarchical reconstruction.

Table 1 contains computation times for the different methods. Since the methods include both algorithmic and hardware-accelerated methods, we also include times for a CPU version of the proposed method. This version makes use of the Intel libraries MKL (for FFT) and IPP (for interpolation), which contain highly optimized standard routines. The proposed algorithm had to be modified slightly in order to use the interpolation methods available for
Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators

Table 1: Computational time (in seconds) of the back-projection for sizes \((N_\theta \times N_s) = (2N \times N)\) (excluding initialization time).

<table>
<thead>
<tr>
<th>Package</th>
<th>Method</th>
<th>PU</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>Log-polar</td>
<td>GPU</td>
<td>1.8e-04</td>
<td>7.4e-04</td>
<td>2.9e-03</td>
<td>1.2e-02</td>
<td>4.5e-02</td>
</tr>
<tr>
<td>ASTRA</td>
<td>FBP</td>
<td>GPU</td>
<td>3.3e-04</td>
<td>1.4e-03</td>
<td>7.1e-03</td>
<td>4.3e-02</td>
<td>2.8e-01</td>
</tr>
<tr>
<td>LP</td>
<td>Log-polar</td>
<td>CPU</td>
<td>2.5e-03</td>
<td>5.8e-03</td>
<td>2.6e-02</td>
<td>1.1e-01</td>
<td>4.0e-01</td>
</tr>
<tr>
<td>NFFT3</td>
<td>Fourier</td>
<td>CPU</td>
<td>6.1e-03</td>
<td>3.6e-02</td>
<td>1.7e-01</td>
<td>6.9e-01</td>
<td>3.2e+00</td>
</tr>
<tr>
<td>NFFT3</td>
<td>Fourier(^{\text{pseudo-polar}})</td>
<td>CPU</td>
<td>1.3e-02</td>
<td>5.6e-02</td>
<td>2.2e-01</td>
<td>9.2e-01</td>
<td>4.6e+00</td>
</tr>
<tr>
<td>IRT</td>
<td>Fourier</td>
<td>CPU</td>
<td>3.3e-02</td>
<td>8.7e-02</td>
<td>3.7e-01</td>
<td>1.5e+00</td>
<td>6.3e+00</td>
</tr>
<tr>
<td>IRT</td>
<td>FBP</td>
<td>CPU</td>
<td>1.1e-02</td>
<td>4.1e-02</td>
<td>3.9e-01</td>
<td>1.7e+00</td>
<td>1.2e+01</td>
</tr>
<tr>
<td>NiftyRec</td>
<td>FBP</td>
<td>GPU</td>
<td>3.7e-03</td>
<td>2.9e-02</td>
<td>2.4e-01</td>
<td>1.7e+00</td>
<td>1.4e+01</td>
</tr>
<tr>
<td>SPIRAL</td>
<td>Hierarchical</td>
<td>CPU</td>
<td>6.7e-02</td>
<td>3.8e-01</td>
<td>2.5e+00</td>
<td>2.1e+01</td>
<td>1.9e+02</td>
</tr>
<tr>
<td>SPIRAL</td>
<td>FBP</td>
<td>CPU</td>
<td>8.0e-02</td>
<td>3.3e-01</td>
<td>5.2e+00</td>
<td>4.9e+01</td>
<td>3.9e+02</td>
</tr>
</tbody>
</table>

the IPP function \texttt{remap}. Specifically, the cubic interpolation method is used (giving a result comparable to that of cubic filtered back-projection).

To exclude initialization effects and assure high GPU load, the times in Table 1 are obtained by executing batches of reconstructions (to fill up the GPU memory) in one function call. Table 2 presents corresponding slowdown coefficients compared to the log-polar implementation. We note that the proposed program on GPU is about 2-6 times faster than ASTRA, substantially faster than the GPU version of NiftyRec, and about 2 or more magnitudes faster than the other CPU-based methods.

8 Performance analysis of the log-polar implementation

For the GPU implementation of the proposed method, version 7.5 of the CUDA Toolkit \[16\] is being used. The computations were performed in single precision.

Table 3 shows a breakdown of the computational times for the log-polar Radon transform. This table shows that most computational time is due to
Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators.

Figure 8: Computational errors of the filtered back-projection for the Shepp-Logan filter.
Table 2: Slowdown compared to the log-polar implementation on GPU for the back-projection for sizes $(N_\theta \times N_s) = (\frac{3}{2} N \times N)$ (excluding initialization time).

<table>
<thead>
<tr>
<th>Package</th>
<th>Method</th>
<th>PU</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>Log-polar</td>
<td>GPU</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Astra</td>
<td>FBP</td>
<td>GPU</td>
<td>1.8</td>
<td>1.9</td>
<td>2.4</td>
<td>3.6</td>
<td>6.2</td>
</tr>
<tr>
<td>LP</td>
<td>Log-polar</td>
<td>CPU</td>
<td>13.7</td>
<td>7.9</td>
<td>8.7</td>
<td>9.5</td>
<td>8.9</td>
</tr>
<tr>
<td>NFFT3</td>
<td>Fourier</td>
<td>CPU</td>
<td>33.1</td>
<td>49.4</td>
<td>56.9</td>
<td>57.5</td>
<td>70.1</td>
</tr>
<tr>
<td>NFFT3</td>
<td>Fourier(pseudo-polar)</td>
<td>CPU</td>
<td>70.6</td>
<td>75.6</td>
<td>75.7</td>
<td>77.4</td>
<td>100.8</td>
</tr>
<tr>
<td>IRT</td>
<td>Fourier</td>
<td>CPU</td>
<td>179.2</td>
<td>118.5</td>
<td>123.9</td>
<td>128.1</td>
<td>139.8</td>
</tr>
<tr>
<td>IRT</td>
<td>FBP</td>
<td>CPU</td>
<td>60.1</td>
<td>55.8</td>
<td>132.3</td>
<td>140.9</td>
<td>270.2</td>
</tr>
<tr>
<td>NiftyRec</td>
<td>FBP</td>
<td>GPU</td>
<td>20.4</td>
<td>39.3</td>
<td>81.2</td>
<td>143.4</td>
<td>305.5</td>
</tr>
<tr>
<td>SPIRAL</td>
<td>Hierarchical</td>
<td>CPU</td>
<td>364.5</td>
<td>511.2</td>
<td>839.9</td>
<td>1763.3</td>
<td>4082.4</td>
</tr>
<tr>
<td>SPIRAL</td>
<td>FBP</td>
<td>CPU</td>
<td>437.4</td>
<td>441.1</td>
<td>1760.1</td>
<td>4151.5</td>
<td>8648.1</td>
</tr>
</tbody>
</table>

Table 3: Time (in seconds) of main computational parts for the log-polar Radon transform.

<table>
<thead>
<tr>
<th>N</th>
<th>Prefilter</th>
<th>Interp</th>
<th>FFT</th>
<th>Other</th>
<th>copy</th>
<th>CPU-GPU</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>1.8e-05</td>
<td>3.2e-05</td>
<td>3.0e-05</td>
<td>3.9e-05</td>
<td>5.4e-05</td>
<td>1.7e-04</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>5.1e-05</td>
<td>1.3e-04</td>
<td>1.7e-04</td>
<td>1.6e-04</td>
<td>2.1e-04</td>
<td>7.1e-04</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>1.8e-04</td>
<td>4.8e-04</td>
<td>7.2e-04</td>
<td>6.3e-04</td>
<td>8.3e-04</td>
<td>2.8e-03</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>8.5e-04</td>
<td>1.9e-03</td>
<td>2.9e-03</td>
<td>2.5e-03</td>
<td>3.3e-03</td>
<td>1.1e-02</td>
<td></td>
</tr>
<tr>
<td>2048</td>
<td>4.9e-03</td>
<td>7.6e-03</td>
<td>1.2e-02</td>
<td>5.8e-03</td>
<td>1.4e-02</td>
<td>4.4e-02</td>
<td></td>
</tr>
</tbody>
</table>

' - Small GPU kernels, GPU-GPU copy.
CPU-GPU data transfers. The transfers can be performed in parallel with computations in the case of processing big three-dimensional data cubes: while one portion of data is computed, another portion can be transferred to/from GPU memory. CUDA Streams can be used to overlap data transfers and computations. In this way, operations in different streams can be interleaved and, when possible, they can run concurrently. This will reduce the impact of the CPU-GPU copy operations on the total execution time.

9 Iterative methods for tomographic reconstruction

In some situations it is preferable to use an iterative method for doing reconstruction from tomographic data. This could, for instance, be because of missing data, e.g., that data for some angles are missing; suppression of artifacts [29]; or that additional information about the noise contamination can be used to improve the reconstruction results compared to direction filtered back-projections (2). Iterative reconstruction methods rely on applying the forward and back-projection operators several times. Iterative algorithms can be computationally expensive when a large number of iterations is required for the algorithm to converge. For that reason, fast algorithms for computing the Radon transform and the associated back-projection can play an important role.

Iterative algebraic reconstruction techniques (ART) are popular tools for reconstruction from incomplete measurements [24]. The approach aims at solving the set of linear equations determined by the projection data. Transmission based tomographic measurements measure the absorption of a media along a line. This puts a sign constraint on the function we wish to recover. It is clearly not ideal to assume that the data is contaminated by normally distributed noise (for which the best estimate is given by the least squares estimates). A more reasonable assumption is that the added noise has a Poisson distribution [6, 45]. The simplest iterative method for solving the estimation/reconstruction problem under this noise assumption is by the expectation-maximization (EM) algorithm. The EM algorithm is well-suited to reconstruct tomography data with non-Gaussian noise [15, 28, 38]. Alternative techniques include, for instance, the row action maximum likelihood algorithm (RAMLA) [10, 14].

More details about the EM algorithm can be found, for instance, in [11,
In our notation, it can (given tomographic data $g$) be expressed as the iterative computations of

$$f^{k+1} = f^k \frac{R^\# \left( \frac{g}{R f^k} \chi_C \right)}{R^\# \chi_C},$$

where the function $\chi_C(\theta, s)$ is one if the line parameterized by $(\theta, s)$ passes through the unit disc (the support of $f$) and zero otherwise. In each step, a Radon transform $R f^k$ and a back-projection $R^\# \left( \frac{g}{R f^k} \right)$ is computed.

It is common in GPU computing that the host-device memory transfers constitute an essential part of the total computational cost. For iterative methods (such as the one described above) it is possible to keep all the necessary data in the GPU memory, thereby limiting the data copy between host and device memory to an initial guess $f^0$, the measured data $g$, and the final result. As with most methods, the proposed method requires some initialization steps (e.g., geometry parameters and convolution kernels $\zeta, \zeta^\#$).

Figure 9a) shows the Radon data with Poisson noise, and Figure 9b) shows the corresponding filtered back-projection. Note that the noise level is rather higher in this reconstruction. In contrast, the denoised recovered data after 50 iterations of the EM algorithm is presented in Figure 9c).

We present performance results for the EM algorithm for the proposed method and the methods implemented on GPU from ASTRA and NiftyRec tomography toolboxes. The computational times for conducting 100 iterations of the EM algorithm for different resolution parameters are presented in Table 4. The Radon data was sampled with parameters $N_\theta = \frac{3}{2} N$ and $N_s = N$, and $M = 3$ partial back-projection was used.
Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators

Table 4: Time (in seconds) for 100 iterations of the EM algorithm for reconstruction of three-dimensional data of sizes \((N \times N \times N)\).

<table>
<thead>
<tr>
<th>Package</th>
<th>Method</th>
<th>PU</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>Log-polar GPU</td>
<td></td>
<td>3.1e+00</td>
<td>2.6e+01</td>
<td>2.1e+02</td>
<td>1.7e+03</td>
<td>1.3e+04</td>
</tr>
<tr>
<td>ASTRA</td>
<td>FBP GPU</td>
<td></td>
<td>2.2e+01</td>
<td>6.8e+01</td>
<td>4.2e+02</td>
<td>5.3e+03</td>
<td>8.2e+04</td>
</tr>
<tr>
<td>NiftyRec</td>
<td>FBP GPU</td>
<td></td>
<td>8.5e+01</td>
<td>1.3e+03</td>
<td>2.1e+04</td>
<td>3.0e+05</td>
<td>4.8e+06</td>
</tr>
</tbody>
</table>

Figure 9: Reconstruction of Radon data with noise. Radon data with Poisson noise is shown in the panel (a), the result of applying filtered back-projection is shown in the panel (b), the panel (c) shows the result after 50 iterations of the EM algorithm.
10 Conclusions

We have described how to implement the Radon transform and the back-projection as convolution operators in log-polar coordinates efficiently on GPUs. We present sampling conditions; provide formulas and numerical guidelines for how to compute the kernels associated with the Radon transform and the back-projection operator; and discuss how the convolutions can be rapidly evaluated by using FFT. The procedure involves several steps of interpolation between data in the Radon domain, the spatial domain, and the log-polar domain. It is comparatively favorable to conduct interpolation in these domains compared to conducting interpolation in the Fourier domain, as the functions will tend to be less oscillatory there. We use cubic spline interpolation which can be efficiently implemented on GPUs, and optimized routines for FFT on the GPU. We conduct numerical tests and conclude that reasonable accuracy can be obtained at favorable computational cost.

Acknowledgments

This work was supported by the Crafoord Foundation (2014-0633) and the Swedish Research Council (2011-5589, 2015-03780)

References


[4] Amir Averbuch, Ronald R Coifman, David L Donoho, Michael Elad,
Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators


Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators


[34] Thammanit Pipatsrisawat, Aca Gačić, Franz Franchetti, Markus Püschel, and Josée MF Moura. Performance analysis of the filtered


Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators


Fast Laplace transforms for the exponential Radon transform

Fredrik Andersson, Marcus Carlsson, and Viktor V. Nikitin

Abstract

The Fourier slice theorem for the standard Radon transform generalizes to a Laplace counterpart when considering the exponential Radon transform. We show how to use this fact in combination with algorithms for the unequally spaced fast Laplace transform to construct fast and accurate methods for computing both the forward exponential Radon transform and the corresponding back-projection operator.

1 Introduction

One of the most popular ways of computing the standard two-dimensional Radon transform for parallel beam geometry relies on Fourier transforms via the Fourier slice theorem. A standard discretization with equally spaced samples in the spatial variables yields a relation between the sampled spatial data and its Fourier transform on a polar lattice. This can be rapidly evaluated by using a combination of standard FFT and algorithms for unequally spaced FFT. In this way the time complexity is essentially reduced by one order, from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2 \log N)$. The main purpose of this paper is to show that a similar reduction in computational speed can be obtained for the exponential Radon transform, by relying on algorithms for unequally spaced fast Laplace transforms instead of the counterpart for Fourier transforms. The same holds for the adjoint operator, sometimes called the exponential back-projection operator, which in particular enables us to implement a fast algorithm for the inverse exponential Radon transform, based on a formula by Tretiak and Metz [34]. The need for fast transforms is particularly important.
when employing iterative schemes for the solution of (ill-posed) inverse problems, arising e.g. from incomplete angular coverage in the measurements. We illustrate this by numerical examples, making use of standard techniques for the reconstruction procedure, in order to keep the focus on producing fast algorithms for the evaluation of the exponential Radon transform and its adjoint.

Upon discretization, the Radon transform becomes an operator on finite dimensional space, having a matrix representation. The same goes for the inverse and adjoint Radon transform, and these discretizations are associated with approximation errors which depend on the choice of specific parameters etc. On top of that, there are errors that arise when replacing the discrete operator with its fast counterpart, which is the topic of this paper. We show that the errors that come from the latter part can be controlled to arbitrary precision level $\epsilon$ at a computational cost that is in practice proportional to $-\log(\epsilon)$. In this way, reconstruction issues concerning for instance noise sensitivity can be isolated to the (standard) sampling setup, with only minor influence coming from the proposed fast computational algorithms.

The inverse problem of reconstructing data modeled by the exponential Radon transform is ill-posed, meaning that addition regularization techniques often needs to be applied in order to obtain suitable reconstructions of data. There are several different regularization approaches in the literature, and the particular type of regularization will typically depend on the structure of the sample measured on. We will therefore not focus on regularization techniques in this paper, but focus on how to construct fast algorithms for computing the exponential Radon transform and its adjoint. We will, however, include a description on how to include a discretized version of the filter operator present in the filtered back-projection type of reconstruction formula for the inversion of Radon data, as it is the most commonly used reconstruction method used for standard Radon data.

Let $s \in \mathbb{R}$ and $\theta \in S^1$, where $S^1$ denotes the unit circle. The standard two-dimensional Radon transform is defined as

$$\mathcal{R} f(s, \theta) = \int_{x \cdot \theta = s} f(x) \, dx = \int_{-\infty}^{\infty} f(s \theta + t \theta^\perp) \, dt,$$

and the parameters $s$ and $\theta$ are used to parameterize the set of lines in $\mathbb{R}^2$. The attenuated Radon transform in $\mathbb{R}^2$ describes the mapping from functions
Fast Laplace transforms for fast inversion of the exponential Radon transform

$f$ and $\mu$ to line integrals of the form

$$\mathcal{R}_{at} f(s, \theta) = \int_{-\infty}^{\infty} f(s\theta + t\theta_{\perp}) e^{-\int_{t}^{\infty} \mu(s\theta + \tau\theta_{\perp}) d\tau} dt.$$  

Integrals of this type appear for instance in medicine, where $f$ describes the intensity distribution of isotopes or radiopharmaceuticals inside a tissue, and where the function $\mu$ describes the attenuation of the tissue. For the case where $\mu$ is constant within a known convex body, then the data obtained from the attenuated Radon transform can be modeled by the simpler exponential Radon transform [33],

$$\mathcal{R}_{\mu} f(s, \theta) = \int_{-\infty}^{\infty} f(s\theta + t\theta_{\perp}) e^{\mu t} dt. \quad (1)$$

The adjoint operator associated with the exponential Radon transform (1) can be written as

$$\mathcal{R}_{\mu}^{*} g(x) = \int_{S^1} e^{\mu x \cdot \theta_{\perp}} g(x \cdot \theta, \theta) d\theta, \quad (2)$$

cf. [25, 34]. It is a weighted integral of $g$ over lines passing through the point $x$, and with $\mu = 0$ this operator is typically referred to as the back-projection operator.

In [25] a generalization of the Fourier slice theorem is derived for the exponential Radon transform along with an integral equation for the reconstruction problem of obtaining $f$ given $\mathcal{R}_{\mu} f$. A similar integral equation is also derived in [5]. There also exist integral equations for the attenuated Radon transform, where function $\mu$ has certain constraints, see [8, 26].

In [34] properties of the exponential Radon transform are derived, along with an inversion formula of filtered back-projection type. In Section 2 we recast this formula in a particularly simple format, see Theorem 2.1. Since then, the exponential Radon transform and the associated inversion problem have been discussed by several authors. For instance, in [1] range conditions of operator $\mathcal{R}_{\mu}$ are shown using Paley-Wiener type theorems. Cormack-type inversion formulas are based on the circular harmonic expansion of the transform and solving integral equations of special type (generalized Cormack
equations) and they are discussed in [27] with generalization for the attenuated Radon transform in [28]. An explicit integral formula for 180-degree data reconstruction is obtained in [29] along with numerical tests for the approximation of the integrals. Another approach to reconstruct data from 180-degree measurements is proposed in [20], the approach is analytical except for the pre-calculated inverse kernel, where the least-squares method is utilized. In [2] the exponential Radon transform is extended to various spaces of distributions. In [37, 38] it was shown that the exponential Radon transform inversion can be expressed as a convolution on Euclidean motion group (the rigid motion on $\mathbb{R}^2$).

2 The Fourier-Laplace slice theorem and inversion in the continuous case

The Fourier slice theorem which relates the one-dimensional Fourier transforms of Radon transformed data with the two-dimensional Fourier transform of data is fairly straightforward to generalize to the exponential Radon transform. To see this, we introduce the transform $\hat{\mathcal{R}}_\mu$ as the one-dimensional Fourier transform of the exponential Radon transform with respect to the first variable $s$ of $\mathcal{R}_\mu f$, i.e.

$$\hat{\mathcal{R}}_\mu f(\sigma, \theta) = \int_{-\infty}^{\infty} \mathcal{R}_\mu f(s, \theta) e^{-2\pi i s \sigma} ds = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(s \theta + t \theta^\perp) e^{-2\pi i s \sigma + \mu t} ds dt =$$

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(s \theta + t \theta^\perp) e^{-2\pi i (s \sigma + t \theta^\perp \cdot \theta)} ds dt =$$

$$\int_{\mathbb{R}^2} f(x) e^{-2\pi i x \cdot (\sigma \theta + \mu x \cdot \theta^\perp)} dx,$$

where the dot denotes scalar product in $\mathbb{R}^2$. Since $f$ by assumption has support in the bounded domain $\Omega$, its Fourier transform extends to an analytic function in $\mathbb{C}^2$, and hence we can express the outcome of the above calculation as

$$\hat{\mathcal{R}}_\mu f(\sigma, \theta) = \hat{f} \left( \sigma \theta + \frac{i \mu}{2\pi} \theta^\perp \right).$$

(3)
which is a generalization of the Fourier slice theorem, sometimes referred to as the Fourier-Laplace slice theorem. For more details, see [25, 34].

Note that $\hat{\mathcal{R}}_\mu$ maps compactly supported functions in $L^2(\mathbb{R})$ to rapidly decaying smooth functions in $L^2(\mathbb{Z})$ where $\mathbb{Z} = (\mathbb{R} \setminus \{0\}) \times S^1$ is a polar-type coordinate system, except for the fact that we allow negative radii. We now wish to express this as a transform acting into a space of functions on Cartesian-type coordinates. We therefore let $R^2$ denote the two copies of the punctured plane on top of each other. Concretely, we set $R^2 = (\mathbb{R} \setminus \{0\}) \times \{1, -1\}$ and define a bijection $\iota : \mathbb{Z} \rightarrow R^2$ via

$$\iota(\sigma, \theta) = \begin{cases} (\sigma \theta, 1) & \text{if } \sigma > 0, \\ (\sigma \theta, -1) & \text{if } \sigma < 0. \end{cases}$$

We will use the notation $\tilde{\xi} = (\xi, \alpha)$ for the independent variable in $R^2$, where $\xi$ is the Cartesian part of $\tilde{\xi}$ and $\alpha$ simply the sign of $\sigma$ in the identity $\tilde{\xi} = \iota(\sigma, \theta)$. For a function $g$ on $\mathbb{Z}$ or $R^2$ we will employ the standard convention of suppressing $\iota$ in the notation, writing $g(\sigma, \theta)$ when the former (polar type) coordinate system is used, and $g(\tilde{\xi})$ when the latter (Cartesian type) coordinate system is used.

