DATA PROCESSING ON THE BODY-CENTERED CUBIC LATTICE

by

Usman Raza Alim

M.Sc., Rochester Institute of Technology, 2007

B.Sc., University of Rochester, 2001

A Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2012

APPROVAL

Name: Usman Raza Alim

Degree: Doctor of Philosophy

Title of Thesis: Data Processing on the Body-Centered Cubic Lattice

Examining Committee: Dr. Hao (Richard) Zhang
Chair

Dr. Torsten Möller, Senior Supervisor
Professor of Computing Science

Dr. Steve Ruuth, Supervisor
Professor (Applied and Computational Mathematics)

Dr. Ghassan Hamarneh, Internal Examiner
Associate Professor of Computing Science

Dr. Michael Unser, External Examiner
Professor of Image Processing
École Polytechnique Fédérale de Lausanne (EPFL)
Switzerland

Date Approved: _____

Partial Copyright Licence

Abstract

The body-centered cubic (BCC) lattice is the optimal three-dimensional sampling lattice. Its optimality stems from the fact that its dual, the face-centered cubic (FCC) lattice, achieves the highest sphere-packing efficiency. In order to approximate a scalar-valued function from samples that reside on a BCC lattice, spline-like compact kernels have been recently proposed. The lattice translates of an admissible BCC kernel form a *shift-invariant* approximation space that yields higher quality approximations as compared to similar spline-like spaces associated with the ubiquitous Cartesian cubic (CC) lattice.

In this work, we focus on the approximation of derived quantities from the scalar BCC point samples and investigate two problems: the accurate estimation of the gradient and the approximate solution to Poisson's equation within a rectangular domain with homogeneous Dirichlet boundary conditions. In either case, we seek an approximation in a prescribed shift-invariant space and obtain the necessary coefficients via a discrete convolution operation. Our solution methodology is optimal in an asymptotic sense in that the resulting coefficient sequence respects the asymptotic approximation order provided by the space.

In order to implement the discrete convolution operation on the BCC lattice, we develop efficient three-dimensional versions of the discrete Fourier and sine transforms. These transforms take advantage of the Cartesian coset structure of the BCC lattice in the spatial domain and the geometric properties of the Voronoi tessellation formed by the dual FCC lattice in the Fourier domain.

We validate our solution methodologies by conducting qualitative and quantitative experiments on the CC and BCC lattices using both synthetic and real-world datasets. In the context of volume visualization, our results show that, owing to the superior reconstruction of normals, the BCC lattice leads to a better rendition of surface details. Furthermore, like the approximation of the function itself, this gain in quality comes at no additional cost.

Dedicated to the memory of my grandfather, Dr. Muhammad Alimuddin; and to the future of my son, Zoraiz Z. Alim...

خودی کو کر بلند اِتنَا کہ ہر تقدیر سے پہلے
خدَا بندے سے خود پوچھے بتَا تیری رضَا کیَا ہے
۔ علَّامہ محمد اِقبَال

Endow your will with such power that at every turn of fate,
God Himself asks of His slave, 'What is it that pleases thee?'

— *Allama Muhammad Iqbal*

Preface

We live in a three-dimensional world; many continuous phenomena that we care about naturally have a trivariate description. Of the many possible representations of these phenomena, grid-based representations are specially attractive since they simplify the tasks that practitioners and researchers encounter in the different stages of data flow. Across various scientific disciplines, the term 'grid-based' is usually synonymous with the three-dimensional Cartesian cubic (CC) lattice. The reason for the popularity of this lattice is the fact that it allows one to apply univariate models to the three dimensions separately. Not surprisingly, acquisition devices are fine-tuned to capture data on the CC lattice, and the subsequent processing and analysis tasks are optimized to work with CC data to gain insight into the underlying continuous phenomena of interest. Some examples of disciplines where the CC lattice is widely used include medical imaging (tomography, ultrasound, magnetic resonance imaging), scientific computing (numerical solution of partial differential equations (PDEs)), computer graphics (fluid effects, textures on graphics processing units (GPUs)) and visualization (iso-surface extraction, volume rendering, flow visualization, uncertainty visualization).

Nature is abound with examples of non-Cartesian lattice patterns. The arrangement of cells in a honeycomb, the minimum energy configurations of crystalline structures, and the efficient grocer's packing of oranges are all familiar examples that attempt to increase the isotropy of the arrangement so as to minimize the occurrence of deep holes. The body-centered cubic (BCC) and face-centered cubic (FCC) lattices are of special interest from a sampling theoretic point of view since these lattices — as we shall explore in more detail in Chapter 1 — provide higher fidelity data representations as compared to the CC lattice. Despite this benefit, these non-Cartesian lattices are rarely used in practice. The

problem really is two-fold. On the one hand, there is little to no data that is directly acquired on these lattices. On the other hand, there is a dearth of tools that can efficiently process non-Cartesian data. This thesis attempts to address the latter issue.