Given any non-zero $x = (x_1, x_2) \in \mathbb{R}^2$ we introduce notation $x^\perp = \frac{(-x_2, x_1)}{|x|}$ and moreover we define

$$\tilde{\xi}^\perp = \alpha \xi^\perp, \quad \tilde{\xi} \in R^2.$$

Note that with this notation we always have $\tilde{\xi}^\perp = \theta^\perp$ whenever $\tilde{\xi} = \iota(\sigma, \theta)$. Finally, we let $H$ be the measure on $R^2$ that coincides with the Lebesgue measure on each of the two planes. Set

$$E(x, \tilde{\xi}) = e^{\mu x \cdot \tilde{\xi}^\perp + 2\pi i x \cdot \xi}$$

and

$$E^*(x, \tilde{\xi}) = e^{-\mu x \cdot \tilde{\xi}^\perp + 2\pi i x \cdot \xi}.$$

Then

$$\hat{\mathcal{R}}_\mu(f)(\tilde{\xi}) = \int f(x)e^{\mu x \cdot \tilde{\xi}^\perp - 2\pi i x \cdot \xi} \, dx = \langle f, E(\cdot, \tilde{\xi}) \rangle.$$ (4)

Using this notation, we shall show that the inversion formula by Tretiak and Metz takes a simple form given in the following theorem:
Theorem 2.1. Let \( f \) be a compactly supported smooth function. Then

\[
f(x) = \frac{1}{2} \int_{|\xi| > \frac{\mu}{2\pi}} \langle f, E(\cdot, \xi) \rangle E^*(x, \xi) dH(\xi)
\]

or, expressed differently,

\[
f(x) = \int_{|\xi| > \frac{\mu}{2\pi}} f(y) \cosh(\mu(y-x) \cdot \xi)\down E(x-y)\down e^{2\pi i(x-y) \cdot \xi} dy d\xi.
\]

Proof. The inversion formula by Tretiak and Metz (see Corollary 1 of [34]) is written as an integral formula involving a family of kernels depending on two parameters \( \sigma, \epsilon \), and a limit as these approach zero. For compactly supported smooth functions it is easy to see that the issues with divergent integrals disappear and the formula takes the following simpler form

\[
f = \mathcal{R}_{-\mu}^* \mathcal{W} \mathcal{R}_\mu f,
\]

where \( \mathcal{W} \) is a convolution (in \( s \)) with the inverse Fourier transform of the function \( w(\sigma) = |\sigma/2|1_{\{\sigma > \frac{\mu}{2\pi}\}}, \) and \( 1_{\{|\sigma| > \frac{\mu}{2\pi}\}} \) denotes the characteristic function of \( \{|\sigma| > \frac{\mu}{2\pi}\} \), (the normalization of the Fourier transform is different in [34], so the kernel looks slightly different there). This leads to

\[
f = \mathcal{R}_{-\mu}^* \mathcal{F}^* M_w \mathcal{F} \mathcal{R}_\mu f = \hat{\mathcal{R}}_{-\mu}^* M_w \hat{\mathcal{R}}_\mu f
\]

where \( M_w \) is multiplication by \( w \). Note that, for a smooth rapidly decaying function \( g \) on \( Z \) we have

\[
\langle \hat{\mathcal{R}}_{-\mu} f, g \rangle = \int \int \int_{\mathbb{R}^2} f(x)e^{-2\pi i x \sigma - \mu x \cdot \theta} d\sigma d\theta d x \overline{g(\sigma, \theta)} d \sigma d \theta = \int \int_{\mathbb{R}^2} f(x) \overline{\int \int e^{2\pi i x \sigma - \mu x \cdot \theta} g(\sigma, \theta) d \sigma d \theta} d x
\]

so

\[
\hat{\mathcal{R}}_{-\mu}^*(g)(x) = \int \int e^{2\pi i x \sigma - \mu x \cdot \theta} g(\sigma, \theta) d \sigma d \theta,
\]
Fast Laplace transforms for fast inversion of the exponential Radon transform

which gives

\[ \hat{\mathcal{R}}_{-\mu}(M_w g)(x) = \int_{S^1} \int_{|\sigma| > \frac{\mu}{2\pi}} \frac{|\sigma|}{2} g(\sigma, \theta)e^{-\mu x \cdot \theta^{\perp} + 2\pi i \sigma x \cdot \theta} d\sigma d\theta = \]

\[ \frac{1}{2} \int_{|\xi| > \frac{\mu}{2\pi}} g(\tilde{\xi})e^{-\mu x \cdot \tilde{\xi}^{\perp} + 2\pi i x \cdot \tilde{\xi}} dH(\tilde{\xi}) = \frac{1}{2} \int_{|\xi| > \frac{\mu}{2\pi}} g(\tilde{\xi})E^*(\tilde{\xi})dH(\tilde{\xi}). \]

The formula (5) now follows by (4), (7) and the above computation. The subsequent formula is an immediate consequence since \( H \) equals Lebesgue measure on each copy of the punctured plane, and \( \tilde{\xi}^{\perp} = (\xi, \alpha)^{\perp} \) has opposite directions depending on whether \( \alpha = 1 \) or \( -1 \), so the evaluation of 5 reduces to

\[ \int_{|\xi| > \frac{\mu}{2\pi}} \int_{|\xi| > \frac{\mu}{2\pi}} f(y) e^{\mu(y-x) \cdot \xi^{\perp} - 2\pi i(y-x) \cdot \xi} dy d\xi + \]

\[ \int_{|\xi| > \frac{\mu}{2\pi}} \int_{|\xi| > \frac{\mu}{2\pi}} f(y) e^{-\mu(y-x) \cdot \xi^{\perp} - 2\pi i(y-x) \cdot \xi} dy d\xi = \]

\[ \int_{|\xi| > \frac{\mu}{2\pi}} \int f(y) \cosh(\mu(y-x) \cdot \xi^{\perp}) e^{-2\pi i(y-x) \cdot \xi} dy d\xi. \]

By the formulas (4) and (5) it is clear that both \( \mathcal{R}_\mu \) and the inversion formula can be evaluated rapidly by using fast Laplace transforms, and this will be explained in greater detail in the coming sections. For the sake of dealing e.g. with incomplete data, we first consider the structure of the formula (5) a bit more in detail. Consider the operator

\[ \mathcal{T}_\nu f = \int \langle f, E(\cdot, \tilde{\xi}) \rangle \nu(\tilde{\xi})E^*(x, \tilde{\xi})dH(\tilde{\xi}), \]  

(9)

(8)

(7)

(6)

(5)

(4)

(3)

(2)

(1)

(0)

(9)

(8)

(7)

(6)

(5)

(4)

(3)

(2)

(1)

(0)
(compare with Theorem 1 in [34]). For every choice of \( v \) the effect of applying \( \mathcal{T}_v \) can thus be described in terms of a convolution, where the kernel only depend on \( v \). Theorem 2.1 can now be rephrased as \( \delta_0 = T_v \) with \( v(\xi) = \frac{1}{2} \mathbf{1}_{|\xi| > \frac{\pi}{\mu}} \) and \( \delta_0 \) denoting the Dirac measure at 0. Formula (8) is of particular interest when \( v \) is chosen to model incomplete measurements, e.g., incomplete angles [2, 29] of not dense enough sampling in the radial direction. Since (8) can be efficiently evaluated using the fast Laplace transform [3], this observation can prove useful for designing iterative methods to remove artifacts due to incomplete sampling in a similar fashion as was suggested for the case of unequally spaced Fourier data, e.g., magnetic data interpolation [7], and in synthetic aperture radar [4].

3 Discretization

We will now consider discretized versions of the exponential Radon transform and its adjoint. The discretization will be fairly standard, using an equally spaced sampling in the spatial coordinates as well as in the \((s, \theta)\)-coordinates. We assume that \( f \) has compact support. Without loss of generality we assume that it has support on the disc \( \{x : |x| < 1/2\} \), and we represent it by sampling on the Cartesian lattice

\[
X = \left[ -\frac{N}{2}, \ldots, \frac{N}{2} - 1 \right] \times \left[ -\frac{N}{2}, \ldots, \frac{N}{2} - 1 \right] \quad \text{for } N \text{ even,}
\]

and let \( \Pi_X \) denote the operator that samples functions on \( X \). We also discretize \( \mathcal{R}_\mu f(s, \theta) \) at equally spaced points in \( \theta \) that cover the whole interval \([0, 2\pi)\). Specifically, let us assume that we have samples at a grid with \( N \) number of points in the \( s \)-direction and \( N_\theta \) points in the \( \theta \)-direction

\[
s_m = \frac{m}{N}, \quad m = -\frac{N}{2}, \ldots, \frac{N}{2} - 1, \\
\theta_l = 2\pi \frac{l}{N_\theta}, \quad l = 0, \ldots, N_\theta - 1.
\]

and define

\[
S = \left\{ (s_m, \theta_l) : m = -\frac{N}{2}, \ldots, \frac{N}{2} - 1, l = 0, \ldots, N_\theta - 1 \right\}.
\]
Fast Laplace transforms for fast inversion of the exponential Radon transform

Via the discrete Fourier transform, the nodes \( s_m \) are associated with frequency nodes

\[
\sigma_k = k, \quad k = -\frac{N}{2}, \ldots, \frac{N}{2} - 1
\]
as well as a corresponding polar grid,

\[
\Sigma = \left\{ (\sigma_k, \theta_l) : k = -\frac{N}{2}, \ldots, \frac{N}{2} - 1, l = 0, \ldots, N\theta - 1 \right\}.
\]

We let \( \mathbb{I}_S \) and \( \mathbb{I}_\Sigma \) denote the sampling operators on the two sets. As usual customary in this field, we identify \( \theta_l \) with the point \((\cos(\theta_l), \sin(\theta_l))\) on \( S^1 \), wherever convenient.

### 3.1 Fast computation of the discretized \( \mathcal{R}_\mu \)

We now compute a discretized version of the exponential Radon transform (1) in two steps using two-dimensional fast Laplace transforms (3) and FFT. A discrete counterpart for computing \( \mathbb{I}_\Sigma \hat{f}(\sigma \theta + \frac{i\mu}{2\pi} \theta^\perp) \) using \( \mathbb{I}_X f \) is given by the operator \( \mathcal{L}_d : \mathbb{R}^X \rightarrow \mathbb{C}^\Sigma \), defined by

\[
\mathcal{L}_d(\mathbb{I}_X f)(\sigma_k, \theta_l) = \sum_{n_1, n_2 = -\frac{N}{2}}^{N-1} f\left(\frac{n_1}{N}, \frac{n_2}{N}\right) e^{(-\mu \theta_l^\perp - 2\pi i \theta_l \sigma_k)\left(\frac{n_1}{N}, \frac{n_2}{N}\right)}.
\]

If \( \mu = 0 \) then sums of the above type can be approximately evaluated (with precision \( \epsilon \)) in \( \mathcal{O}(N^2 \log N + NN\theta \log(1/\epsilon)) \) by using unequally spaced fast Fourier transforms (USFFT): [6, 15, 16]. To account for the case where \( \mu \neq 0 \), we will instead use unequally spaced fast Laplace transforms [3] (USFLT), at the same computational cost and the same precision dependence as for the USFFT. We will denote the corresponding operator \( \mathcal{L}_d^{\text{USFLT}} \). See Appendix 6 for more information on this topic in two dimensions.

Once \( \hat{f}(\sigma_k \theta + \frac{i\mu}{2\pi} \theta^\perp) \) is (approximately) computed, a discrete version of \( \mathcal{R}_\mu(f)(s_m, \theta_l) \) can be obtained by a one-dimensional discrete inverse Fourier
transform

\[ R_\mu f(s_m, \theta_l) \approx \frac{1}{N} \sum_{k=-N/2}^{N-1} \hat{f}(\sigma_k \theta_l + \frac{i \mu}{2\pi} \theta_l^{\perp}) e^{2\pi i \sigma_k s_m} \approx \]

\[ \frac{1}{N} \sum_{k=-N/2}^{N-1} \mathcal{L}_d(\mathcal{F}_X f)(\sigma_k, \theta_l) e^{2\pi i \sigma_k s_m}. \]

We denote the one-dimensional discrete inverse Fourier transform acting on the first variable by \( \mathcal{F}_d^{-1} : \mathbb{C}^\Sigma \to \mathbb{C}^S \), i.e.

\[ \mathcal{F}_d^{-1} g(s_m, \theta_l) = \sum_{k=-N/2}^{N-1} g(\sigma_k, \theta_l) e^{2\pi i \sigma_k s_m}. \]

Summing up, we can approximately compute \( \mathcal{F}_d^{-1} \mathcal{R}_\mu f \) by \( \mathcal{F}_d^{-1} \mathcal{L}_d^{\text{USFLT}} \mathcal{F}_X f \).

An analysis of the errors involved in this approximation is found in Appendix 3.2.

### 3.2 Error analysis

The total error of our discretization/approximation is

\[ \| \mathcal{F}_d^{-1} \mathcal{R}_\mu f - \mathcal{F}_d^{-1} \mathcal{L}_d^{\text{USFLT}} \mathcal{F}_X f \|_2 \]  

(11)

which by the triangle inequality and the fact that DFT is isometric can be estimated by the sum of the error caused in the discretization of the one-dimensional Fourier integrals,

\[ \| \mathcal{F}_d^{-1} \mathcal{R}_\mu f - \mathcal{F}_d^{-1} \mathcal{F}_X f \|_2 = \| \mathcal{F}_d^{-1} \mathcal{L}_d^{\text{USFLT}} \mathcal{F}_X f \|_2 \]

(12)

the discretization error for the Laplace transform

\[ \| \mathcal{F}_d^{-1} \mathcal{F}_X f - \mathcal{F}_d^{-1} \mathcal{F}_X f \|_2 \]

(13)

and the error due to the fast evaluation of the discrete Laplace transform

\[ \| (\mathcal{L}_d - \mathcal{L}_d^{\text{USFLT}}) \mathcal{F}_X f \|_2 \]

(14)
Fast Laplace transforms for fast inversion of the exponential Radon transform

The first two errors are due to the discretization, and the third one is due to
the proposed fast computational algorithm. As mention above, the error in
(14) will be bounded by the choice of precision parameter \( \epsilon \), see Section 6.
We will now show that the errors in (12) and (13) bear similarities to standard
sampling results about essentially bandlimited functions.

To this end we introduce

\[
B_{\mu,N}(f) = \sup_{\max(|\omega_1|,|\omega_2|) \leq \frac{N}{4}, \max(|\sigma_1|,|\sigma_2|) \leq \mu, \ n_1, n_2 \neq 0} \left\{ \sum_{n_1, n_2 \in \mathbb{Z}} \left| \hat{f} \left( \omega_1 + \frac{\sigma_1}{2\pi i} + n_1 N, \omega_2 + \frac{\sigma_2}{2\pi i} + n_2 N \right) \right| \right\},
\]

as a measurement of the frequency content of \( f \) outside \([-N/2,N/2]^2\). Note
that for bandlimited functions whose frequency support is contained in the
central square with side length \( N \) it holds that \( B_{0,N}(f) = 0 \).

**Proposition 3.1.** Assume that \( f \) has support in \([-1/2,1/2]^2\). The errors (12)
and (13) are then both bounded by a constant times \( B_{\mu,N}(f) \).

**Proof.** By (3) it follows that

\[
\mathcal{R}_\mu \hat{f}(s_m, \theta_l) - \mathcal{F}_d^* \left( \mathbb{I} \mathcal{P} \hat{f}(\sigma \theta + \mu \theta \perp) \right)(s_m, \theta_l) =
\mathcal{F}_\sigma^* \left( \hat{f}(\sigma \theta + \mu \theta \perp) \right)(s_m, \theta_l) - \mathcal{F}_d^* \left( \mathbb{I} \mathcal{P} \hat{f}(\sigma \theta + \mu \theta \perp) \right)(s_m, \theta_l).
\]

Set \( g(\sigma) = \hat{f}(\sigma \theta + \mu \theta \perp) \) and note that its frequency support lies within
\((-1/2,1/2)\). The Poisson summation formula yields

\[
\mathcal{F}_\sigma^* \hat{g}(s) = \sum_{k=-\infty}^{\infty} g(k) e^{2\pi i k s},
\]

where the summation from \(-N/2 \) to \( N/2 - 1 \) corresponds to \( \mathcal{F}_d^* \). It is now
easy to see that the difference (16) is bounded by \( B_{\mu,N}(f) \), yielding an \( \ell^\infty \)
estimate of (12). The corresponding \( \ell^2 \) estimate follows by the equivalence
of finite dimensional norms.
The estimate for (13) is similar. By the two-dimensional Poisson summation formula (applied to $f(x)e^{-\mu \theta \cdot x}$) we have

$$
\frac{1}{N^2} \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f\left(\frac{n_1}{N}, \frac{n_2}{N}\right)e^{(-\mu \theta \cdot \frac{2\pi i}{N} + N(n_1, n_2)} = \\
\sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \hat{f}\left(\theta_l \sigma_k + \frac{\mu \theta \cdot \frac{2\pi i}{N}}{N^2} + N(n_1, n_2)\right).
$$

By the assumption on the support of $f$, the first double sum can be replaced by $\sum_{N/2}^{N/2} \sum_{N/2}^{N/2}$, so the first line equals $\mathcal{L}_d(\mathrm{III}_X f)(\sigma_k, \theta_l)$. It follows that

$$
\left| \hat{f}(\sigma_k \theta_l + \mu \theta \cdot \frac{2\pi i}{N}) - \mathcal{L}_d(\mathrm{III}_X f)(\sigma_k, \theta_l) \right|
\leq \left| \sum_{n_1 \neq 0} \sum_{n_2 \neq 0} \hat{f}\left(\theta_l \sigma_k + \frac{\mu \theta \cdot \frac{2\pi i}{N}}{N^2} + N(n_1, n_2)\right) \right|
$$

which yields a pointwise estimate of the difference in (13). The desired conclusion again follows by the equivalence of finite dimensional norms.

3.3 Fast computation of the discretized adjoint

The back-projection operator $\mathcal{R}_\mu^*$ in (2) can be discretized in a similar way. Note that

$$
\mathcal{R}_\mu^* g(x) = \\
\int_{S_1} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(s, \theta) e^{-2\pi i s \sigma} d s e^{x(2\pi i \sigma \theta + \mu \theta \cdot \frac{2\pi i}{N})} d \sigma d \theta = \left(\hat{\mathcal{R}}_\mu^* \mathcal{F}_{s \rightarrow \sigma} g\right)(x)
$$

which follows from (2) and the fact that

$$
g(x \cdot \theta, \theta) = \mathcal{F}_{\sigma \rightarrow x, \theta}^* \mathcal{F}_{s \rightarrow \sigma}(g(s, \theta)).$$

In the discretized setting, the inner integral (i.e. $\mathcal{F}_{s \rightarrow \sigma}$) corresponds to $\mathcal{F}_d$ whereas the two outer integrals (i.e. $\hat{\mathcal{R}}_\mu^*$) correspond to $\left(\mathcal{L}_d^\text{USFLT}\right)^\ast$. We remark that the latter do not correspond to a two-dimensional Fourier-Laplace
Fast Laplace transforms for fast inversion of the exponential Radon transform

integral (since the factor $|\sigma|$ is missing), but it can still be evaluated fast, i.e., in $O(N^2 \log N)$ time. In the terminology of the appendix, the USFLT operation changes from a $\mathbb{Z}^2 \to \mathbb{C}^2$ operation (for $\mathcal{L}_d^{USFLT}$) to a $\mathbb{C}^2 \to \mathbb{Z}^2$ operation (for $(\mathcal{L}_d^{USFLT})^*$), designed to rapidly evaluate sums of the type

$$
\sum F_{k,l} e^{(\mu \theta_k^1 - 2\pi i \sigma_k \theta_k^1)(\frac{n_1}{N}, \frac{n_2}{N})},
$$

(18)

for all values $(\frac{n_1}{N}, \frac{n_2}{N})$ in $X$, where $F_{k,l}$ can be arbitrary numbers. For the evaluation of $\mathcal{I}_S \mathcal{H}_\mu g$ via $(\mathcal{L}_d^{USFLT})^* \mathcal{F}_d \mathcal{I}_S g$, we obviously have $F = \mathcal{F}_d \mathcal{I}_S g$, but we shall see in the next section that other choices also will be relevant. An error analysis of this approximation can be done in a similar way as in Section 3.2, but we omit this.

3.4 Inversion from complete measurements

To invert measurements of the exponential Radon transform on the polar grid $S$, we use the formula (7);

$$
f = \mathcal{R}_-^* \mathcal{F}^* M_w \mathcal{F} \mu \mathcal{R}_\mu f = \mathcal{H}_\mu^* M_w \mathcal{H}_\mu f
$$

(19)

A discrete approximation $g$ of $\mathcal{H}_\mu f$ on $\Sigma$ is easily computed by applying $\mathcal{F}_d$ to the available data $\mathcal{I}_S \mathcal{R}_\mu (f)$, and the operation $\mathcal{H}_\mu^*$ corresponds to $(\mathcal{L}_d^{USFLT})^*$. Since $M_w$ is multiplication by the weight $w$, the inversion could in principle be done via

$$
f \approx (\mathcal{L}_d^{USFLT})^* M_w g,
$$

but the discontinuities of $w$ will introduce large errors for low-frequency components of $f$. To compensate for this, we will evaluate $\mathcal{H}_\mu f$ on a polar grid $\Sigma$ with a more dense radial sampling near the singularities $\sigma = \pm \frac{i \mu}{2\pi}$. The operator $(\mathcal{L}_d^{USFLT})^*$ computes sums of the type (18) over unequally spaced grids, and in particular there is no need to have equally spaced sampling in $\sigma$. However, if we increase the sampling near the singularity, the corresponding sum (18) will not be a good approximation of the corresponding integral in $\mathcal{H}_\mu^*$, and we therefore need to compensate for this by modifying the weight.
w, using composite rules for quadratures. We briefly outline this in more detail below.

In practice, the bottleneck of the reconstruction algorithm lies in evaluating $(\mathcal{L}_d^{\text{USFTL}})^*$, and hence it is desirable to keep the total number of grid points low. We therefore propose to use a dense sampling in $\sigma$ near the singularities $\pm \frac{\mu}{2\pi}$. We remark that values of $\frac{\mu}{2\pi}$ typically are less than 2, and that the original grid $\Sigma$ is sampled with radial sampling distance of unit length. An increased sampling near $\frac{\mu}{2\pi}$ can thus be achieved by increasing the amount of grid points in $\Sigma$ near the origin. Recall that $f$ is assumed to have support on the unit circle, which implies that $f(\sigma, \theta)$ is supported on $[-1/2, 1/2]$ in the $s$-variable. By zero-padding of $f(\sigma, \theta)$ in the $s$-direction and application of FFT, an oversampled representation of $g(\sigma, \theta)$ can be obtained (on an equally spaced grid in $\sigma$, but not all values will be used).

To derive appropriate weights, it is sufficient to consider accurate quadratures for

$$\int_{|\sigma|>\frac{\mu}{2\pi}} \left| \frac{\sigma}{2} \right| g(\sigma, \theta)e^{-\mu x \cdot \theta^\perp + 2\pi i \sigma x \cdot \theta} d\sigma = \int_{|\sigma|>\frac{\mu}{2\pi}} |\sigma| h(\sigma),$$

where we fix $\theta$ and set $h(\sigma) = \frac{1}{2} g(\sigma, \theta)e^{-\mu x \cdot \theta^\perp + 2\pi i \sigma x \cdot \theta}$. Concerning the region around the singularity $\sigma = \frac{\mu}{2\pi}$, we construct a composite quadrature by considering

$$\int_a^b \sigma h(\sigma) d\sigma,$$  \hspace{1cm} (20)

for several small intervals $[a, b]$, containing, say, $M < 10$ grid points. On each such interval there are coefficients $c = \{c_m\}_{m=0}^{M-1}$ such that

$$h(\sigma) \approx \sum_m c_m \sigma^m$$ \hspace{1cm} (21)

is an $M$th-degree polynomial approximation of $h$ on the interval. These coefficients can be found by solving $h(\sigma_k) = \sum c_m \sigma^m_k$ (for $k$ ranging in the relevant range for $[a, b]$), or in matrix form $h = V c$, where matrix $V = V(m, k)$ is the Vandermonde matrix built up by $\sigma^m_k$.

Upon inverting $V$, which is independent of $h$, we do not need to compute
the coefficients explicitly. Inserting this approximation in (20) gives

\[ \int_a^b \sigma h(\sigma) d\sigma \approx \int_a^b \sum_m \sum_k V^{-1}(m,k)h(\sigma_k)\sigma^{m+1} d\sigma = \sum_k \sum_m V^{-1}(m,k)h(\sigma_k) \left( \frac{b^{m+2}}{m+2} - \frac{a^{m+2}}{m+2} \right) = \sum_k \omega_k h(\sigma_k). \]

In this way, we obtain quadrature weights \( \omega_k \) for the equally spaced approximation of (20). The weights can be combined to form composite quadrature rules that take the singularity at \( \sigma = \frac{\mu}{2\pi} \) into account. The same procedure can be performed when dealing with the singularity at \( \sigma = -\frac{\mu}{2\pi} \). The left panels of Figure 1 demonstrate the effect of the correction with the weights \( \omega_k \) near both the singularities.

Figure 1: Weights for the approximation of the integral \( \int_{|\sigma| \geq \frac{\mu}{2\pi}} |\sigma| h(\sigma) d\sigma \), scaling by layers. Left panel: weights after trapezoidal rule correction near singularities \( |\sigma| = \frac{\mu}{2\pi} \). Right panel: total weights with undersampling for high frequencies. The black and red dots use values at every fourth sample (corresponding to the first two terms in the approximation (22)), while the blue dots use every second sample (corresponding to the third term in (22)).

In order for the approximation step in (21) to be accurate, \( h \) has to be
sufficiently oversampled. However, this is only needed in a vicinity of the singularity, and a regular sampling is sufficient away from the singularity. A direct usage of a trapezoidal rule would, however, use a sampling that is determined by the needed sampling rate around the singularity. We therefore split the integral as

\[
\int_{\frac{\mu}{2\pi}}^{b} \sigma h(\sigma) d\sigma = \int_{\frac{\mu}{2\pi}}^{c} \sigma h(\sigma) v(\sigma) d\sigma + \int_{d}^{b} \sigma h(\sigma) (1 - v(\sigma)) d\sigma,
\]

where \(d < c\), \(v\) is a smooth function satisfying \(0 \leq v(\sigma) \leq 1\), \(v(\sigma) = 1\) if \(\sigma < d\), \(v(\sigma) = 0\) if \(\sigma > c\). Now, the derived quadrature rule with a dense sampling can be used for the left integral in the right-hand side above, while a reduced sampling density can be used on the right integral. This procedure can also be used several times to gradually decrease the sampling rate. In the numerical simulations presented in Section 4, we use an oversampling factor of four to account for the singularity at \(\frac{\mu}{2\pi}\), and make sampling reduction in two steps, i.e., by a quadrature rule of the form

\[
\int_{|\sigma| \geq \frac{\mu}{2\pi}} |\sigma| h(\sigma) d\sigma \approx \sum_{|k| < K_{1}} w_{4k} h(\sigma_{4k}) + \sum_{|k| < K_{2}} w_{4k+2} h(\sigma_{4k+2}) + \sum_{|k| < K_{3}} w_{2k+1} h(\sigma_{2k+1}),
\]

where \(K_{3} \leq K_{2} << K_{1}\). The setup is illustrated in the right panels of Figure 1.

### 4 Numerical simulations

In this section reconstruction results and computational speed tests are presented. C++ routines were written for the fast exponential Radon transform and the fast Laplace transform along with MEX interfaces to MATLAB. For accelerations we used the tools from Intel Composer XE 2016, including the Intel Math Kernel Library (MKL), Intel Performance Primitives (IPP), and OpenMP API. For the simulation we used a standard desktop with an Intel i7-3820K processor and eight OpenMP threads, and the computations were performed in single precision.

To begin with, accuracy tests were performed on the Shepp-Logan [30] phantom, where we use the modified version available in MATLAB. The
Fast Laplace transforms for fast inversion of the exponential Radon transform

Figure 2: Weighting with factor $e^{\mu t}$ for a line through the phantom image. Exponential weight $\gamma(t) = e^{\mu t}$ is plotted by the dashed line, ratios between maximal and minimal weights equal to 1, 10, 100, 1000, respectively.

phantom consists of linear combinations of characteristic functions of ellipses, and its support is inscribed in a circle, see the left panel of Figure 2. The red line is depicted with correspondence to a specific angle $\theta$ and polar sample $s$. Values through this line multiplied by factor $e^{\mu t}$ are plotted in the left panel of Figure 2. The upper part of Figure 3 show the exponential Radon transform for different values of the exponential parameter $\mu$.

In order to make accuracy comparisons, we do not carry out comparisons with the original function, but use instead a slightly smoothed out version of the phantom. This is to account for the fact that the characteristic function of the ellipses does not decay rapidly in the frequency domain, and hence high-
Figure 3: Exponential Radon data of the Shepp-Logan phantom for different values of the exponential parameter $\mu$. Corresponding reconstruction with the FBP method.
Fast Laplace transforms for fast inversion of the exponential Radon transform

Figure 4: Inversion of the data originating from $\mu = \log(1000)$, for different values of $\mu$ in the reconstruction. The second line shows corresponding absolute reconstruction errors.
frequency errors would be dominant in the error plots. A reconstruction method using only a finite frequency range should have as aim to recover that frequency range accurately, and it is therefore natural to measure the accuracy in this way. The minor smoothing effect can be noted in the plots of Figure 2.

The lower part of Figure 3 illustrates reconstruction errors for different values of $\mu$ chosen to be the same as for the corresponding projection from the upper part of the same Figure. To show how the choice of $\mu$ affects the reconstruction we compute reconstructions using the fast Laplace-based filtered back-projection. For the data from the right top panel of Figure 3 (the case where $\mu = \log(1000)$), we present reconstruction results using different (incorrect) values of the exponential weighting parameter. Figure 4 shows how the reconstructions are distorted depending on incorrect selection of $\mu$. We can also see that the reconstruction is accurate for the correct value of $\mu$, and how incorrect values of the attenuation parameter gives rise to low-frequent artifacts. In the filtered back-projection method the high-frequency components are boosted, and boosting of high frequent noise is also caused by the exponential factor in the back-projection operator. In particular for high attenuation parameters, this reconstruction technique becomes sensitive to noise. This is illustrated in Figure 5, which shows filtered-back-projection reconstruction results for varying values of the attenuation parameter $\mu$. The Radon data for these different attenuation values are contaminated with white Gaussian noise with a signal-to-noise ratio (SNR) of 20 dB.

From this example it becomes clear that additional regularization methods should be considered when dealing with noisy measurements. There are an abundance of techniques for doing this, and as mentioned in the introduction the aim of this paper is restricted to constructing fast algorithms for the exponential Radon transforms. The kind of preferable regularization technique will typically depend on the data that is considered. We mention some suggestions on regularization techniques that have been made in the literature, either for the standard or the exponential Radon transform.

One of the most direct ways of regularization is by the choice of filter. It is standard to make use of filters that suppress the very highest frequencies, for instance using Shepp-Logan or Hamming filters [18, 24].

Another simple alternative is to employ Tikhonov regularization [23, 31, 32]. In [14] a modified Tikhonov regularization scheme is proposed to deal
Figure 5: Exponential Radon data of the Shepp-Logan phantom for different values of the exponential parameter $\mu$, with noise contributed. Corresponding reconstruction with the FBP method.
Table 1: Computational times for the exponential Radon transform for images of size $(N \times N)$ with $N_\theta = \lceil \pi N \rceil$.

<table>
<thead>
<tr>
<th>N</th>
<th>$t_{\text{USFLT}}$</th>
<th>$t_{\text{Direct}}$</th>
<th>$t_{\text{USFLT}}/(N^2 \log N)$</th>
<th>$t_{\text{USFLT}}/N^2$</th>
<th>$t_{\text{Direct}}/N^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^7$</td>
<td>1.8e-02</td>
<td>1.4e-01</td>
<td>2.3e-07</td>
<td>1.1e-06</td>
<td>6.5e-08</td>
</tr>
<tr>
<td>$2^8$</td>
<td>3.5e-02</td>
<td>7.4e-01</td>
<td>9.6e-08</td>
<td>5.3e-07</td>
<td>4.4e-08</td>
</tr>
<tr>
<td>$2^9$</td>
<td>1.2e-01</td>
<td>5.8e+00</td>
<td>7.6e-08</td>
<td>4.8e-07</td>
<td>4.3e-08</td>
</tr>
<tr>
<td>$2^{10}$</td>
<td>5.2e-01</td>
<td>4.9e+01</td>
<td>7.1e-08</td>
<td>4.9e-07</td>
<td>4.5e-08</td>
</tr>
<tr>
<td>$2^{11}$</td>
<td>2.2e+00</td>
<td>3.9e+02</td>
<td>7.0e-08</td>
<td>5.3e-07</td>
<td>4.6e-08</td>
</tr>
<tr>
<td>$2^{12}$</td>
<td>8.7e+00</td>
<td>3.2e+03</td>
<td>6.3e-08</td>
<td>5.2e-07</td>
<td>4.7e-08</td>
</tr>
<tr>
<td>$2^{13}$</td>
<td>3.5e+01</td>
<td>2.7e+04</td>
<td>5.7e-08</td>
<td>5.2e-07</td>
<td>4.9e-08</td>
</tr>
</tbody>
</table>

with the attenuated Radon transform, and error estimates and stability results for this approach are presented. For another inversion approach where Tikonov regularization is applied to exponential Radon data, see [35].

Iterative methods such as the EM algorithm can also be useful for suppressing noise [13, 19, 36]. Here the filter in the filtered back-projection is not used, but instead the exponential Radon transform and its adjoint are iteratively applied several times to obtain a reconstruction.

As an alternative regularization technique, one can use a sparse data representation by using tight frames. In [21] the tight frame of wavelets is proposed for decomposition of the Radon operator. The reconstruction is then based on thresholding of the noisy wavelet coefficients. Similar procedure of the Radon data inversion is performed by using other types of frames: curvelets [9], shearlets [12], Gabor frames [10]. Another useful approach is to enforce total variation penalties [11]. This approach is used in [39, 17, 22] for noise suppression in the reconstruction from Radon data. This method favors piecewise constant reconstructions.

Let us now turn our attention to the computational speed of the reconstruction method. Tables 1 and 2 show the total times of the simulations and the times per operation (to verify the time complexity) of applying the exponential Radon transform and backprojection, respectively, for different image sizes.

The time complexity of USFLT is $O(N^2 \log N + (2M + 1)^2) (2M + 1) N^2$ (see Appendix), and that is the main computational part of the exponential
Table 2: Computational times for backprojection of the exponential Radon transform for images of size $(N \times N)$ with $N_\theta = \lceil \pi N \rceil$.

<table>
<thead>
<tr>
<th>N</th>
<th>$t_{\text{USFLT}}$</th>
<th>$t_{\text{Direct}}$</th>
<th>$t_{\text{USFLT}}/(N^2 \log N)$</th>
<th>$t_{\text{USFLT}}/N^2$</th>
<th>$t_{\text{Direct}}/N^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^7$</td>
<td>4.2e-02</td>
<td>1.2e-01</td>
<td>5.3e-07</td>
<td>2.6e-06</td>
<td>5.8e-08</td>
</tr>
<tr>
<td>$2^8$</td>
<td>9.1e-02</td>
<td>7.6e-01</td>
<td>2.5e-07</td>
<td>1.4e-06</td>
<td>4.5e-08</td>
</tr>
<tr>
<td>$2^9$</td>
<td>2.5e-01</td>
<td>6.3e+00</td>
<td>1.5e-07</td>
<td>9.5e-07</td>
<td>4.7e-08</td>
</tr>
<tr>
<td>$2^{10}$</td>
<td>7.5e-01</td>
<td>5.2e+01</td>
<td>1.0e-07</td>
<td>7.1e-07</td>
<td>4.9e-08</td>
</tr>
<tr>
<td>$2^{11}$</td>
<td>2.9e+00</td>
<td>4.3e+02</td>
<td>9.1e-08</td>
<td>6.9e-07</td>
<td>5.0e-08</td>
</tr>
<tr>
<td>$2^{12}$</td>
<td>1.1e+01</td>
<td>3.8e+03</td>
<td>7.6e-08</td>
<td>6.4e-07</td>
<td>5.5e-08</td>
</tr>
<tr>
<td>$2^{13}$</td>
<td>4.6e+01</td>
<td>3.1e+04</td>
<td>7.7e-08</td>
<td>6.9e-07</td>
<td>5.6e-08</td>
</tr>
</tbody>
</table>

Radon transform. Typically $M$ is about 8-10 for single precision in the case where $\mu$ represents variations of up to a factor of $\log(1000)$, therefore in practice the factor $(2M+1)(2M+1)$ will dominate the $\nu^2 \log N$ factor. This effect can be seen in the ratio columns of Table 1 and 2. We also verify that the USFLT-based method outperforms the direct method as expected.

5 Conclusions

We have shown how to construct fast algorithms for the computation of the exponential Radon transform and the associated (adjoint) back-projection operator by using algorithms for fast Laplace transforms. In addition we have included estimates for the discretization errors that arise, and shown how to separate them into parts where one bears similarities to standard sampling arguments of (almost) bandlimited functions, and where the other part is cause be the (fast) approximate evaluation of discrete Laplace transforms. The latter part can be controlled at arbitrary numerical precision.

6 Acknowledgments

This work was supported by the Crafoord Foundation (2014-0633) and the Swedish Research Council (2011-5589, 2015-03780).
Appendix; Computation of $\mathcal{L}^{\text{USFLT}}_d$

For the sake of completeness, we present algorithms for fast evaluation of two-dimensional discrete Laplace transforms. These algorithms are analogous to the one-dimensional ones presented in [3], but in this appendix they are adapted for computing the exponential Radon transforms.

Let $n_1, n_2$ and $\tilde{\xi}_j$ be as in Section 3. We describe how to fast evaluate the operations

$$f = \mathcal{L}_{\mathbb{Z}^2 \to \mathbb{Z}^2}(F) : \quad f_{n_1, n_2} = \sum_j F_j e^{(\mu \tilde{\xi}_{1j} - 2\pi i \xi_{1j}) \frac{n_1}{N} + (\mu \tilde{\xi}_{2j} - 2\pi i \xi_{2j}) \frac{n_2}{N}},$$

(23)

$$F = \mathcal{L}_{\mathbb{Z}^2 \to \mathbb{C}^2}(f) : \quad F_j = \sum_{n_1, n_2} f_{n_1, n_2} e^{(\mu \tilde{\xi}_{1j} - 2\pi i \xi_{1j}) \frac{n_1}{N} + (\mu \tilde{\xi}_{2j} - 2\pi i \xi_{2j}) \frac{n_2}{N}}.$$  

(24)

where $f_{n_1, n_2}$ and $F_j$ represent any complex numbers. Note that with $f_{n_1, n_2} = f \left(\frac{n_1}{N}, \frac{n_2}{N}\right)$ and $F_j = \hat{f} \left(\xi_j + \frac{i\mu \tilde{\xi}_j}{2\pi}\right)$, the operation (24) corresponds to the evaluation of (10) in Section 3, whereas (23) is needed in (18).

Methods for unequally spaced fast Laplace transforms (USFLT) utilize convolution style operations with FFT to achieve approximations of the sums (23, 24) with arbitrary, but fixed, accuracy. We make a heuristic description of how USFLT works, for a more precise formulation along with error estimates, see [3].

Let $\mathcal{F}$ denote the Fourier transform and introduce the Gaussian $\varphi(x) = e^{-\alpha(x_1^2 + x_2^2)}$ and modulated Gaussian $\varphi_{\tilde{\xi}_j}(x) = e^{\mu \tilde{\xi}_{1j} x_1 + \mu \tilde{\xi}_{2j} x_2} \varphi(x)$, where $\alpha$ is a parameter which we discuss how to choose below. Note that in the frequency domain it holds that

$$\hat{\varphi}_{\tilde{\xi}_j}(v_1, v_2) = \frac{\pi}{\alpha} e^{-\frac{x^2}{\alpha}} \left(\left(v_1 - \frac{\mu \tilde{\xi}_{1j}}{2\pi}\right)^2 + \left(v_2 - \frac{\mu \tilde{\xi}_{2j}}{2\pi}\right)^2\right),$$

i.e. $\hat{\varphi}_{\tilde{\xi}_j}$ is a frequency modulated Gaussian. Next, by choosing $g = \frac{f}{\varphi}$ one
Fast Laplace transforms for fast inversion of the exponential Radon transform can derive the relation
\[
\hat{f}(\mu \tilde{\xi}^j - 2\pi i \xi_j) = \int_{\mathbb{R}^2} f(x) e^{(\mu \tilde{\xi}^j - 2\pi i \xi_j) \cdot x} dx = \mathcal{F}\left( \frac{f}{\varphi_{\xi_j}^\perp} \right)(\xi_j) = \\
\hat{g} \ast \hat{\varphi}_{\xi_j}^\perp(\xi_j) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{g}(v_1, v_2) \hat{\varphi}_{\xi_j}^\perp(v_1 - \xi_{1j}, v_2 - \xi_{2j}) dv_1 dv_2, 
\]

(25)

The integral in (25) can be approximated by the trapezoidal rule because of the fact that \( \hat{\varphi}_{\xi_j}^\perp \) are smooth functions that decay rapidly to zero (the amount of oscillation is limited by the parameter \( \mu \)). It turns out that accurate results can be obtained by \((v_1, v_2)\) on an equally spaced lattice with some oversampling parameter \( \nu \). Crucial for the construction of fast algorithms is that the function \( \hat{\varphi}_{\xi_j}^\perp \) has small (numerical) support, since this will keep down the computational times for convolutions with \( \hat{\varphi}_{\xi_j}^\perp \). It turns out that the (numerical) width of \( \hat{\varphi}_{\xi_j}^\perp \) is proportional to \(-\log \epsilon\), where \( \epsilon \) is the desired precision. More precisely, values of \( \hat{\varphi}_{\xi_j}^\perp \) that are smaller than a certain threshold level

\[
T_\epsilon = \frac{1}{\pi} \sqrt{-\alpha \log \epsilon + \frac{\alpha^2}{4} + \frac{\mu^2}{4}},
\]

can be omitted. In practice we thus replace \( \hat{\varphi}_{\xi_j}^\perp \) by

\[
\hat{\varphi}_{\xi_j}^\epsilon(v_1, v_2) = \begin{cases} 
\hat{\varphi}_{\xi_j}^\perp(v_1, v_2), & \text{if } |v_1| \leq T_\epsilon \text{ and } |v_2| \leq T_\epsilon, \\
0, & \text{otherwise},
\end{cases}
\]

Now, if we assume that the discrete Fourier transform (realized by FFT) provides an accurate approximation of \( \hat{g} \) at this oversampled grid, the evaluation of (24) can be approximated in three steps:

1. Division in the space domain
2. FFT
3. Convolution-type operation in the frequency domain
We refer to [3] for proofs and more details, and remark that \( \mu = \log A \) in the notation of that paper.

The approximations to prescribed precision \( \epsilon \) can be achieved by a suitable choice of \( \alpha \) in combination with proper oversampling \( \nu \). It then holds that the difference between the sum of (23) and that described by the above algorithm can be bounded for instance by

\[
C \epsilon \|f\|_{\ell^2} ,
\]

where \( C \) is a constant that is independent of \( f \). A similar result can also be obtained for the approximation of (24).

Following [3], the parameter \( \alpha \) should be chosen as

\[
\alpha = \frac{2\nu - 1}{2\nu(\nu - 1)} \mu + \frac{1}{\nu(\nu - 1)} (-\log \epsilon),
\]

where \( \nu \) is an oversampling factor for Fourier transform (usually \( \nu = 2 \)). In practice this means that the integral (25) is replaced by a summation over \( k_1, k_2 \in \mathbb{Z} \) using \( \hat{\varphi}^\epsilon \left( \frac{k_1}{\nu} - \xi_{1j}, \frac{k_2}{\nu} - \xi_{2j} \right) \). Note that for each \( j = 1 \ldots J \), these values will be non-zero only for values of \( k_1, k_2 \) such that \( |k_1 - \nu \xi_{1j}| \leq \nu \epsilon \) and \( |k_2 - \nu \xi_{2j}| \leq \nu \epsilon \). By introducing \( M = \lceil \nu \epsilon \rceil \) we find that the contributing values of \( k_1, k_2 \) satisfy \( [\nu \xi_{1j}] - M \leq k_1 \leq [\nu \xi_{1j}] + M \) and \( [\nu \xi_{2j}] - M \leq k_2 \leq [\nu \xi_{2j}] + M \). The notations \([x], [x] \) denote the nearest integer to \( x \) and the smallest integer less or equal to \( x \), respectively.

Finally, we have an approximation formula for (24) with accuracy \( \epsilon \):

\[
F_j = \sum_{n_1, n_2} f_{n_1, n_2} e^{(\mu \xi_{1j} - 2\pi i \xi_{1j}) \frac{n_1}{N} + (\xi_{2j} - 2\pi i \xi_{2j}) \frac{n_2}{N}} \approx \sum_{k_1=-\frac{N}{2}}^{\frac{N}{2}} \sum_{k_2=-\frac{N}{2}}^{\frac{N}{2}} \left( \sum_{n_1, n_2} f_{n_1, n_2} e^{-2\pi i \left( \frac{n_1}{N} + \frac{k_1}{\nu} \right) \xi_{1j} + \frac{n_2}{N} \xi_{2j}} \right) \hat{\varphi}^\epsilon \left( \frac{k_1}{\nu} - \xi_{1j}, \frac{k_2}{\nu} - \xi_{2j} \right),
\]

where the sums over \( k_1, k_2 \) variables consist each of \( (2M + 1) \) non-zero terms, respectively. The constant \( M \) is typically equal to 8-10 for single precision of computations, depending on the choice of \( \mu \) and \( \nu \). Algorithm \( \mathcal{L}_{\mathbb{Z}^2 \rightarrow \mathbb{C}^2}(f) \) describe these computational steps, with the assumption \( |\xi_j| \) are inside a circle slightly smaller than \( \frac{N}{2} \). The first operation (division) has \( \mathcal{O}(N^2) \) computa-
Fast Laplace transforms for fast inversion of the exponential Radon transform

Algorithm $\mathcal{L}_{\mathbb{C}^2 \to \mathbb{Z}^2}$.

1. $G_{k_1, k_2} = \sum_{j=1}^{J} F_j \frac{\phi}{\xi_j} \left( \frac{k_1}{v} - \xi_{1j}, \frac{k_2}{v} - \xi_{2j} \right), \quad (-\frac{vN}{2} \leq k_1, k_2 < \frac{vN}{2})$

2. $h_{n_1, n_2} = \frac{1}{vN^2} \sum_{k_1=-\frac{vN}{2}}^{\frac{vN}{2}-1} \sum_{k_2=-\frac{vN}{2}}^{\frac{vN}{2}-1} G_{k_1, k_2} e^{-2\pi i \left( \frac{k_1 n_1}{vN} + \frac{k_2 n_2}{vN} \right)}, \quad (-\frac{vN}{2} \leq n_1, n_2 < \frac{vN}{2})$

3. $f_{n_1, n_2} = \frac{h_{n_1, n_2}}{\varphi_0(\frac{n_1}{vN}, \frac{n_2}{vN})}, \quad (-\frac{N}{2} \leq n_1, n_2 < \frac{N}{2})$.

In terms of time complexity, the second operation is evaluated by FFT in $O(v^2N^2 \log(N))$ time, and the third operation cost $O((2M+1)^2NN_0)$ operations, since for each $j$ there are only $(2M+1)^2$ non-zero contributions. The total time complexity is thus $O(v^2N^2 \log(N) + (2M+1)^2NN_0)$. In terms of time complexity, this is dominated by FFT part, while in practice most time is typically spent on the third step.

The algorithm for $\mathcal{L}_{\mathbb{C}^2 \to \mathbb{Z}^2}(f)$ (23) can be also constructed by applying the above operations in reverse order, since $\mathcal{L}_{\mathbb{C}^2 \to \mathbb{Z}^2}(f)$ is adjoint to $\mathcal{L}_{\mathbb{Z}^2 \to \mathbb{C}^2}(f)$. These steps are gathered in Algorithm $\mathcal{L}_{\mathbb{Z}^2 \to \mathbb{C}^2}$ below.

Algorithm $\mathcal{L}_{\mathbb{Z}^2 \to \mathbb{C}^2}$.

1. $g_{n_1, n_2} = \frac{f_{n_1, n_2}}{\varphi_0(\frac{n_1}{vN}, \frac{n_2}{vN})}, \quad (-\frac{N}{2} \leq n_1, n_2 < \frac{N}{2})$

2. $G_{k_1, k_2} = \frac{1}{vN^2} \sum_{n_1=-\frac{N}{2}}^{\frac{N}{2}-1} \sum_{n_2=-\frac{N}{2}}^{\frac{N}{2}-1} g_{n_1, n_2} e^{-2\pi i \left( \frac{k_1 n_1}{vN} + \frac{k_2 n_2}{vN} \right)}, \quad (-\frac{vN}{2} \leq k_1, k_2 < \frac{vN}{2})$

3. $F_j = \sum_{k_1=[v\xi_{1j}]-M}^{[v\xi_{1j}]+M} \sum_{k_2=[v\xi_{2j}]-M}^{[v\xi_{2j}]+M} G_{k_1, k_2} \hat{\phi}_{\xi_j} \left( \frac{k_1}{v} - \xi_{1j}, \frac{k_2}{v} - \xi_{2j} \right), \quad (1 \leq j \leq J)$. 

107
Acknowledgments

This work was supported by the Crafoord Foundation (2014-0633) and the Swedish Research Council (2011-5589, 2015-03780)

References


Fast Laplace transforms for fast inversion of the exponential Radon transform


Fast Laplace transforms for fast inversion of the exponential Radon transform


Fast hyperbolic Radon transform represented as convolutions in log-polar coordinates

Viktor V. Nikitin       Fredrik Andersson       Marcus Carlsson
Anton A. Duchkov

Abstract

The hyperbolic Radon transform is a commonly used tool in seismic processing, for instance in seismic velocity analysis, data interpolation and for multiple removal. A direct implementation by summation of traces with different moveouts is computationally expensive for large data sets. In this paper we present a new method for fast computation of the hyperbolic Radon transforms. It is based on using a log-polar sampling with which the main computational parts reduce to computing convolutions. This allows for fast implementations by means of FFT. In addition to the FFT operations, interpolation procedures are required for switching between coordinates in the time-offset; Radon; and log-polar domains. Graphical Processor Units (GPUs) are suitable to use as a computational platform for this purpose, due to the hardware supported interpolation routines as well as optimized routines for FFT. Performance tests show large speed-ups of the proposed algorithm. Hence, it is suitable to use in iterative methods, and we provide examples for data interpolation and multiple removal using this approach.

1 Introduction

In the processing of Common-Midpoint gathers (CMPs), the hyperbolic Radon transform has proven to be a valuable tool for instance in velocity analysis [7, 13]; aliasing and noise removal [24]; trace interpolation [3, 26]; and attenuation of multiple reflections [14]. The hyperbolic Radon transform is
defined as

$$\mathcal{R}_h f(\tau, q) = \int_{-\infty}^{\infty} f\left(\sqrt{\tau^2 + q^2 x^2}, x\right) dx,$$  \hspace{1cm} (1)$$

where the function $f(t, x)$ usually corresponds to a CMP gather. Here, the parameter $q$ characterizes an effective velocity value; and $\tau$ represents the intercept time at zero offset.

Several versions of Radon transforms are used in seismic processing, e.g., straight-line, parabolic, and hyperbolic Radon transforms. In many applications there is a need for a sparse representation of seismic data using hyperbolic wave events. One way to get sparse representations is by using iterative thresholding algorithms with sparsity constraints [8, 21]. Popular applications using such representations are seismic data interpolation and wavefield separation [16, 23]. Since iterative schemes for computing such representations require the application of the forward and adjoint operators several times, it becomes important to use fast algorithms for the realization of the operators to the limit the total computational cost.

Note that the direct summation over hyperbolas in (1) has a computational complexity of $O(N^3)$, given that the numbers of samples for the variables $t, x, \tau, q$ are $O(N)$. There are many effective ($O(N^2 \log N)$) methods for rapid evaluation of the traditional Radon transforms, or the parabolic Radon transform, see [4, 11, 22]. The hyperbolic Radon transform is, however, more challenging. Nonetheless, a fast method for hyperbolic Radon transforms was recently presented in [15]. This method is based on using the fast butterfly algorithms described in [6, 18, 19].

A fast method for the standard Radon transform was proposed in [1] by expressing the Radon transform and its adjoint in terms of convolutions in log-polar coordinates. Computationally efficient algorithms for GPUs were presented for this approach in [2]. In this paper we propose to use the same approach and construct algorithms with complexity $O(N^2 \log N)$ for evaluation of the hyperbolic Radon transform. We present computational performance tests confirming the expected accuracy and the computational complexity, as well as predicted computational speed-ups for parallel implementations. Finally, we present several synthetic and real data tests using the hyperbolic Radon transform for data interpolation and multiple attenuation.
Fast hyperbolic Radon transform represented as convolutions in log-polar coordinates

2 Method

To begin with, we note that functions \( f(t, x) \) describing CMP gathers are symmetric with respect to \( x = 0 \). Hence, by introducing

\[
\tilde{f}(s, y) = \frac{f(\sqrt{s}, \sqrt{y})}{2\sqrt{y}},
\]

it follows that

\[
\mathcal{R}_h f(\tau, q) = 2 \int_0^\infty f(\sqrt{\tau^2 + q^2 x^2}, x) \, dx = 2 \int_0^\infty \tilde{f}(\tau^2 + q^2 y, y) \, dy.
\]

The resulting expression in (3) has a form of the Radon transform over straight lines, and a fast algorithm for the evaluation of this was presented in [2], referred to as the log-polar Radon transform which is based on rewriting the key operations as convolutions in a log-polar coordinate system. In Section 2.1 we briefly recall the construction of the log-polar Radon transform and discuss how to adjust this method for optimal performance when processing seismic data, and in Section 2.2 we introduce coordinate transforms as well as sampling/interpolation requirements for accurate evaluation of \( \mathcal{R}_h f(\tau q) \).

2.1 Log-polar Radon transform

The standard Radon transform (cf. (3)) can be written in terms of a double integral

\[
\mathcal{R} \tilde{f}(\tau^2, q^2) = \int \int \tilde{f}(s, y) \delta(s - \tau^2 - q^2 y) \, dy \, ds,
\]

where \( \delta \) denotes the Dirac distribution. In [2] one works with the log-polar coordinates

\[
\begin{align*}
\begin{cases}
    s = e^\rho \cos(\theta'), \\
y = e^\rho \sin(\theta'),
\end{cases} \\
\begin{cases}
    \tau^2 = \frac{e^\rho}{\cos(\theta)}, \\
    q^2 = -\tan \theta.
\end{cases}
\end{align*}
\]

By introducing \( \zeta(\theta, \rho) = \delta(\cos(\theta) - e^\rho) \), it turns out that the Radon transform can be efficiently evaluated using the log-polar Radon transform

\[
\mathcal{R}_{lp} \tilde{f}(\theta, \rho) = \cos(\theta) \int \int \tilde{f}(\theta', \rho') \, e^{\rho'} \, \zeta(\theta - \theta', \rho - \rho') \, d\rho' \, d\theta',
\]

117
Figure 1: CMP gather, cutting information for high offset and small time intercept (left). Hyperbolic Radon transform, the circles are artifacts from truncation (right).

where, by abuse of notation, we use the same notation $\tilde{f}$ for both coordinate representations.

However, the above representation is not suitable for treating functions $\tilde{f}$ with support near 0, since this corresponds to $\rho' = -\infty$. One therefore applies scaling, rotation and translation to work with functions supported within a subset of a circle-sector of opening angle $\beta$ as in Figure 2, right. Due to certain technicalities [2], the implementation of $\mathcal{R}_{lp}$ works best when evaluating only for values $\theta \in [-\beta/2, \beta/2]$. We will refer to this algorithm as the partial $\mathcal{R}_{lp}$.

With this in mind, we now briefly explain how to make slight modifications to the above scheme, better suited for the processing of CMP gathers. A simplified synthetic example of a typical CMP gather is shown in Figure 1. Note that the function continues outside the maximum limits given by $x$ and $t$, leading to a truncation of (3), (which can be seen e.g. as the circular artifacts in Figure 1). Also note that there is no data in the region above a line $t = kx$, i.e. high offset $x$ and small time intercept $t$, so to decrease the amount of computations we may ignore this piece. In the coordinates $(s, y)$ this triangle is again a triangle, but with equation $s = k^2 y$. We set $\gamma = \arctan k^2$. Thus, we are in practice only interested in evaluating (4) for data $\tilde{f}$ on a right trapezoid
with the form illustrated in Figure 2, left. Besides, one is usually also interested only in values of \((\tau, q)\) in a limited range \([\tau_{\text{min}}, \tau_{\text{max}}] \times [q_{\text{min}}, q_{\text{max}}]\).

In order for this to correspond to a symmetric interval of \(\theta\), we set \(\beta = \arctan(q_{\text{max}}^2) - \arctan(q_{\text{min}}^2)\) and modify the relation between \(\theta\) and \(q\) in (5) as follows

\[
\theta = \alpha - \arctan(q^2),
\]

where \(\alpha = (\arctan(q_{\text{max}}^2) + \arctan(q_{\text{min}}^2))/2\). For a particular value of \(\theta\) the output of the partial \(\mathcal{R}_{\text{lp}}\) correspond to integrals over lines whose angle with respect to the vertical axis is \(\theta\). In order for these to correspond to desired values of \(q\), one needs to rotate \(\tilde{f}\) so that the \(t\) axis makes an angle \(\alpha\) with respect to the horizontal axis in Figure 2, right. Moreover, due to the problems at the origin, \(\tilde{f}\) needs to be dilated and translated so that it fits within the circle sector of radius 1 and opening angle \(\beta\), as in Figure 2, right. This has the effect that the trapezoidal support is inscribed inside a square with side length \(a\), located so that three of its corners lie on the border of the sector. It
can be shown that
\[ a = \frac{\sin(\beta)}{\sqrt{\sin(2\alpha)\sin(\beta) + \cos(\beta)(2\sin(2\alpha) + \sin(\beta)) + 1}}, \]
\[ O = O \left( \frac{a \sin(\alpha + \frac{\pi}{4})\tan(\beta)}{\sqrt{2}}, \frac{a \cos(\alpha + \frac{\pi}{4})\tan\left(\frac{\beta}{2}\right)}{\sqrt{2}} \right). \]

The line \( L_1 \) passes through the fourth corner of the trapezoid and is orthogonal to the border of the sector; the distance from the origin \( O_0 \) to the line \( L_1 \) is indicated by \( a_r \) and indicates the the first non-zero contribution to the partial \( \mathcal{R}_{lp} \).

In summary, we are interested in the values of the log-polar Radon transform in the range \( \left[ -\frac{\beta}{2}, \frac{\beta}{2} \right] \times [\log(a_r), 0] \). With this setup the log-polar Radon transform can be computed in terms of the finite convolution
\[ \mathcal{R}_{lp} \tilde{f}(\theta, \rho) = \cos(\theta) \int_{-\frac{\beta}{2}}^{\frac{\beta}{2}} \int_{\log(a_r)}^{0} \tilde{f}(\theta', \rho')e^{\rho'} \zeta(\theta - \theta', \rho - \rho') d\rho' d\theta' = \]
\[ \cos(\theta) \mathcal{F}^{-1} \left( \mathcal{F} \left( \tilde{f}(\theta, \rho)e^{\rho} \right) \hat{\mathcal{F}}(\hat{\theta}, \hat{\rho}) \cdot \mathcal{F} \zeta(\hat{\theta}, \hat{\rho}) \right)(\theta, \rho). \]
Here, \( \mathcal{F} \) denotes the two-dimensional Fourier transform. We use the notation \((\hat{\theta}, \hat{\rho})\) for the reciprocal variables of \((\theta, \rho)\). The function \( \hat{\zeta}(\hat{\theta}, \hat{\rho}) \) can be accurately evaluated numerically (in a precomputing step) in contrast to \( \zeta(\theta, \rho) \) which is defined in terms of distributions and is discontinuous along a curve, see [2] for a detailed description. To avoid wrapping effects, zero-padding is applied in the log-polar domain. The effects of the convolutions are schematically illustrated in Figure 3. The trapezoid containing the support of the data is transformed to the shape indicated by the black points after a change to log-polar coordinates; the green lines show shifted versions of the function \( \zeta \); and the support after the log-polar Radon transform is applied is indicated by the thick black curves. By using this scheme we conclude that the rectangle \([-\beta, \beta] \times [\log(a_r), 0] \) is a good choice for enclosing the support of the functions, which is needed for the discrete evaluation of the integrals by means of convolutions in log-polar coordinates.

We now describe how \( \mathcal{R}_{lp} \) can be used to recover \( \mathcal{R}\hat{f} \) for a function \( \hat{f} \) with support in the unit rectangle. The change of coordinates \((s, y)\) for the log-polar setup is described by the transformation \( T \),

\[
T \begin{pmatrix} t \\ x \end{pmatrix} = a \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} s - 0.5 \\ y - 0.5 \end{pmatrix} + \begin{pmatrix} O_1 \\ O_2 \end{pmatrix} \tag{7}
\]

as well as the change of coordinates \((\tau, q)\) for the log-polar setup can be expressed by \( S \), which can be found by scaling, rotation and translation procedures for the log-polar setup. Some tedious manipulations yield

\[
S \begin{pmatrix} \tau^2 \\ q^2 \end{pmatrix} = \left( a(\tau^2 - \frac{1}{2}) \cos(\alpha) + a \frac{\sin(\alpha)}{2} + O_1 + \left( a(\tau^2 - \frac{1}{2}) \sin(\alpha) - a \frac{\cos(\alpha)}{2} + O_2 \right) \phi \right) \phi
\]

with \( \phi = \tan(\alpha - \arctan(q^2)) \).

Moreover, we introduce two transformations for switching to log-polar coordinates according to relations (5):

\[
P_1 \begin{pmatrix} t \\ x \end{pmatrix} = \left( \log(\sqrt{t^2 + x^2}) \right) \arctan \left( \frac{x}{t} \right), \quad P_2 \begin{pmatrix} \tau^2 \\ q^2 \end{pmatrix} = \left( \log(\tau^2 \cos(-\arctan(q^2))) \right) - \arctan(q^2) \tag{8}
\]
To the end, by introducing linear operators

\[ T \tilde{f} = \tilde{f}(T^{-1}P_1^{-1}) \quad S g = g(S^{-1}P_2^{-1}) \]

the Radon transform over straight lines and its adjoint operator can be computed (up to a scaling factor) by

\[ R \tilde{f}(\tau, q) = S^{-1} R_{lp}(T \tilde{f})(\tau, q), \]
\[ R^* g(t, x) = T^{-1} R^*_{lp}(S g)(t, x). \]

### 2.2 Hyperbolic coordinates

Let \( f \) be a CMP gather measured on the rectangle

\[ \{ (t, x) : 0 \leq t \leq T, 0 \leq x \leq X \}. \]

which we treat as a function on all of \( \mathbb{R}^2 \) which is 0 outside this rectangle. Note that

\[ R_h(f)(\tau, q) = X T \int_0^1 \int_0^1 f(T t, X x) \delta \left( t T - \sqrt{\tau^2 + q^2 x^2 X^2} \right) d x d t = \]
\[ X \int_0^1 \int_0^1 f(T t, X x) \delta \left( t - \sqrt{(\tau/T)^2 + q^2 x^2 X^2/T^2} \right) d x d t = \]
\[ X R_h \left( f(T \cdot, X \cdot) \right) \left( \frac{\tau}{T}, \frac{q X}{T} \right), \]

which allows us to assume that \( f \) is given on the rectangle \([0, 1] \times [0, 1]\) to begin with. Upon corresponding rescaling of \( \tau := \frac{\tau}{T} \) and \( q := \frac{q X}{T} \), we are interested in evaluating \( R_h f \) on the rectangle

\[ \{ (\tau, q) : \tau_{min} \leq \tau \leq 1, q_{min} \leq q \leq q_{max} \}, \]

where \( \tau_{min} \) corresponds to the arrival of the first event in the rescaled coordinates.

Now we recall the expression (6) representing the finite convolution for computing the log-polar Radon transform. It can be rapidly evaluated in
Fast hyperbolic Radon transform represented as convolutions in log-polar coordinates

terms of FFT if the log-polar samples \((\theta, \rho)\) are given on an equally spaced grid. Since data is assumed to be sampled in the \((t, x)\) domain, a resampling is needed. We propose to do this using cardinal B-spline interpolation [9, 25], since this type of interpolation is particularly well suited for GPU implementations (cf. [20]). This technique is related to that used for fast unequally-spaced Fourier transforms (USFFT) [5, 10], in the way that the interpolation is conducted by smearing data in one of the domains, and the compensating for that effect is done in the reciprocal domain.

In (6) we have to compute \(\mathcal{F}\left(\tilde{f}(\theta, \rho)e^{\rho}\right)\left(\hat{\theta}, \hat{\rho}\right)\) which we can write as

\[
\mathcal{F}\left(\tilde{f}(\theta, \rho)e^{\rho}\right)\left(\hat{\theta}, \hat{\rho}\right) = \frac{\mathcal{F}\left(\left(\tilde{f}e^{\rho}\right) * B_3\right)\left(\hat{\theta}, \hat{\rho}\right)}{\mathcal{F}B_3\left(\hat{\theta}, \hat{\rho}\right)},
\]

where \(B_3\) is the cubic (cardinal) B-spline. Here we only consider frequencies \((\hat{\theta}, \hat{\rho})\) in a rectangle \(L\), where \(|\mathcal{F}B_3(\hat{\theta}, \hat{\rho})|\) does not become too small.

By using the coordinate transformations (2), (7) and (8) let

\[
\begin{pmatrix}
\varphi(t, x) \\
\eta(t, x)
\end{pmatrix} = P_1 T \begin{pmatrix} t^2 \\ x^2 \end{pmatrix}
\]

be log-polar coordinates that correspond to the coordinates \((t^2, x^2)\) in the time-offset domain. In these coordinates (10) takes the form

\[
\mathcal{F}\left(f(\theta, \rho)e^{\rho}\right)\left(\hat{\theta}, \hat{\rho}\right) = \frac{\mathcal{F}\left(\iint \frac{f(t,x)}{2x} e^{\eta} J(t,x)B_3(\theta-\varphi, \rho-\eta) \, dt \, dx\right)\left(\hat{\theta}, \hat{\rho}\right)}{\mathcal{F}B_3\left(\hat{\theta}, \hat{\rho}\right)},
\]

where the division by \(2x\) is related to the transformation (2). However, the Jacobian determinant \(J(t, x) = \left|\frac{\partial(\varphi, \eta)}{\partial(t, x)}\right|\) is a easily seen to consist of smooth bounded functions multiplied with \(2x\) (coming from the derivative of \(x^2\)), which cancels out this seeming singularity at \(x = 0\). Subsequently the integrals and the Fourier transforms above can be well approximated using the trapezoidal rule and FFT for approximative evaluation of \(\mathcal{F}\). If \((t_j, x_k)\) are regular sampling points in the time-offset domain, we introduce the approx-
imation to (11) by
\[ g(\hat{\theta}, \hat{\rho}) = \mathcal{F}\left( \frac{\sum_{j,k} f(t_j, x_k) e^{\eta(t_j, x_k)} J(t_j, x_k) B_3(\theta - \varphi(t_j, x_k), \rho - \eta(t_j, x_k))}}{c} \right)(\hat{\theta}, \hat{\rho}) \]
\[ \mathcal{F} B_3(\hat{\theta}, \hat{\rho}) \]

where \( c \) is a constant related to the sampling intervals. This approximation is then accurate for values of \((\hat{\theta}, \hat{\rho})\) in the rectangle \( L \) mentioned above. As outlined in the previous section, this allows us to efficiently compute approximations of (6) on a regular lattice in the log-polar coordinate system \((\theta, \rho)\) via the formula
\[ \mathcal{R}_{lp} \tilde{f}(\theta, \rho) \approx \cos(\theta) \mathcal{F}^{-1} \left( g(\hat{\theta}, \hat{\rho}) \cdot \mathcal{F} \zeta(\hat{\theta}, \hat{\rho}) \right)(\theta, \rho). \] (12)

The final interpolation from the log-polar \((\theta, \rho)\) lattice to the Radon \((\tau, q)\) lattice can be done by using cubic B-splines and a slight modification of (12). Here, we again employ the transformations (7-8). In this case, let
\[ \begin{pmatrix} \varphi(\tau, q) \\ \eta(\tau, q) \end{pmatrix} = P_2 S \begin{pmatrix} \tau^2 \\ q^2 \end{pmatrix} \]
be the log-polar coordinates that correspond to the coordinates \((\tau^2, q^2)\) in the Radon domain. The interpolation from the log-polar \((\theta, \rho)\) lattice to the Radon \((\tau, q)\) lattice can then be done by using (a discrete version of)
\[ \mathcal{R}_h f(\tau, q) = \cos(\varphi) \int_{-\rho}^{\rho} \int_{\log(a_x)}^{0} \left( \mathcal{F}^{-1} \left( \frac{\chi_L g(\hat{\theta}, \hat{\rho}) \mathcal{F} \zeta(\hat{\theta}, \hat{\rho})}{\mathcal{F} B_3(\hat{\theta}, \hat{\rho})} \right)(\theta, \rho) \right) B_3(\varphi - \theta, \rho - \eta - \rho) d\rho d\theta, \] (13)
where \( \chi_L \) denotes the characteristic function of the set \( L \).

Numerical evaluation of the approximations (11) and (13) appear to be well-suited for parallel computations, particularly on GPUs. For FFT we make use of the high-performance cuFFT library, efficient GPU kernels can be constructed for the smearing operations and for the vector multiplications. The discrete version of the operator \( \mathcal{R}_h \), as explained in the previous sections, will be denoted by \( R_h \).
3 Reconstruction techniques

The adjoint operator for the hyperbolic Radon transform $R_h^*$ is defined by using the inner product equality
\[ \langle R_h f, g \rangle = \langle f, R_h^* g \rangle, \tag{14} \]
for arbitrary $f$ and $g$. The operator is easy to construct by using the approach with switching to log-polar coordinates, essentially by reversing the order of the operations. With the adjoint operations at hand, one can consider iterative methods for representing $f$ by sparse sums of hyperbolic wave events, and related interpolation and reconstruction techniques. A popular such method is based on the soft thresholding method for obtaining sparse representations proposed in [8]. In this setting it means to consider the minimization of
\[ \| R_h^* g - f \|_2^2 + \mu \| g \|_1, \tag{15} \]
for some choice of sparsity parameter $\mu$.

By a simple modification of Theorem 3.1 in [8], this minimization problem is solved by the iterations
\[ g^n = S_{c^2\mu}(g^{n-1} + c^2 R_h(f - R_h^* g^{n-1})), \quad n = 1, 2, \ldots, \tag{16} \]
where $g^0$ is arbitrary, $c$ is a positive constant such that $c \| R_h \| < 1$, and $S_\mu$ is a soft-thresholding function defined as
\[ S_\mu(v) = \begin{cases} v + \frac{\mu}{2}, & \text{if } v \leq -\frac{\mu}{2}, \\ 0, & \text{if } |v| < \frac{\mu}{2}, \\ v - \frac{\mu}{2}, & \text{if } v \geq \frac{\mu}{2}. \end{cases} \]

To perform interpolation in the case of missing data, let $S$ be a subset of the $(t_j, x_k)$ grid where we do have measurements of $f$. We are then interested in minimizing
\[ \sum_{(t_j, x_k) \in S} (R_h^* g - f)^2(t_j, x_k) + \mu \| g \|_1, \]
which, defining $f$ to be 0 where data is missing, is solved by the iteration
\[ g^n = S_{c^2\mu}(g^{n-1} + c^2 R_h(f - \chi_S R_h^* g^{n-1})), \quad n = 1, 2, \ldots. \]
Here \( \chi_S \) is the characteristic function of \( S \). Again, this scheme is efficiently evaluated using the fast implementation of \( R_h \) explained in the previous section.

## 4 Discretization

In this section we derive guidelines for how to choose discretization parameters. For simplicity, we assume to work with regular sampling in the time-offset, and in the Radon domain; but the log-polar-based method can be easily generalized for unequally-spaced grids in these two domains.

In order to apply FFTs, samples in log-polar coordinates \((\theta, \rho) \in [-\beta, \beta] \times [\log a, 0]\) must be chosen on an equally spaced grid. By using coordinate conversions for the log-polar setup given by

\[
\begin{pmatrix}
\varphi(t, x) \\
\eta(t, x)
\end{pmatrix} = P_1 T \begin{pmatrix} t^2 \\ x^2 \end{pmatrix}.
\]

In order to maintain accurate interpolation, we choose the sample spacing in \( \theta \) and \( \rho \) with respect to the largest distance between sample points in the \( \varphi \) and \( \eta \) variables, i.e.,

\[
\begin{align*}
\Delta \theta &\geq \max_{t_j, x_k} \left( \left| \varphi(t_j, x_k) - \varphi(t_j + \Delta t, x_k) \right|, \left| \varphi(t_j, x_k) - \varphi(t_j, x_k + \Delta x) \right| \right), \\
\Delta \rho &\geq \max_{t_j, x_k} \left( \left| \eta(t_j, x_k) - \eta(t_j + \Delta t, x_k) \right|, \left| \eta(t_j, x_k) - \eta(t_j, x_k + \Delta x) \right| \right).
\end{align*}
\]

This choice will determine the log-polar frequency range that can be covered, which in turn determines the resolution in the \((\tau, q)\) (Radon) domain. The quadratic behavior in the time sampling can be fairly well described in terms of the log-polar sampling, as long as time range is not too large. In the case of large time ranges, it can be beneficial to split the time-offset and Radon domains in parts and consider the log-polar Radon transform for each of these parts, in order to avoid too large differences in sample densities.

For instance, for small values of \( \tau \) the grid for the Radon domain becomes more dense and samples \( \Delta \theta, \Delta \rho \) should be chosen to be smaller. Suppose that we have already rescaled \( f \) according to (9), and note that the function is...
Fast hyperbolic Radon transform represented as convolutions in log-polar coordinates

0 until the arrival of the first event at $\tau_{\text{min}}$. We may then split the integral in the following way

$$\mathcal{R}_h f(\tau, q) = \int_{\tau_{\text{min}}}^{1} \int_{0}^{1} f(t, x) \delta(t - \sqrt{\tau^2 + q^2 x^2}) dx dt =$$

$$\int_{\tau_{\text{min}}}^{a} \int_{0}^{1} f(t, x) \delta(t - \sqrt{\tau^2 + q^2 x^2}) dx dt +$$

$$\int_{a}^{1} \int_{0}^{1} f(t, x) \delta(t - \sqrt{\tau^2 + q^2 x^2}) dx dt$$

(19)

for some $a$ between $\tau_{\text{min}}$ and 1. For numerical evaluation of the first integral by using the log-polar based method samples in $\theta, \rho$ determined according to (18) become more dense, see Figure 4 for a schematic description. The red dots in Figure 4b indicate log-polar samples after conversion to discrete coordinates in the $(t, x)$ domain illustrated in Figure 4a. Equally spaced samples in the log-polar domain (gray dots) are chosen with respect to maximal distances between points (18). Figure 4c demonstrates samples in the log-polar domain corresponding to small values of $t$ (located above the gray line in Figure 4a). The splitting procedure is not computationally intensive and can be applied several times to achieve accuracy for small values of $\tau$.

5 Accuracy and performance tests

For the sake of quality comparisons, we perform the same tests as the ones presented by [15] for the fast hyperbolic Radon transform based on fast butterfly algorithms. The method of [15] is available in the open source software package Madagascar [12]. The synthetic CMP gather (Figure 5a) was used as a reference for making comparisons. As a reference method, we use a standard C implementation of the direct summation given by (3). Here cubic interpolation is used for the interpolation in time.

The fast butterfly algorithm has several parameters for controlling efficiency and accuracy, for details we refer to the pages 5, 6 in [15]. The parameter $M$ ($N$ in the paper) is of the order of the maximum value of the phase function $|\Phi(x, k)|$ used for the approximation; and parameters $q_{k_1}, q_{k_2}, q_{x_1}, q_{x_2}$ control the number of Chebyshev points. According to the results presented
in [15], the set of parameters \( (q_i = 9, M = 64) \) shows an accuracy level of about \( O(10^{-3}) \) for images of size 1000 \( \times \) 1000. We performed tests for \( N_t = N_x = N_c = N_q = N \) where \( N \) was chosen as different powers of 2, and for obtaining an approximate accuracy level of \( O(10^{-3}) \) we used \( (q_i = 9, M = N/16) \) in accordance with the tests conducted in [15].

Normalized errors compared to direct summation over hyperbolas for
the log-polar-based and for the fast butterfly algorithm are demonstrated in Figure 5c and Figure 5d, respectively. The figures show that the two methods have the same order of errors. The errors for the log-polar-based method are mostly observed in the region of small time intercept ($\tau$) and high values of slowness ($q$). These accuracy problems can be reduced by additional splittings of the integral for the hyperbolic Radon transform, similar to the one suggested in the expression (19). To be concrete, the presented results for the log-polar-based method were obtained after one splitting in the time variable, and one splitting in the slowness variable.
Figure 5: Hyperbolic Radon transform. Corresponding normalized errors compared to direct summation over hyperbolas.

Table 1 demonstrates the computational times for the fast butterfly algorithm; for the CPU and the GPU versions of the log-polar-based method, respectively; and for the direct summation over hyperbolas. The table confirms the complexity of the proposed method and shows that a substantial performance gain is obtained by using GPUs. It is common in GPU computing that time to copy data between host and device memory constitute an essential part of the total computational costs (for our tests it takes ≈ 30% of...
Fast hyperbolic Radon transform represented as convolutions in log-polar coordinates. This time can be neglected in the case of using iterative schemes since it is then possible to keep all data in the GPU memory. For the tests performed, we used a standard desktop with an Intel Core i7-3820 processor and NVIDIA GeForce GTX 970 video card with PCI Express x16 graphic interface. All computations were performed in single precision. We note that parallel versions of butterfly algorithms have been described in [19], but we use the single core implementation described in [15] to make sure that the computational times are in accordance with the results reported in [15].

In Figure 6 we show the output of the adjoint operator for the hyperbolic Radon transform, as well as $\ell^1$ regularized reconstruction given by (15). The proposed algorithm passes the inner product test (14) with a relative error $O(10^{-5})$. The iterative reconstruction demonstrates good quality (compare figures 5a and 6b).

Table 2 shows computational times using a GPU implementation of the proposed log-polar-based hyperbolic Radon transform, and 64 iterations of the iterative scheme (16). The table also contains times for single iteration of the forward and adjoint operators. One can see that in comparison to the GPU results in Table 1, the times for the forward operator are lower due to limited number of host-device data transfers. For this scheme data was copied only for an initial guess $g^0$; the measured data $f$; and the final result.

![Figure 6: Output of the adjoint operator for the hyperbolic RT (left), and the result from using 30 soft-thresholding iterations from (16) (right).](image-url)
Table 1: Computational time (in sec) for the hyperbolic Radon transform via direct summation over hyperbolas, fast butterfly algorithm and via the log-polar-based method (CPU and GPU), speed-up compared to the direct summation.

<table>
<thead>
<tr>
<th>N</th>
<th>Direct sums CPU, 1 core</th>
<th>Fast butterfly CPU, 1 core</th>
<th>Log-polar CPU, 8 cores</th>
<th>Log-polar GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>time</td>
<td>time</td>
<td>time</td>
</tr>
<tr>
<td></td>
<td>speed up</td>
<td>speed up</td>
<td>speed up</td>
<td>speed up</td>
</tr>
<tr>
<td>$2^9$</td>
<td>4.8e+00</td>
<td>1.1e+00</td>
<td>4.3</td>
<td>3.3e-02</td>
</tr>
<tr>
<td>$2^{10}$</td>
<td>4.0e+01</td>
<td>4.5e+00</td>
<td>9.0</td>
<td>1.2e-01</td>
</tr>
<tr>
<td>$2^{11}$</td>
<td>3.2e+02</td>
<td>1.8e+01</td>
<td>17.8</td>
<td>4.7e-01</td>
</tr>
<tr>
<td>$2^{12}$</td>
<td>2.5e+03</td>
<td>7.3e+01</td>
<td>33.7</td>
<td>2.0e+00</td>
</tr>
</tbody>
</table>

Table 2: Computational time (in sec) for 64 soft-thresholding iterations.

<table>
<thead>
<tr>
<th>N</th>
<th>Total time</th>
<th>Average time per iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Forward operator</td>
</tr>
<tr>
<td>$2^9$</td>
<td>8.1e+00</td>
<td>1.9e-03</td>
</tr>
<tr>
<td>$2^{10}$</td>
<td>3.1e+01</td>
<td>7.4e-03</td>
</tr>
<tr>
<td>$2^{11}$</td>
<td>1.2e+02</td>
<td>2.9e-02</td>
</tr>
<tr>
<td>$2^{12}$</td>
<td>4.6e+02</td>
<td>1.1e-01</td>
</tr>
</tbody>
</table>
6 Applications

In this section we mention some applications of the fast hyperbolic Radon transform. These are fairly standard, but the examples could be of practical interest due to the substantial computational speedup of the proposed implementation of the hyperbolic Radon transform.

6.1 Multiple attenuation.

A well-known method for the attenuation of multiple reflections in CMP gathers is based on conducting the attenuation in a Radon domain. Here, multiples and primaries can be separated due to their differences in moveout. We have tested the method described in Chapter 3 for the synthetic CMP gather in Figure 7a. Figure 7c illustrates the Radon data, and note that the primaries and multiples are difficult to separate. The corresponding result after using 30 soft-thresholding iterations is illustrated in Figure 7d. The black line indicates the border between primaries and multiples, and Figure 7b shows the reconstructed primaries.

6.2 Interpolation.

Here we show some examples where we use soft-thresholding for conducting interpolation in cases of missing traces in the sampling setups. The CMP gather in Figure 8a contains 50% randomly missing traces. For the data reconstruction we use the simple modification (17) of the iterative scheme for obtaining sparse representations. Figure 8b shows reconstruction results after 30 soft-thresholding iterations. Note the absence of high amplitude artifacts produced by the proposed method. To control the obtaining results, we also consider synthetic CMP gathers with 90% of missing traces, see figures 9a,c. In spite of the low amount of given data, it is still possible to reconstruct the structure of the waves (Figure 9b). Moreover, varying the parameter of soft-thresholding ($\mu$, see Chapter 3), one can improve the reconstruction quality. Here, the increase of the parameter $\mu$ leads to a better accuracy for low-amplitude events; conversely, high-amplitude events can be reconstructed with smaller values of $\mu$. In Figure 9d we show the result of the reconstruction with soft-thresholding iterations where the parameter $\mu$ was increased by 10 times compared to the one used for reconstructions in Figures
8b and Figure 9b.

6.3 2D field data.

As an example of real data processing, we consider a CMP gather from the Canterbury data set [17]. Multiple reflections in this CMP gather start at around 2.2 s (Figure 10a). Attenuation of the multiples was carried out after applying the reconstruction method from Chapter 3 with 40 soft-thresholding iterations and the related muting procedure (Figure 10d). The part of the Radon image corresponding to multiples was taken back to the time-offset domain (Figure 10c) and subtracted from the initial CMP gather (Figure 10b).
Fast hyperbolic Radon transform represented as convolutions in log-polar coordinates

(a) CMP gather

(b) Reconstructed primaries

(c) Hyperbolic Radon transform

(d) Sparse representation

Figure 7: Multiple attenuation with 30 soft-thresholding iterations.
Figure 8: Interpolation into missing traces with 30 iterations of the soft-thresholding algorithm.
Fast hyperbolic Radon transform represented as convolutions in log-polar coordinates

Figure 9: Interpolation into missing traces with the soft-thresholding algorithm.
7 Conclusions

A fast log-polar-based method for the evaluation of the hyperbolic Radon transform has been presented. According to the tests performed, the method demonstrates reasonable accuracy and favorable computational costs com-
Fast hyperbolic Radon transform represented as convolutions in log-polar coordinates compared to other methods. The accuracy of the method can be increased when considering higher order interpolation kernels for coordinate conversions between time-offset, Radon, and log-polar domains. Numerical tests show that the GPU implementation is more than 10000 faster for large data sets in comparison to a direct implementation in standard C of sums over hyperbolas, and a substantial speedup is also obtained compared to alternative fast methods.

Acknowledgments

This work was supported by the Crafoord Foundation (2014-0633) and the Swedish Research Council (2011-5589, 2015-03780)

References


Fast hyperbolic Radon transform represented as convolutions in log-polar coordinates


Paper IV
Directional interpolation of multicomponent data

Fredrik Andersson Adriana Citlali Ramírez Pérez
Torgeir Wiik Viktor V. Nikitin

Abstract

A method for interpolation of multicomponent streamer data based on using the local directionality structure is presented. The derivative components are used to estimate a vector field that describes locally the direction with the least variability. Given this vector field, the interpolation can be phrased in terms of the solution of a partial differential equation that describes how energy is transported between different parts of the data. The approach can be efficiently implemented using readily available routines for computer graphics. The method is robust to noise in the measurements, and in particular it is robust towards the high levels of low-frequent noise that is present in the derivative components of the multicomponent streamer data.

1 Introduction

When acquiring 3D marine seismic data, we often use survey designs that are aimed at achieving a cost-effective sampling of the subsurface. Seismic spatial sampling refers to the distribution of shotpoints (source firing locations) and recording positions (receiver’s locations for a given fired source). In a 3D experiment, ideally, that distribution should sample both sides (source and receiver side) with enough density over a 2D carpet covering the area under study [10, 23]. The goal is to measure seismic data that are properly sampled in all spatial dimensions: inline (x), crossline (y), offset, and azimuth. Properly sampled, in this case, refers to a fit-for-purpose survey design and corresponding data acquisition. The completeness of such sampling is related to the
highest practical frequency of interest given the processing strategy/objectives and the geology under study [18].

Based on the design of most used processing algorithms, it is common to consider uniform sampling, based on the Nyquist sampling theory, as an ideal sampling pattern. However, lower sampling rates can be obtained by introducing additional constraints, for instance that the reconstructed wave field should have a sparse representation. The best sampling pattern depends on how the measured data would be used. Some advanced processing algorithms have strong requirements on data completeness, and the lack of such completeness forces in approximations that often result in shortcomings in the quality of the results. For example, data-driven free-surface and internal multiple prediction require (in principle) ideally sampled data with coincident sources and receivers [5, 17, 25].

The practical aspects of marine acquisition limit our ability to acquire such ideally sampled datasets. Towed streamer data, for example, are better sampled on the receiver side than on the source side; furthermore, the receiver side is well sampled in the inline direction (commonly ranging from 1.125m to 12.5m sample rate) while the crossline direction is much sparser (often sampled every 100m). Ocean bottom acquisitions usually have a relatively homogeneously sampled 2D carpet of shotpoints (every ≈50m both inline and crossline), while the receivers can be sparse in both directions (e.g., nodes every 200-500m). The sampling sparsity (in one or more dimensions) often leads to unsatisfied assumptions or approximations to the theory behind specific algorithms, and limits the maximum frequency used in processing the measured data. Highly sparse sampling can also lead to the use of 1D or 2D processing algorithms instead of the more accurate 3D ones. Thus reducing the quality of the end product (e.g., final image or inversion) and/or increasing the need for approximations, adaptive filters, and signal processing manipulation.

Although sampling completeness is a common challenge for the processing of several types of data, we focus here on towed-streamer seismic data sampling. It has been common practice to measure only one wave-field component (pressure) in towed streamer surveys. Therefore, most of the interpolation methods have been targeted at interpolating pressure measurements [1, 11, 19]. However, new acquisition technologies that measure more than one wave-field component at the streamers are now available [12, 15, 20]. These tech-
nologies measure either dual data components (analogous to pressure field and its normal particle velocity) or three components (analogous to pressure field plus its normal and crossline particle velocity components). The particle velocity components are associated, in a static way, to the wave-field derivatives with respect to spatial coordinates \([8]\). The depth derivative is usually measured to be used in up/down separation algorithms \([2]\), also known as receiver-side deghosting. It is well known that the depth derivative is complementary to the scalar wavefield, pressure field, allowing to separate the measured wavefield into its up-going and down-going (ghosts) parts at the receiver location, where it was measured, see for example \([27]\) and references therein. The measurement of the crossline component on the other hand has been motivated by the challenge of interpolating data in between sparse measurements in the crossline direction. Popular methods recently developed to use this extra measurement in the interpolation of pressure data are based on Fourier reconstruction and the theory of matching pursuit, as documented in e.g. \([22]\). Fourier reconstruction/interpolation methods share the assumption that the spectrum of a given signal is sparse, however, they are challenged by spatial aliasing \([13]\). Different alternatives have been proposed to reduce the adverse effects of spatial aliasing within Fourier reconstruction methods. State of the art algorithms use information extracted from unaliased frequencies to estimate attributes that can guide the interpolation at higher (aliased) frequencies. The information contained in spatial derivative(s) of a given wavefield that is to be reconstructed or interpolated partially removes the spatial aliasing challenge, by theoretically allowing Fourier methods to work with data up to and below the second order of aliasing \([14]\).

In this paper, we focus on the problem of interpolating towed streamer data where multicomponent measurements exist, and where the crossline direction is sparsely sampled. Following the ideas presented in \([16]\), we propose to use an assumption of few crossing events as a regularizing constraint for the interpolation, which is formulated using structure tensors \([9]\). Under this condition, we can obtain slowly varying directions along which the pressure field should be constant. The last fact is then used to derive a partial differential equation for performing the interpolation. By this approach we keep the values at the known locations intact. The structure tensors average information locally, and hence the obtained estimates of local directionality will be quite robust towards measurement noise. The derivative parts of the
multicomponent streamer data will have rather high noise levels in the lower frequencies. We will present strategies on how to limit the impact of this kind of noise in the interpolation method. Moreover, we will show that the same directionality structure can be used for the interpolation of both the pressure components, and the depth derivative component. This implies that there is no need for measuring the crossline derivative of the depth derivative, which seems to be needed at a first glance.

2 Method and Theory

2.1 Model

As a model for a given seismic dataset, we assume that it locally consists of a sum of plane waves. By locally plane wave we mean that the change of direction (curvature) of the wave event is small in comparison to the effective signature of the wave. Under this definition, the wave event can be a local wave packet containing more than one event, locally having the same directionality. For example, the wave packet could be a primary and its ghosts. The model will treat the packet as a single local event with a long effective signature. This means that while the wave event itself can be rather oscillatory, the direction of the wave is varying slowly. This is a feature that will play a key role in this work, because for slowly varying functions interpolation can be expected to work well in contrast to the case of the interpolation of rapidly oscillatory functions.

Large parts of seismic data seem to locally have just one individual wave event. At other portions of the data there are crossing events. A key observation for this work is that for the case of a function being described by two intersecting plane waves in three dimensions, there is a line along which the function is constant. This means that undersampled data can be perfectly recovered, by interpolating along that direction. For more complicated scenarios, we seek to find a method that detects the direction where the data varies the least, and conduct the interpolation in this direction. This seems to work quite well in practice, as will be illustrated later in this paper.

This approach can be extended so that perfect recovery can be obtained for the case where the data locally needs more than two plane waves to be well approximated, but at the cost of more parameters to estimate, and that the
Directional interpolation of multicomponent data

computational cost for conducting the interpolation increases substantially. A description on how to treat the case of three local plane waves is provided in appendix A.1.

Let us begin with the following observation: For any plane wave \( u = u(t, x, y) \) with a normal not being perpendicular to the \( y \)-direction, there are constants \( q_t \) and \( q_x \) so that

\[
u_y + q_t u_t + q_x u_x = 0,
\]

(1)
because by definition a plane wave can be written as \( u(t, x, y) = f(t \tau_0 + x \xi_0 + y \eta_0) \) for some one-dimensional function \( f \) and constants \( \tau_0, \xi_0, \eta_0 \). Choosing \( q_t = \tau_0/\eta_0 \) and \( q_x = \xi_0/\eta_0 \) yields the relation above.

Note that for a single plane wave the constants \( q_t \) and \( q_x \) in (1) are not uniquely determined, but any choice of \( q_t \) and \( q_x \) such that \((q_t, q_x, 1) \parallel \nabla u \) will suffice. However, in the case where \( u = u_1 + u_2 \), where \( u_1 \) and \( u_2 \) are plane waves, \( q_t \) and \( q_x \) will be uniquely determined by the relation

\[
(q_t, q_x, 1) \parallel \nabla u_1 \times \nabla u_2.
\]

(2)

As pure plane waves in many cases are not sufficient for modeling, we want to find a way to describe perturbations of plane waves, i.e., waves that locally have one dominant oscillatory direction and small curvature. A generalization of (1) would then read

\[
u_y + q_t(t, x, y) u_t + q_x(t, x, y) u_x = 0,
\]

(3)

where \( q_t \) and \( q_x \) are now smoothly varying functions. Suppose now that there is information about \( u \) (and its derivatives) in some region, but that this information is not available elsewhere. Under the assumption that \( u \) is described well as perturbations of at most two plane waves locally, one approach for doing interpolation (or extrapolation) of \( u \) would be to try to estimate the functions \( q_t \) and \( q_x \) above; and under the assumption that \( q_t \) and \( q_x \) are slowly varying functions, conduct an interpolation of \( q_t \) and \( q_x \) outside the known region, and then conduct the actual interpolation in \( u \) by solving (3). Since the functions \( q_t \) and \( q_x \) in this way implicitly describe a directional interpolation, we will refer to the vector field \((q_t(t, x, y), q_x(t, x, y))\) as a Directional Interpolation Vector Field (DIVF). Let us now turn our focus to how to make DIVF estimations given data in some region. From (2) we
get a hint about how this can be done. However, the relation (2) will not be useful in practice for several reasons. To begin with, it is a point-wise estimate without any smoothness condition built in. Secondly, in the case of a single plane wave, the relation is not uniquely defined, and finally, in the case where \( u \) locally consists of two plane waves \( u_1 \) and \( u_2 \), it would be necessary to first separate \( u \) into its components in order to apply (2). Instead, we would like to use a property similar to (2), but one where directionality is a locally defined, smoothly varying property.

2.2 Estimation of the directional interpolation vector fields

**Structure tensors** capture directionality information of waves in the presence of few crossing events. They have appeared in the seismic literature for instance in \([21,7,4]\); for an overview of other areas where they are used see \([26]\). The structure tensor of a function \( u \) is defined as \( T_\alpha(u)(x) = \sum_{\alpha} \mathbf{g}_\alpha \ast (\nabla u)(\nabla u)^T(x) \), where \( \mathbf{g}_\alpha \) denotes a Gaussian with half-widths specified by the parameter \( \alpha = (\alpha_1, \ldots, \alpha_d) \). Specifically, in the real valued, three-dimensional case \( u = u(t,x,y) \), it reads

\[
T_\alpha(u)(t,x,y) = \begin{pmatrix}
\mathbf{g}_\alpha \ast u_t^2 & \mathbf{g}_\alpha \ast u_t u_x & \mathbf{g}_\alpha \ast u_t u_y \\
\mathbf{g}_\alpha \ast u_t u_x & \mathbf{g}_\alpha \ast u_x^2 & \mathbf{g}_\alpha \ast u_x u_y \\
\mathbf{g}_\alpha \ast u_t u_y & \mathbf{g}_\alpha \ast u_x u_y & \mathbf{g}_\alpha \ast u_y^2
\end{pmatrix}(t,x,y).
\]

At each point \((t,x,y)\), it evaluates to a \( 3 \times 3 \) positive semidefinite symmetric matrix. An eigenvalue decomposition \( T_\alpha(u)(t,x,y) = \mathbf{U} \text{diag}(\{\sigma_j\}) \mathbf{U}^T \) will reveal information about the underlying waves. For instance, if \( u \) behaves locally as a plane wave at \((t,x,y)\), then \( \sigma_2(t,x,y) \approx \sigma_3(t,x,y) \approx 0 \), and if \( u \) consists locally of two plane waves (with different normal directions) then \( \sigma_3(t,x,y) \approx 0 \), but \( \sigma_1(t,x,y) \) and \( \sigma_2(t,x,y) \) will not be close to zero, cf. \([4]\).

An appealing feature with the elements of the structure tensors \( T_\alpha(u) \) is that they vary slowly, even for rapidly oscillating functions \( u \), due to the convolution with the Gaussian \( \mathbf{g}_\alpha \). This means that there is a good chance that reasonably accurate interpolated values can be obtained for the elements of \( T_\alpha(u) \), even when only partial information about \( u \) is available; e.g., the
Directional interpolation of multicomponent data

Figure 1: Illustration of direct interpolation of functions and the interpolation of the elements of the structure tensor.

case were \( u \) is known at slices \( y = y_1 \) and \( y = y_2 \), and the values of the elements of \( T_\alpha(u)(t, x, y) \) is sought for at \( y_1 < y < y_2 \).

To illustrate this behavior, let us consider a simplified two-dimensional example. The top and bottom panels of Figure 1, each shows three plane waves with different directions. The two figures differ in the way that the plane waves are translated by comparatively small distances. However, the distances are large enough so that one would see in total 6 plane waves when doing a linear interpolation between the two functions. The result from the linear interpolation is shown in the left middle panel.

The panels 2-4 from the left show the elements of the structure tensors, while the rightmost two panels show the corresponding singular values of the structure tensors. The magnitude of the singular values illustrates (in the 2D case) the number of crossing events; \( \sigma_1 \approx \sigma_2 \approx 0 \) in the absence of waves, \( \sigma_1 \not\approx 0, \sigma_2 \approx 0 \) indicate the presence of one (planar) wave; and the case \( \sigma_1 \not\approx 0, \sigma_2 \not\approx 0 \) indicate that there are two or more waves present (or that the noise level is substantially high). In three dimensions, \( \sigma_3 \approx 0 \) indicates that there are locally two or fewer plane waves present in the data.
The top and bottom panels correspond to the two waves in the left top and left bottom panels, whereas the middle panels correspond to the linear interpolation between the structure tensor elements. In contrast to the linear interpolation that is performed directly on the wave functions, the linear interpolation yields expected results on the structure tensor elements. The reason for this is that they vary slowly. The interaction scale can be controlled by modifying the width of the Gaussian $g_\alpha$, i.e. by controlling the parameter $\alpha$.

Let us now consider the problem where we know the structure tensor $T_\alpha(u)$ (for instance by interpolation). Let us assume that the underlying wave fronts have at a maximum two crossing events. Then the smallest eigenvalue of $T_\alpha(u)$ is zero for all $(t, x, y)$. This means in turn that $u$ should be constant along the line that is parallel to the eigenvector corresponding to $\sigma_3$, i.e., that

$$u_t q_t + u_x q_x + u_y = 0, \quad (4)$$

if $(q_t(t, x, y), q_x(t, x, y), 1)^T$ is an eigenvector to $T_\alpha(u)(t, x, y)$ corresponding to the eigenvalue $\sigma_3(t, x, y) = 0$. In the case of perturbed plane waves, $\sigma_3$ will not be zero but small in comparison to $\sigma_1$ and $\sigma_2$. Hence, the structure tensor can be used to estimate the DIVF components $q_t$ and $q_x$. We will now take a slightly different approach in order to stabilize the estimates. By assuming that $q_t$ and $q_x$ are represented by

$$q_t(t, x, y) = \sum_{n=1}^{N} c_t(n) \phi_n(t, x, y), \quad q_x(t, x, y) = \sum_{n=1}^{N} c_x(n) \phi_n(t, x, y), \quad (5)$$

for some suitable set of smoothly varying and localized functions $\phi_n$. We will now try to estimate the coefficients $c_t(n)$ and $c_x(n)$ instead of making pointwise estimations of $q_t(t, x, y)$ and $q_x(t, x, y)$. In this way the smoothness of $q_t$ and $q_x$ are implicitly controlled by the smoothness of $\phi_n$. In addition regularity penalties on the coefficients $c_t(n)$ and $c_x(n)$ can be used to increase the smoothness.
We now study the minimization of
\[
\iint \int (u_y + q_t u_t + q_x u_x)^2 \, dt \, dx \, dy = \iiint ((q_t \quad q_x \quad 1) \cdot \nabla u)^2 \, dt \, dx \, dy = \\
\iint \int (q_t \quad q_x \quad 1) \begin{pmatrix} u_t^2 & u_t u_x & u_t u_y \\ u_t u_x & u_x^2 & u_x u_y \\ u_t u_y & u_x u_y & u_y^2 \end{pmatrix} (q_t \quad q_x \quad 1) \, dt \, dx \, dy.
\]

By using the representations (5), we can write the expression as
\[
\iint \int (u_y + q_t u_t + q_x u_x)^2 \, dt \, dx \, dy = \iiint \sum_{m,n} (c_t(m) \phi_m c_x(m) \phi_m \frac{1}{N}) \left( \begin{array}{ccc}
u_t^2 & u_t u_x & u_t u_y \\
u_t u_x & u_x^2 & u_x u_y \\
u_t u_y & u_x u_y & u_y^2 \end{array} \right) (c_t(n) \phi_n c_x(n) \phi_n \frac{1}{N}) \, dt \, dx \, dy = \\
\sum_{m,n} c_t(m)c_t(n) \iint \int \phi_m \phi_n u_t^2 \, dt \, dx \, dy + \\
c_x(m)c_x(n) \iint \int \phi_m \phi_x(n) u_x^2 \, dt \, dx \, dy + \\
\frac{1}{N^2} \iint \int u_y^2 \, dt \, dx \, dy + \\
(c_t(m)c_x(n) + c_t(n)c_x(m)) \iint \int \phi_m \phi_n u_t u_x \, dt \, dx \, dy + \\
\frac{1}{N} (c_t(m) + c_t(n)) \iint \int \phi_m u_t u_y \, dt \, dx \, dy + \\
\frac{1}{N} (c_x(m) + c_x(n)) \iint \int \phi_m u_x u_y \, dt \, dx \, dy.
\]

By introducing
\[
\begin{align*}
H_{tt}(m,n) &= \iint \int \phi_m \phi_n u_t^2 \, dt \, dx \, dy, \\
H_{xx}(m,n) &= \iint \int \phi_m \phi_n u_x^2 \, dt \, dx \, dy, \\
H_{xt}(m,n) &= \iint \int \phi_m \phi_n u_t u_x \, dt \, dx \, dy, \\
J_t(n) &= \frac{1}{N} \iint \int \phi_n u_t u_y \, dt \, dx \, dy, \\
J_x(n) &= \frac{1}{N} \iint \int \phi_n u_x u_y \, dt \, dx \, dy.
\end{align*}
\]

(6)
this minimization problem can be written as
\[
\min_{c_t, c_x} \left( c_t c_x \left( \begin{array}{cc} H_{tt} & H_{tx} \\ H_{tx}^* & H_{xx} \end{array} \right) \right) + 2 \left( \begin{array}{cc} J_t \\ J_x \end{array} \right) \left( \begin{array}{c} c_t \\ c_x \end{array} \right) + \text{const.}
\]

The solution to this quadratic problem is
\[
\left( c_t, c_x \right) = -\left( \begin{array}{cc} H_{tt} & H_{tx} \\ H_{tx}^* & H_{xx} \end{array} \right)^{-1} \left( \begin{array}{c} J_t \\ J_x \end{array} \right) = -H^{-1}J.
\]

The matrix $H$ is sparse if the functions $\phi_n$ are such that only a few (neighboring) ones overlap. It is straightforward to add additional regularity constraint by minor modifications of $H$.

### 2.3 Ray interpolation

Given that the directional interpolation vector field components $q_t$ and $q_x$ are known, let us turn our focus to how to solve the partial differential equation (3). This is a transport equation. There are two main approaches for solving this numerically: by finite differences, or by ray tracing. In this paper we will focus on the ray tracing approach. We introduce coordinates $\tau = \tau(t, x, y)$ and $\xi = \xi(t, x, y)$ satisfying the ordinary differential equations

\[
\begin{align*}
\frac{\partial \tau}{\partial y}(t, x, y) &= -q_t(\tau(t, x, y), \xi(t, x, y)), & \tau(t, x, 0) &= t, \\
\frac{\partial \xi}{\partial y}(t, x, y) &= -q_x(\tau(t, x, y), \xi(t, x, y)), \quad \xi(t, x, 0) &= x.
\end{align*}
\]

Now, consider $u(\tau(t, x, y), \xi(t, x, y))$, and in particular how it is changing with respect to $y$, i.e.,
\[
\frac{\partial u}{\partial y} u(\tau(t, x, y), \xi(t, x, y)) = \left( \frac{\partial u}{\partial y} + q_t \frac{\partial u}{\partial t} + q_x \frac{\partial u}{\partial x} \right)(t, x, y) = 0.
\]

We see that $u$ has to be constant along the rays $(\tau(t, x, y), \xi(t, x, y))$. Using this fact the solution to (3) can be computed in two steps: First the rays $(\tau(t, x, y), \xi(t, x, y))$ are computed by solving the ordinary differential equations (8), and after that the solution $u$ is computed at an equally spaced grid.
by interpolating the data along these rays. Given measurements of $u(t, x, y)$ and $\frac{\partial u(t, x, y)}{\partial y}$ at ordered slices $y = y_l$, $l = 1, \ldots, L$, the interpolation can be made in pairs where data are propagated forward and backward, respectively, along the characteristic rays. A linear interpolation between the two of them is used to generate the final result.

This procedure can be implemented using standard routines for computer graphics, in particular standard routines from OpenGL (Open Graphics Library) [6]. The library represents a cross-platform interface for interaction with a graphics processing unit (GPU), to achieve hardware-accelerated rendering. The interpolation procedure can now be realized using GPU vertex and fragment shaders, see [6, Ch.17]. These routines are heavily optimized, and the overall cost of the interpolation procedure will be proportional to that for the GPU I/O operations.

3 Results

We begin by discussing how to choose the functions $\phi_n$ used to describe the DIVF components $q_t$ and $q_x$. It is natural to choose them so that they form a partition of unity in $\mathbb{R}^3$, and are relatively smooth with small overlap. One way of doing this is by using tensor products of scaled and translated functions of the form

$$
\psi(t) = \begin{cases} 
\sin \left(\frac{\pi}{2} (3t^2 - 2t^3)\right)^2, & \text{if } 0 \leq t < \frac{1}{2}, \\
\sin \left(\frac{\pi}{2} (3(1-t)^2 - 2(1-t)^3)\right)^2, & \text{if } \frac{1}{2} \leq t < 1, \\
0, & \text{otherwise}
\end{cases}
$$

It is easy to see that

$$\sum_j \psi(t + j) = 1, \quad (9)$$

and that $\psi \in C^3$. It is also easy to modify if higher regularity is desired. Note also that $\psi(\cdot)$ overlaps only with $\psi(\cdot - 1)$ and $\psi(\cdot + 1)$ in the sum (9), and that if

$$b(t) = \sum_j c_j \psi(t + j),$$
then for each fixed $t$ at most two of the coefficients $c_j$ are contributing to the value $h(t)$. Using a tensor product representation along with a reordering, we can construct a sequence of functions $\{\phi_n\}_n$ such that at each point $(t, x, y)$ at most six coefficients $c_t(n)$ would contribute to the value of $q_t(t)$ and likewise for $q_x$. Keeping this overlap number small is desirable for rapid evaluation of the right hand sides of (8) when computing the characteristic rays of (3). The short overlap in the support of the different functions $\phi_n$ will keep the matrices defined by (6) sparse which is beneficial when solving (7).

### 3.1 Interpolation of the pressure field

In seismic experiments, the crossline component of the pressure field is measured using either geophones or accelerometer. Thus, the measured crossline...
wavefield is a component of either the particle velocity or the acceleration. These measurements can then be related to the crossline spatial derivative, using the static relation

$$\nabla u = -\rho a = -\rho \frac{\partial v}{\partial t},$$

where $\rho$ is the density of the medium, $a$ is the acceleration, and $v$ is the particle velocity.

In the following, we will show and discuss the result of numerical experiments using a simulated data set where $u$ and its crossline derivative are modeled. The data was modeled using an elastic finite difference algorithm, and a 2.5D earth model based on geology from the North Sea. A shot gather of this synthetic dataset is shown in Figure 2. In this figure the data modeled is displayed in a three-dimensional plot that also depicts the sampling setup. Note that the events are connected in a smooth fashion. The available information is shown by the red-blue slices, and the sought for (crossline) information is shown in yellow-blue at inline coordinates $x = [256, 631, 1000, 1381]$. This data looks locally like plane waves, but with some small perturbations including minor curvature. This implies that at regions where there are at most two dominant events, there will be a direction where the data is varying slowly, and hence a direction that is well-suited for conducting interpolation. As the complexity increases, the proposed method aims at choosing the direction with the least variability and performing interpolation along this direction. The data was modeled such that the receivers along the streamer (inline) direction were separated at a distance of 6.25 meters from each other. The sampling in the $y$-direction is 75 meters, which corresponds to an undersampling of a factor 12 in the crossline direction. This direction is also the one where the model and corresponding data have maximal complexity. In total the data set consists of 24 shots. To use as a benchmark, a pressure dataset was also modeled with receivers sampled every 6.25 meters in both horizontal directions.

We begin by studying the quality of the crossline interpolation of the pressure field in the absence of noise. Figure 3 shows the resulting crossline interpolation as well as the true (simulated) values for a fixed inline receiver coordinate. In order to highlight the interesting features, the amplitude has been compensated in depth and the shown information is clipped at 10% of the maximum value.
Figure 3: Interpolation of the pressure component. Left four panels show the sampled data, where the missing traces are shown in white; the middle four the interpolated results; and the right four panels show the true simulated data.
Figure 4: The interpolation result shown in a time slice for $t = 2$ seconds. The top panels shows the sampling of $P$ for four shots, the middle panels shows the corresponding interpolated results; and the bottom four panels shows the true simulated data.
We can see that the interpolated result and the true data agree fairly well. Some differences can be noted, in particular for some of the weak and steep events some of the energy is lost. To further illustrate the reconstruction quality, Figure 4 shows a counterpart of Figure 3, but this time in the form of time slices.

As linear interpolation is conducted along the characteristic rays (between two slices of available information), we can expect some loss of amplitude between the slices due to errors (or model deviation) in the estimated DIVF \((q_t, q_x)\). This is an effect that should increase with frequency, as small variations in directionality will cause a large difference in phase, which leads to cancellation as data is averaged. The effect should thus be revealed by looking at how the (temporal) frequency content varies in the crossline direction. In Figure 5 this effect is illustrated. We can see that the interpolation demonstrates good results also in this test. The \(FK\)-plot clearly shows that the undersampling problem is no longer present, and the \(FY\)-plot shows that the crossline amplitude loss is comparatively small.
Figure 5: The left image shows a cross plot of the temporal frequency content. In the right image both the temporal and spatial ($F_Y$) frequency content is shown, with the subsampled data to the left; the interpolated result in the middle; and the true data to the right.
3.2 Interpolation of the Z-component

We now turn our focus to the crossline interpolation of the depth derivative \( Z \). In this section we will provide a heuristic derivation, but a more formal derivation is given in appendix A.2. We start by extending our data to depth, i.e., let \( u = u(t,x,s,y,z) \). For a fixed depth \( z_0 \), the relation (3) holds, and hence

\[
\frac{\partial u}{\partial y} + q_t \frac{\partial u}{\partial t} + q_x \frac{\partial u}{\partial x} = 0, \quad z = z_0.
\]  

Now, at a slightly different depth the wave field will have slightly shifted events, but we do not expect the directionality to change much (the measurements are assumed to be made in a constant velocity, isotropic medium (water)). Hence, it seems plausible that if \( q_t \) and \( q_x \) were to be extended in depth as well, they would not only vary slowly at a fixed depth but also between depths. Hence, if we apply a partial derivative with respect to \( z \) to (10) we have

\[
\frac{\partial u_z}{\partial y} = \frac{\partial u_z}{\partial y} = -\frac{\partial z \left( q_t \frac{\partial u}{\partial t} + q_x \frac{\partial u}{\partial x} \right)}{z} 
\approx -\left( q_t \frac{\partial u_z}{\partial t} + q_x \frac{\partial u_z}{\partial x} \right).
\]

This implies that the same DIVF \((q_t, q_x)\) can be used to make crossline interpolation of both the pressure and the depth derivative.

We will now verify this statement by simulations. The available simulated data did not include a dense crossline sampling of the depth derivative. However, it is well known that if both \( P \) and \( Z \) are known, the weighted average of the two should effectively attenuate ghosts (deghosting by PZ-summation) [24]. A more refined deghosting can be expressed in the Fourier domain and accounts for directionality effects [3]. For the sake of transparency we use the simple average in this example. If the \( Z \)-component is correctly interpolated this should show in the sense that the PZ-summation yields a deghosted result.

Figure 6 shows the results after interpolation of \( P \) and \( Z \) components, as well as the resulting PZ summation. The data are shown for one specific channel, and for all 24 shots. The data are interpolated to samples every 6.25m, obtained from a decimated sampling of a factor 12 (75m crossline sampling).
Figure 6: Left panels: Crossline plots for 24 sources at $x = 1000$. Top panel shows the interpolated $P$ component, the middle panel the interpolated $Z$ component, and the bottom panel the corresponding $PZ$ summation. The vertical lines show shot 12. Right panels: Crossline plot for source 12, inline offset $x = 1000$, and $[0,2.5]$ time interval. The left panel shows the interpolated $P$-component, the middle panel shows the interpolated $Z$-component, and the right panel shows the corresponding $PZ$ summation.

for the three components. The data have been $t^2$-compensated and clipped to $\pm 10\%$.

### 3.3 Noise in the crossline component

In practice, marine towed streamer measurements of the $Y$-component will have a substantially higher level of noise contamination; in particular, its low
temporal frequencies are highly affected by noise. In Figure 7 average spectra are shown for the $P$, $Y$ and $Z$-components in the used synthetic data set. In the simulations of this section noise with the average amplitude spectrum indicated by red is added to the $Y$ and $Z$ components. Note that the signal to noise ratio is rather small for the $Y$ component in this case, and in particular small for the lower frequencies. A main advantage with the proposed method is that it only uses the $Y$ component to estimate the directional interpolation vector field. A reasonable assumption on data is that the local plane waves are imprints of the (broadband) source wavelet. This implies that energy is consistently present for low, medium and high frequencies along the directions of the planes, and hence the estimation of the directional interpolation vector field should be fairly robust to filtered out frequency band of the signal. The $P$ component is, of course, filtered in the same way when conducting the estimation of the directional interpolation vector field. Figure 7 shows the sum of the absolute value of the Fourier transform of the $Y$-component for all traces, along with the filter used for estimating directions. Note also that due to the presence of the derivatives in the estimation of $q_t$ and $q_x$, see e.g. (6),
the high frequency part will be weighted much higher than the low-frequency part. It therefore seems reasonable that good estimates of the DIVF can be obtained by simply disregarding the low-frequency information of the $P$ and the $Y$-components when estimating $q_t$ and $q_x$. Note that this does not mean any loss of low-frequency information in the final interpolation, as the full frequency range of $P$ can be used in the interpolation step when solving the PDE of (3).

In Figure 8 the effects of the noise in the $Y$ component on the interpolation is shown. The figure shows crossline plots for groups of two shots. The sampling of $P$ is shown to very left, and then the corresponding sampling of $Y$ is shown. Next the noisy sampling is shown, with noise colored according to Figure 7. Following that, are the two interpolation results using the $Y$ without and with noise respectively, and to the very right the difference between the two interpolations is shown. We can see that this difference is remarkably small given the low signal to noise ratio of noisy $Y$ component. The effect is analogous for all of shots in this dataset.

### 3.4 Noise in the depth derivative

The treatment of high noise levels in the low-frequency parts of the $Z$-component requires a different strategy than that for the $Y$-component. Here it is important to keep and recover the low-frequency information, as it is needed when interpolating between slices. We propose to make use of the fact that typical seismic events are expected to be visible in several of the crossline slices, and that the noise between receivers is independent. To illustrate the idea, consider the illustration in Figure 9. The left group of traces shows a horizontal plane wave. The middle group shows the same traces, but now with noise contamination. In the case where we know that the traces represent a horizontal plane wave, we can perform averaging between the traces. The right group of traces shows the effect of doing this. We can see that the noise is substantially reduced.

We can apply a similar idea for suppressing noise in the $Z$ component, since we know the local directionality structure. Diffusion can be applied between slices by making averaging instead of linear interpolation along the characteristic rays. In contrast to before we will thus not preserve the information at the slices $y = y_l$. Now, if we know that the noise is predominantly in the low-frequency range, we can choose to keep the high frequency intact
and perform diffusion between slices in the low frequency parts. In the simulations presented here we satisfy with dividing the data into two parts, one that is kept intact and one were diffusion is applied. For some noise colors it could be useful to use a more refined scheme where the amount of diffusion depends on the frequency content.

In Figure 10 the results of the interpolation of \( Z \) using noisy values of \( Y \) and \( Z \) are shown in the inline direction. The left panel shows the pressure \( P \), next the noise samplings of \( Y \) and \( Z \) are shown, respectively. Following that is the interpolated result for \( Z \), \( P \), and \( PZ \), respectively. We can see in the interpolation of \( Z \) that the noise have been heavily suppressed. Some effects

Figure 8: The effect of noise in the \( Y \) component. From left to right: \( P \) sampled; \( Y \) without noise; \( Y \) with noise; \( P \) interpolated without noise; \( P \) interpolated with noise; and the difference between the two interpolations. at \( x = 631 \) m.
are still visible, and the effects of the directionality are also visible. Note that the noise suppression shown here is only due to the crossline diffusion. It is possible that these results could be improved further by applying additional in-slice processing.

In Figure 11 the results of the interpolation of Z using noisy values of Y and Z are shown in the crossline direction. Here the noisy sampling of Z is shown in the leftmost panel, and the interpolations for Z, P and PZ are shown in the panels from left to right. Again, we can see a substantial noise reduction in the Z component, while the crossline interpolation is still quite successful. This result agrees with the other inline coordinates as well as shots.

4 Conclusions

A method is presented for the interpolation of pressure and depth derivative data by estimation of direction in the slices where the data to be interpolated are available and well sampled (inline), in addition to the derivative of the
Figure 10: Inline plot of the effect of noise in the Z and Y components. From left to right: P sampled; Y sampled with noise; Z sampled with noise; Z denoised and interpolated; P interpolated; and PZ denoised and interpolated pressure field in the direction where the interpolation is desired (crossline, in this case). The idea is to estimate the local directionality from the available data and from this information locally choose the direction with the minimum variability and conduct the interpolation in this direction. This procedure will produce perfect interpolation in the case where there is less than or equal to two local directions in the data. The approach can be extended so that perfect results can be obtained when there are more than two simultaneous directions present, at the cost of having to estimate more parameters. From the simulations presented, the suggested approach seems to do well in practice.

This method does not rely on any underlying assumption about the water
velocity, or any assumption on the sea surface boundary condition. Thus, it is an entirely data-driven algorithm. In this work, we only presented up/down separated data using direct $PZ$ summation. The more complete formulation to separate the up-going and down-going pressure field components contains an obliquity factor compensation that is not used in $PZ$ summation. The addition of the 3D obliquity factor is straightforward after interpolation of $P$ and $Z$ using extended structure tensors. After interpolation, both inline and crossline directions are well sampled, thus, both directions can be Fourier transformed to take the data to $\tau - p - q$ or $f, k_x, k_y$ domain, where the obliquity factor can be described as a multiplicative factor. The method is very robust to low-frequency noise in both the $Z$ and the $Y$-components.
Extensions and applications of this method to interpolate other data types (e.g., electromagnetic data and seismic measured at the ocean bottom or in a vertical seismic profile) are possible. The problem of interpolating ocean bottom seismic (OBS) data is one of particular interest. In OBS experiment, the pressure field and the particle velocity vector are commonly measured, resulting in measuring two horizontal components of the velocity, $X$ and $Y$, in addition to $Z$ and the scalar field $P$. As opposed to streamer seismic, OBS $X$, $Y$, and $Z$ measurements do not have strong low frequency noise, as these are purely static measurements. The challenge for interpolation is the fact that the receiver stations are more sparse than on streamer data. When cables are used, one direction is better sampled (inline sampling averages then 50m) than the orthogonal one (200-400m are common in crossline sampling rate). When nodes are used, both directions tend to be equally sparse (often in the order of 200-400m). The source is well sampled in both horizontal direction, often ranging from 37.5 to 50m sampling rate. Thus, the most challenging interpolation is on the receiver side. In addition to the sparse sampling, the receiver side poses a problem different from streamer data: The directional components measure both pressure field and shear waves. Because of this, it might be less challenging to interpolate the $Z$ component using $Z$ and $Y$ to calculate the DIVF, because they both contain complementary pressure and shear waves. Thus, the interpolation of the pressure field, based on the theory presented here, has to be updated to deal with the physics of scalar and vector measurements in elastic media. A possibility for the interpolation of the pressure field, using $Y$ would be to precondition $Y$, by considering the shear wave information and directionality as noise and removing it prior to using the method presented here. This is the subject of current research and we will report on this in the future.

A Appendix

A.1 Extended structure tensors

Assume that $u : \mathbb{R}^3 \to \mathbb{R}$. Let $\tilde{u} : \mathbb{R}^3 \to \mathbb{C}$ denote the analytic part of $u$, i.e.,

$$
\tilde{u}(t, x, y) = \int_0^\infty \hat{u}(\omega, x, y)e^{2\pi i t \omega} d\omega.
$$

170
We use the $\hat{u}$ notation to denote Fourier transforms. To reduce complexity, we indicate implicitly on which dimensions the Fourier transform acts by the variable names used. The function $\hat{u}$ can be extended to $\hat{u}_{\text{ext}} : \mathbb{R}^4 \to \mathbb{C}$ by the one way wave operator in the $(t,x)$-plane,

$$\hat{u}_{\text{ext}}(t,x,y,s) = \int_0^\infty \int_{-\infty}^\infty \hat{u}(\omega,k_x,y)e^{2\pi i (t\omega + xk_x + s\sqrt{\omega^2 + k_x^2})} dk_x d\omega.$$ 

By abuse of notation, we identify

$$\partial \hat{u}_{\text{ext}} / \partial s(t,x,y) = \partial \hat{u}_{\text{ext}} / \partial s(t,x,y,0).$$

We now define the *extended structure tensor*. Let

$$T_u = T_u(t,x,y) = \text{Re} \left( K^* \begin{pmatrix} \frac{\partial \hat{u}}{\partial t}^2 & \frac{\partial \hat{u}}{\partial t} \frac{\partial \hat{u}}{\partial x} & \frac{\partial \hat{u}}{\partial t} \frac{\partial \hat{u}}{\partial s} & \frac{\partial \hat{u}}{\partial t} \frac{\partial \hat{u}}{\partial y} \\ \frac{\partial \hat{u}}{\partial t} \frac{\partial \hat{u}}{\partial x} & \frac{\partial \hat{u}}{\partial x}^2 & \frac{\partial \hat{u}}{\partial x} \frac{\partial \hat{u}}{\partial s} & \frac{\partial \hat{u}}{\partial x} \frac{\partial \hat{u}}{\partial y} \\ \frac{\partial \hat{u}}{\partial t} \frac{\partial \hat{u}}{\partial s} & \frac{\partial \hat{u}}{\partial x} \frac{\partial \hat{u}}{\partial s} & \frac{\partial \hat{u}}{\partial s}^2 & \frac{\partial \hat{u}}{\partial x} \frac{\partial \hat{u}}{\partial y} \\ \frac{\partial \hat{u}}{\partial t} \frac{\partial \hat{u}}{\partial y} & \frac{\partial \hat{u}}{\partial x} \frac{\partial \hat{u}}{\partial y} & \frac{\partial \hat{u}}{\partial s} \frac{\partial \hat{u}}{\partial y} & \frac{\partial \hat{u}}{\partial y}^2 \end{pmatrix} \right),$$

where $K$ is a fixed Gaussian.

Following the approach given in [4], it is readily verified that if $u$ consists locally of fewer than or exactly three plane waves, then $\text{rank}(T_u) \leq 3$. Thus, there is a function $q : \mathbb{R}^3 \to \mathbb{R}^4$, refer to as the *directional interpolation vector field*, with components

$$q(t,x,y) = (q_t(t,x,y), q_x(t,x,y), q_s(t,x,y), -1)^*,$$

such that

$$T_u(t,x,y)q(t,x,y) = 0, \quad \forall (t,x,y).$$

Concerning the other implicated direction, if (11) is true, then it holds approximately that $\hat{u}_{\text{ext}}$ does not vary along the direction indicated by $q$, i.e.,

$$\partial \hat{u}_{\text{ext}} / \partial y = q_t \partial \hat{u}_{\text{ext}} / \partial t + q_x \partial \hat{u}_{\text{ext}} / \partial x + q_s \partial \hat{u}_{\text{ext}} / \partial s.$$
Note that $\frac{\partial \tilde{u}_{\text{ext}}}{\partial s}$ can be computed in the hyperplane $s = s_0$. Thus, the transport equation (12) can be solved in the hyperplane $s = 0$, i.e., for $(t, x, y) \in \mathbb{R}^3$ only.

From knowledge of $u(t, x, y)$ and $\frac{\partial u}{\partial y}(t, x, y)$ at the planes $y \in \{y_l\}_{l=1}^L$, the following scheme for conducting interpolation can then be used:

1. Estimate the directional interpolation vector field $q = q(t, x, y)$ from pressure and crossline derivative information at $y \in \{y_l\}_{l=1}^L$.

2. Conduct interpolation by solving

$$\frac{\partial u}{\partial y} = q_t \frac{\partial u}{\partial t} + q_x \frac{\partial u}{\partial x} + q_s \frac{\partial u}{\partial s}, \quad (12)$$

using values of $u = \tilde{u}_{\text{ext}}$ at $y \in \{y_l\}_{l=1}^L$ as initial values, in order to find the values of $u(t, x, y)$ for $y \notin \{y_l\}_{l=1}^L$.

### A.2 Indirect estimation of the directional interpolation vector field

Suppose that $G(t, x, y, z)$ is a solution to the homogeneous wave equation

$$\frac{\partial^2 G}{\partial t^2} - c_0^2 \Delta(G) = 0. \quad (13)$$

First consider the case of a single plane wave, e.g.,

$$G(t, x, y, z) = e^{2\pi i(t \omega_t + x \omega_x + y \omega_y + z \omega_z)}.$$

Because

$$\left( \frac{\partial^2}{\partial t^2} - c_0^2 \Delta \right) G = -4 \pi^2 \left( \omega^2 - c_0^2 \left( k_x^2 + k_y^2 + k_z^2 \right) \right) G = 0,$$

it follows that $(\omega, k_x, k_y, k_z)$ must satisfy the relation

$$k_z = \pm \sqrt{\frac{\omega^2}{c_0^2} - (k_x^2 + k_y^2)}.$$

The two solutions correspond to down-going and up-going waves, respectively. By Fourier decompositions we conclude that every solution to the homogeneous wave equation (13) has to be of the form
Directional interpolation of multicomponent data

\[
G(t, x, y, z) = \int \int \int \left( \hat{G}_{\text{up}}(\omega, k_x, k_y)e^{-2\pi i k_z} + \hat{G}_{\text{down}}(\omega, k_x, k_y)e^{2\pi i (\omega t + k_x x + k_y y)} \right) \, d\omega \, dk_x \, dk_y,
\]

where \( k_z = k_x (\omega, k_x, k_y) = \sqrt{\omega^2/c_d^2 - (k_x^2 + k_y^2)} \).

Let us now fix the depth parameter \( z = z_0 \), and suppose that \( u(t, x, y) = u(t, x, y, z_0) \) is a real valued solution to (13). We also assume that \( u \) is locally composed of \( J \) plane waves, i.e., that

\[
u(t, x, y) = \sum_{j=1}^{J} u_j(n_{t,j} t + n_{x,j} x + n_{y,j} y),
\]

where the vectors \( n_j = (n_{t,j}, n_{x,j}, n_{y,j}) \) specify the normals of the plane waves. This means that \( \hat{u} = 0 \) unless \( (\omega, k_x, k_y) \) lies on one of the rays associated with \( n_j \). Given \( \{n_j\}_{j=1}^{J} \), we say that \( u \in \mathcal{U}_{n} \) if \( u \) is such that

\[
\hat{u}(\omega, k_x, k_y) = 0 \quad \text{if} \quad (\omega, k_x, k_y) \parallel n_j, \quad \forall j.
\]

Now, if \( J \leq 3 \), the extended structure tensor \( T_u \) contains the necessary information to recover the directions \( n_j \) from \( u \) if \( u \in \mathcal{U}_n \); and if \( J = 3 \) and \( u, v \in \mathcal{U}_n \) and \( \text{rank}(T_u(t, x, y)) = \text{rank}(T_v(t, x, y)) \), then \( \text{null}(T_u(t, x, y)) = \text{null}(T_v(t, x, y)) \) meaning that there is a unique solution \( q \) to

\[
T_u(t, x, y)q = T_v(t, x, y)q = 0.
\]

Now, from (14) we see that

\[
\hat{u}(\omega, k_x, k_y) = \hat{u}_{\text{up}}(\omega, k_x, k_y)e^{-2\pi i k_z} + \hat{u}_{\text{down}}(\omega, k_x, k_y)e^{2\pi i k_z}.
\]

We now want to show that \( u_{\text{up}}, u_{\text{down}} \in \mathcal{U}_n \). To do this, let us assume that \( (\omega, k_x, k_y) \) is such that there is no \( j \) such that \( (\omega, k_x, k_y) \parallel n_j \). The same thing then clearly holds for \( (-\omega, -k_x, -k_y) \). Because

\[
\hat{u}(\omega, k_x, k_y) = \hat{u}(-\omega, -k_x, -k_y) = 0,
\]

we have that

\[
\hat{u}_{\text{up}}(\omega, k_x, k_y) = -\hat{u}_{\text{down}}(\omega, k_x, k_y)e^{4\pi i k_z (\omega, k_x, k_y)},
\]

\[
\hat{u}_{\text{up}}(-\omega, -k_x, -k_y) = -\hat{u}_{\text{down}}(-\omega, -k_x, -k_y)e^{4\pi i k_z (-\omega, -k_x, -k_y)}.
\]
We can now use the fact that

\[ k_z(-\omega, -k_x, -k_y) = k_z(\omega, k_x, k_y) \]

in combination of a complex conjugation to rewrite the second equation as

\[ \widetilde{\mu}_{\text{up}}(-\omega, -k_x, -k_y) = -\widetilde{\mu}_{\text{down}}(-\omega, -k_x, -k_y)e^{-4\pi i z k_x(\omega, k_x, k_y)}. \]

By using the symmetry inherited from the fact that \( u \) is real valued, we can write this as

\[ \hat{\mu}_{\text{up}}(\omega, k_x, k_y) = -\hat{\mu}_{\text{down}}(\omega, k_x, k_y)e^{-4\pi i z k_x(\omega, k_x, k_y)}. \quad (18) \]

By combining (17) and (18) we can now conclude that

\[ \hat{\mu}_{\text{up}}(\omega, k_x, k_y) = \hat{\mu}_{\text{down}}(\omega, k_x, k_y) = 0, \]

and hence it follows that \( u_{\text{up}}, u_{\text{down}} \in \mathcal{U} \).

For the \( z \)-derivative of \( u \), it holds that

\[ \frac{\partial \widetilde{u}}{\partial z} = 2\pi i k_z \left( -\hat{\mu}_{\text{up}}e^{-2\pi i z k_z} + \hat{\mu}_{\text{down}}e^{2\pi i z k_z} \right), \]

and from the argumentation above we conclude that

\[ u \in \mathcal{U} \Rightarrow \frac{\partial u}{\partial z} \in \mathcal{U}. \]

An important practical implication follows from this fact. Because of (16), we can use the same directional interpolation vector field both for interpolating the \( P \) component and the \( Z \)-component. This means that we do not need to know the derivative of the \( Z \) component with respect to \( y \) for the proposed interpolation method to be applicable for interpolating the \( Z \)-component.

In general, it holds for any pseudo-differential operator \( \mathcal{P} \) that

\[ u \in \mathcal{U} \Rightarrow \mathcal{P}(u) \in \mathcal{U}. \]

An important example concerns filtering in the time domain. These results imply that we can compute the directional interpolation vector field using filtered values of \( P \) and \( Y \), as long as we do not filter out too much of the
energy in the different directions. Because a broadband measurement setup is such that the source output has a broad spectrum, we anticipate that the estimation of the directional interpolation vector field can be sufficient if only the low frequency parts of the spectrum are excluded. In practice this is quite important, as it is difficult to design towed streamer hardware capable of accurately measuring the low frequency components in the \( Y \) and \( Z \) components. In the proposed setup we can instead use the medium and high frequency parts of the \( Y \)-component (and similar parts of \( P \)-component) to estimate the directional interpolation vector field, which then can be used in combination with the \( P \) (or \( Z \)) component alone to interpolate data between streamers.

**Acknowledgments**

This work was supported by the Crafoord Foundation (2014-0633) and the Swedish Research Council (2011-5589, 2015-03780)

**References**


Parallel algorithm of 3D wave-packet decomposition of seismic data: Implementation and optimization for GPU

Viktor V. Nikitin  Anton A. Duchkov  Fredrik Andersson

Abstract

In this paper, we consider 3D wave-packet transform that is useful in 3D data processing. This transform is computationally intensive even though it has a computational complexity of $O(N^3 \log N)$. Here we present its implementation on GPUs using NVIDIA CUDA technology. The code was tested on different types of graphical processors achieving the average speedup up to 46 times on Tesla M2050 compared to CPU sequential code. Also, we analyzed its scalability for several GPUs. The code was tested for processing synthetic seismic data set: data compression, de-noising, and interpolation.

1 Introduction

Reflection seismology is one of the most powerful methods of exploration geophysics. It uses artificially generated elastic/acoustic waves to study the Earth’s subsurface structure and properties. During seismic exploration surveys sources at the earth’s surface are used for generating seismic waves propagating into the subsurface. Waves reflected from geological boundaries propagate back to the surface where they are recorded by geophone arrays. The recorded wave field can then be used for constructing structural images of the subsurface boundaries, which in turn can be used for mineral resources exploration [7].

A schematic illustration of a marine seismic survey depicted in left panel of Fig. 1, where a ship is towing a source (air gun) and a long floating array of receivers (hydrophones). The reflection response (pressure recorded by
Parallel algorithm of 3D wave-packet decomposition of seismic data: Implementation and optimization for GPU

Figure 1: Seismic exploration; left – marine acquisition with incident and reflected rays schematically illustrating wave propagation down from the source and the reflected wave propagation up to the receivers, right – shot gather, pressure amplitude as a function of receiver coordinate and time.

hydrophones) for one source pulse, the shot gather, is shown in the right panel of Figure 1. Source pulses are repeated on intervals of a few seconds while the vessel is moving along straight line. This produces a 3D cube of seismic data for each one profile. Some of the main properties of seismic data are: multidimensionality (seismic data ranges from 3D to 5D); large data size (several Gbs for one profile, several Tbs for survey); irregular structure (missing and corrupted traces, distorted profiles, etc.).

Methods developed in image processing have been used for some time by seismic researchers. One of such methods is to decompose data (image) using specific redundant representations (redundant “basis” functions) – localized plane waves or wave packets. Popular classes of this type include dyadic parabolic decompositions [13], curvelets [4], shearlets [11], brushlets [12], beamlets [14], Gaussian wave packets [1], etc. A main advantage of such decompositions is that many functions with wave-like structure can be well approximated using a linear combination of a small number of these “basis”
functions. Wave-packet representations can be a useful tool in several seismic processing tasks: data compression, de-noising, and regularization (recalculation from irregular grid to a regular one). A new basis of 3D wave packets was recently developed specifically for seismic applications [8]. In this paper we address the problem of developing a fast parallel implementation of 3D wave-packet transform pairs working on GPUs. We aim at optimizing the code for processing large volumes of data.

2 Wave-packet decomposition

In a forward wave-packet transform (WPT) a sampled function \( f(x) \) is mapped to a set of coefficients \( \{c_\gamma\} \),

\[
C : f(x) \rightarrow \{c_\gamma\}, \text{ s.t.} \\
\sum_{\gamma} c_\gamma \phi_\gamma(x) = f(x)
\]  

(1)

where \( \phi_\gamma(x) \) is a wave packet parameterized by \( \gamma \).

Each wave packet has compact (numerical) support in the Fourier domain. In particular, this support is contained in boxes of prescribed sizes and orientations. The different boxes in this tiling correspond to wave packets with different orientation and scale, and form a partition of unity. Thus, wave packets create a tight frame (redundant “basis”) for the decomposition of 3D functions. This also means that the inverse transform is simply equal to the adjoint operation of the forward transform.

Let us demonstrate the computational procedure for a discrete forward WPT (see details in [5]):

- Division in the spatial domain;
- "large" forward Fourier transform for \( f(x) \) on the global grid;
- short convolution operations for the evaluation of the Fourier transforms on local grids defined on each box;
- multiplication of a spectrum on each box by a weighting function (pre-computed parts of a partition of unity);
- “small” inverse Fourier transforms for each box.
The final output will be an approximation (to prescribed accuracy) of the wave-packet coefficients \( \{ c_\gamma \} \). The above procedure allows for rapid computation of wave packet coefficients. The first three steps are part of unequally spaced fast Fourier transforms (USFFT), cf. [2, 6]. The spectrum is typically highly oscillatory, and naive interpolation methods would perform poorly.

The computational complexity of such algorithms is the same as for standard Fast Fourier Transform \( \mathcal{O}(N \log N) \). However, it contains \( CN \) operations, where the constant \( C \) is typically high enough to dominate the \( \log N \) behavior in practice. The USFFT computations will therefore dominate the computational cost of wave-packet transforms. It is used for computing spectrum values on irregular grid that is a union of local grids defined on all boxes in the Fourier domain. Further in the text we focus on the most computationally intensive part of the USFFT itself, which consists of weighted summations (short convolutions).

**Gathering.** In forward WPTs we use USFFT to map input function \( f(x) \) defined on the regular global grid to its spectrum \( F(\omega') \) defined on rotated grids as schematically illustrated, for 2D case, in the left panel of Figure 2. As a part of this procedure we need to do weighted summations as shown in the right panel of Figure 2. Here, values on the local rotated grid (shown with crosses) are computed, given the values on the global regular grid (shown with grey squares). The gathering operation is a weighted summation: for each point \( \omega \) of the local grid (crosses) we sum values from the regular grid (squares) located in the cube of radius \( hq \) around it (depicted by a dashed
Parallel algorithm of 3D wave-packet decomposition of seismic data: Implementation and optimization for GPU

square in the right panel of Figure 2) with some Gaussian weight $w(r)$:

$$F(\omega'_x, \omega'_y, \omega'_z) = 
\sum_{i,k,j=-hq}^{hq} w(i+d_x) w(j+d_y) w(k+d_z) F(\omega_x+i, \omega_y+j, \omega_z+k),$$

(2)

where $\omega = (\omega_x, \omega_y, \omega_z)$ is the point of the local grid (crosses), and $\omega' = (\omega'_x, \omega'_y, \omega'_z)$ is the point of the global grid (grey squares) closest to $\omega$, and $(d_x, d_y, d_z)$ are distances between $\omega$ and $\omega'$, $(i, j, k)$ are running indices; $w$ is the weighting function, $\lambda$ is the coefficient depending on the required interpolation accuracy.

Scattering. For implementing the inverse WPT we must reverse the operations from the previous subsection: one has to write values from every point of the local grid (crosses) to points of the global grid (gray squares) with different weights $w(r)$. This scattering operation is performed in the sphere with radius $hq$ (dashed square in Figure 2, right).

Let us summarize some characteristics of computations. First, it is sufficient to use single-precision operations. Second, the WPT algorithm has very large amount of simple operations (weighted summations). Finally, it is necessary to organize access to large data volumes during the weighted summations. Thus, graphical processing units (GPU) are suitable to use for these operations. The main resource-intensive parts of the WPT algorithm are interpolation procedures (gathering and scattering); and Fourier transforms. In the next section, we describe how these were ported to GPU platforms.

3 Porting and optimization on a GPU

There are hundreds of boxes used for tiling the Fourier domain. The total number of grid points on all boxes (irregular grid) varies between 100 and 500 thousand points. For the short convolutions, it is necessary to compute weighted sums around each of these points (one summation consists of several thousands of elementary operations).

We used CUDA technology for porting these computations to a GPU platform. For testing the code we used synthetic data cubes with up to $512^3$
samples.

3.1 Fourier transforms on GPU

FFT algorithms are used twice in our transform: one large FFT operation on the global grid, and many smaller FFT operations on each box. CUFFT library is used for implementing FFT on a GPU. The last version of CUDA allows obtaining a speed up $\sim 25$ times compared to the FFTW library on a CPU (see CUDA Toolkit v. 4.1).

3.2 Short convolution operations on GPU

The scatter and gather operations can be implemented for each box independently if we organize access to the function spectrum on the global grid. In case the spectrum does not fit in the device memory, it should be divided into sub-domains large enough to contain data for several boxes. One sub-domain (shown by a dashed square in Figure 2, left) is loaded into the device memory at the time for processing a group of boxes.

In the forward WPT we implement gathering using the same number of GPU threads as the number of points in the box ($1-3 \times 10^5$ threads). Each thread computes a weighted sum (2) for one point (crosses in the right panel of Figure 2).

For the inverse WPT we can also set the number of threads for scattering equal to the number of points in the box. However, we need to scatter data around each point writing it to the global grid (gray squares Figure 2, right). One can easily see that scattering regions (dashed rectangles in Figure 2, right) are overlapping, which may results in racing conditions (threads attempt writing to the same memory cell simultaneously) and wrong final result.

The use of atomic operations (resource lockout before writing data) is unattractive because there can be more than 700 threads trying to access to the same memory cell simultaneously. To avoid data writing conflicts, we choose instead to modify the algorithm.

We decided to substitute scattering by the gathering operation for the inverse WPT as well. This can be done by making use of the particular geometry. The loop will not run over local grid points (crosses) in this case but
over global grid points (gray squares), cf. Figure 2, right. Weighted summation (2) is performed in the dashed rectangle, gathering values from local grid (crosses). There will be only one write operation for global grid points (gray squares).

Thus, both the weighted summation operations needed in the forward and inverse transform will be implemented using the gathering operation (2). One kernel (function implemented on a GPU) contains a triple loop \(0 \leq i \leq 10, 0 \leq j \leq 10, 0 \leq k \leq 10\) with a total of 1331 iterations. It also contains a step where it is checked if a point lie near the box \((\text{range} \leq hq)\), calculating the weight function \(w(r)\), and readings elements from the global GPU memory. After all these operations a resulting value can be computed. This triple loop is very computationally intensive and it needs to be optimized.

### 3.3 Optimization of smearing

We have optimized the code in the following main directions [3, 10]:

- optimization of inner loop operations,
- optimization of instructions,
- memory optimization,
- optimization of GPU occupancy,
- optimization of the gather operations for the inverse transform.

Some of them are discussed below.

#### 3.3.1 Memory optimization

GPU memory bandwidth has a strong influence on the code performance. We tried to obtain real data throughput for our program to close to the theoretical memory bandwidth.

For reading spectrum values each thread must connect the device global memory. Also note that this operation is located in a nested triple loop. Modern devices (Fermi with compute capability 2.0) have L1 cache for each multiprocessor. But in our program it will be used ineffectively, because data access pattern is semichaotic, and multiprocessors do not have access to each other’s
L1 cache. Due to many cache misses (more than 90%) we decided to turn off L1 cache.

In case of semi-chaotic data access pattern we can use shared memory and texture memory for avoid memory latency. Shared memory is small and it is physically located on each streaming multiprocessor (SM). Thus, one SM cannot access shared memory of another SM. Texture memory in CUDA appeared to very useful for our algorithm. Accesses to the global memory using texture reference are cached; texture cache is the same for all SMs. When there is a cache hit GPU spends only 4 cycles for reading compared to the global memory access that takes 400 cycles.

Using registers is another important stage in memory optimization. For example, for computing the weighted sum (2) in the internal loop each thread writes intermediate summation result to the global memory. For each iteration in the loop GPU spends 400 cycles. We gained performance by using a register variable for storing intermediate results within the loop.

A lot of variables are created in the kernel during its execution. In order to avoid spilling the registers (when some variables will be placed in a slow device memory) we made use of constant and shared memory with a latency time of 4 cycles. For storing constants we created a structure and placed it in the constant memory. Also some local variables were placed in shared memory.

Finally we achieved an increase in actual data throughput from 20% (before optimization) up to 70% of the theoretical memory bandwidth.

3.3.2 Optimization of GPU occupancy

Sizes of register and shared memory and also block size (BS) have a big influence on GPU performance. We analyzed our program using NVIDIA Compute Visual Profiler. This profiler makes it possible to measure runtime for separate program parts, number of cache misses, bank conflicts, etc. Also we used CUDA Occupancy Calculator for finding optimal BS, amount of registers per thread and shared memory per thread block. In particular, optimal block size appeared to be 336.
3.3.3 Optimization of weighted summation operations for the inverse transform

As described earlier in the text we had to modify the inverse transform code so that to implement scattering as a gathering operation rather in order to avoid memory access conflicts. We are now looping through the global grid points (gray squares) performing weighted summation of the values from the local grid points (crosses) located in the cube of radius $hq$ around it. Apparently we do not want to loop through the whole global grid which is rather big compared to a small box so that most of points will be far away from the local grid of interest (crosses). The checking if it belongs to the region of interest happens outside the main loop and takes only 20 operations, i.e. less than 1% of all operations.

On a GPU each thread block is executed on one SM, on one SM can be used for running several thread blocks. If we use 1D thread blocks we will pass through the grid sequentially. Many threads in a warp (warp is a group of threads executed physically in parallel on GPU) will perform incomparable amount of operations due to being outside or inside the region of interest (divergent branches). The minimal amount of operations for a thread is 20, maximal – 6000. For optimization of this process we made 3D grids of threads in a block. We adjusted these values analyzing statistics from Compute Visual Profiler and CUDA Occupancy Calculator. Finally, we have found optimal parameters maintaining the GPU occupancy at the level of 67% while reducing the number of divergent branches from 80% down to 15%.

4 Using several GPUs

For processing large data sets it is important to check algorithm scalability for several GPUs. Examples of better scalability for multicore over multi-GPU configurations are known in the literature [9]. We modified our code for running on several GPUs using OpenMP streams and analyzed its scalability for large number of GPUs. We tried two variants of multi-GPU implementation.

Variant 1. In the first variant we started with computing the large FFT operation on 1 GPU, and then distributed the weighted summation operations and the small FFT computations (corresponding to different boxes) to different GPUs. The processing time does not need to be the same for differ-
ent boxes, and to account for that complicated logistics is required for collecting data back correctly while forming the final output. The code was tested on 1 and 2 GPUs and then Amdahl’s law was used to predict its scalability for more GPUs. We predicted only $3.5 \times$ speedup for 8 GPUs. There is no sense in using many GPUs in this case because there are many sequential operations (the large FFT operation is executed on one GPU, there is intensive data transfer between CPU and GPU, etc.).

**Variant 2.** In this case we exploited a specific structure of seismic data which has a limited frequency band generated by the source. This absence of very low frequencies allows for dividing large data volume into smaller blocks and processing them independently. Some very fast postprocessing is necessary to merge wave-packet coefficients (or image blocks) back together. Our experiments showed that block size $256^3$ is sufficient for subdividing any larger data sets.

We have testing the second variant of the code on two GPUs and then applied Amdahl’s law. We obtained almost linear speedup for this case: the sequential part of the code is less than 1% and the theoretical performance gain for 4 and 8 GPUs is 3.93 and 7.70, respectively. It allows the usage of a large number of GPUs simultaneously for processing large data volumes.

## 5 Testing for seismic applications

Wave-packet decomposition is providing sparse representation of seismic data (cf. [4]), i.e. these data can be well approximated by a small amount of large wave-packet coefficients. Moreover, wave-packet transform can be used for separating noise from target waves because random noise is scattered along large number of small coefficients and coherent noise will be correspond to wave packets with different orientations (see [8]).

Suppose that we have a cube of seismic data contaminated with noise or with missing traces (Figure 5, left). In the figures on the cube sides we show sections through the data cube corresponding to thin lines passing through the middle of the cube. The procedure of de-noising and traces filling can be done by applying forward WPT, thresholding (keeping only coefficients with absolute value above certain level) followed by an inverse WPT using the threshold coefficients:

$$C_f(x) = \{c_γ\}, \quad \{c_γ\} \rightarrow thresholding \rightarrow \{c_γ'\}, \quad C^{-1}\{c_γ'\} = f'(x), \quad (3)$$
where $f(x)$ is a data with suppressed noise and carried out interpolation (Figure 5, right); $C$ is the forward WPT operator (1). We can also measure data compression ratio ($CR$) as a ratio of non-zero wave-packet coefficients to the total number of discrete samples in input data. The right panels of Figure 5 show reconstructions using compression factors of 50 times ($CR = .02$) respect to the total number of coefficients without losing information about target waves.

6 Discussions

Table 1 demonstrates speed-up of forward and inverse transforms for different platforms. One can see that GPU platform results are much better than results of CPU platform (using the Intel Math Kernel Libraries): about 51 times
Parallel algorithm of 3D wave-packet decomposition of seismic data: Implementation and optimization for GPU

(forward transform) and 41 times (inverse transform) compared to sequential code.

Table 1: Performance for $256^3$ image on different platforms.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Forward (s)</th>
<th>Inverse (s)</th>
<th>Speed-up forward/inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tesla M2010 (GPU)</td>
<td>57.5</td>
<td>76.1</td>
<td>51/41</td>
</tr>
<tr>
<td>Tesla C2010 (GPU)</td>
<td>61.7</td>
<td>85.4</td>
<td>47/37</td>
</tr>
<tr>
<td>GeForce Quadro 4000 (GPU)</td>
<td>80.0</td>
<td>109.1</td>
<td>37/29</td>
</tr>
<tr>
<td>4× AMD Opteron 2218 (MPI)</td>
<td>1113.8</td>
<td>1140.1</td>
<td>3/3</td>
</tr>
</tbody>
</table>

We can work with large data by dividing it into blocks with suitable sizes. These cubes can be processed on different GPUs independently. For the algorithm we have an almost linear scalability. For example, using 2 GPUs we can obtain speed-ups of 90 compared to the performance of sequential code.

7 Conclusions

During this work we have ported sequential code of wavepacket transform to GPU using CUDA technology. We have done algorithmic and program optimizations for effective execution on GPU achieving an average speed-up of about 46 times on a Tesla M2050 compared to the execution time of CPU based sequential code. The obtained accuracy (5 digits) is acceptable for seismic data analysis.

The program implementation of the wave-packet decomposition will be of interest for scientists who are involved in development of seismic exploration methods and for organizations which are involved in seismic data processing.

On the other hand wave-packet decomposition can prove to be useful in other areas of imaging, for instance in medical tomography.

Acknowledgments

We thank A.A. Romanenko for his help. The work was supported by the Swedish Foundation for International Cooperation in Research and Higher
Parallel algorithm of 3D wave-packet decomposition of seismic data: Implementation and optimization for GPU

Education, the Craaford Foundation, the Swedish Research Council, and the Russian Ministry of Education. The first author was also in part supported by National Research University of Information Technologies, Mechanics and Optics within the contract No. 11.G34.31.0019 from 02 December 2010, Integration Project of SB RAS 127.

References


Parallel algorithm of 3D wave-packet decomposition of seismic data: Implementation and optimization for GPU