The problem of acquiring data on the BCC lattice is, quite arguably, an important research question in its own right. Indeed, our initial foray into the the world of non-Cartesian computing attempted to extend some well-established acquisition algorithms to the BCC lattice. In particular, we extended the mesoscopic Lattice Boltzmann method for simulating incompressible flows [AEM09] and adapted the iterative expectation-maximization algorithm for computed tomography (CT) to the BCC lattice [FAVM09]. In either case, the BCC lattice outperforms its Cartesian counterpart. Fluid simulations carried out on the BCC lattice are more stable and exhibit less numerical dissipation. Likewise, tomographic reconstruction of synthetic data on the BCC lattice leads to lower root mean square (RMS) errors. There are still many open questions and challenges. However, addressing them is deferred to future research.

Scope and Organization

As the title suggests, this thesis deals with the efficient processing of data that is *assumed* to have been acquired on the BCC lattice. For validation purposes, we shall work with BCC datasets that are either synthetically generated or have been adequately subsampled from some high resolution CC version so as to ensure that the resulting data satisfies the assumptions that underly a processing model. In particular, we shall be dealing with *clean* data that is either inherently noise-free or has been adequately post-processed to remove noise. Making our processing pipeline robust in the presence of noise is a subject of future research.

While the problem of reconstructing a continuous function from its scalar BCC samples has sparked quite a bit of research activity, the approximation of derived quantities has received little attention. Thus, bridging this gap is the major theme of this thesis. Our main focus is to seek efficient ways to approximate the gradient of a trivariate scalar function from its point samples (Chapter 2 and Chapter 3). This problem frequently arises in volume visualization where, the gradient provides informative shading cues. Our results clearly demonstrate that a proper treatment of this problem can reveal visual details that may be lost due to a naïve gradient approximation strategy.

We also delve into the problem of solving Poisson's equation within a rectangular domain with homogeneous Dirichlet boundary conditions (Chapter 4). Here, the point samples represent the right hand side of the Poisson equation (see (4.2) on page 66) and the goal is to accurately recover a continuous approximation of the left hand side.

For both of these approximation/reconstruction tasks, the common theme is the idea of approximating a smooth derived function in prescribed *shift-invariant* spaces. These are spaces spanned by a basis obtained via the lattice translates of a generating function. The generating function may or may not be compactly supported. However, most problems of practical interest demand that the generator be compactly supported in order to make the reconstruction tractable. Toward this end, compact spline-like kernels are noteworthy since they are built by piecing together polynomials that can be readily evaluated. We review these concepts in more detail in the introductory chapter (Chapter 1) where, we also establish the mathematical notations and conventions that are used throughout the remainder of the thesis.

Working with shift-invariant spaces has the advantage that the required coefficients (weights) that go with the basis functions can be obtained by convolving the point samples with a discrete filter. On the CC lattice, discrete convolutions are frequently implemented in the Fourier domain via the fast Fourier transform (FFT): a versatile and indispensable processing tool that has been highly optimized. It turns out that the FFT can also be used to efficiently implement discrete convolution operations on the BCC lattice. This topic, as well as the efficient evaluation of the discrete sine transform (DST) on BCC, are discussed in more detail in Chapter 5. Lastly, Chapter 6 discusses related avenues of future research.

Chapter 1 forms the backbone of this thesis and all other chapters depend on it. On the other hand, the topic of gradient estimation (Chapter 2 and Chapter 3), approximating the solution to Poisson's equation (Chapter 4), and discrete trigonometric transforms on BCC (Chapter 5), are presented such that they can be read independently of each other. There is some overlap which is duly pointed out where necessary.

Acknowledgments

A work of this magnitude is rarely a solo effort. I am grateful to many individuals who, in one way or another, helped me bring this work to a fruitful culmination.

I would like to thank:

- Torsten Möller, my senior supervisor, for introducing me to this fascinating research area and for being readily available for his support and guidance through thick and thin.

- The brilliant minds who helped shape my work: Steve Ruuth, Dimitri Van de Ville, Laurent Condat, Thierry Blu, Michael Unser.

- The amazing group of students that I had the pleasure of interacting with at the GrUVi lab: Alireza Entezari, Steven Bergner, Oliver Van Kaick, Ramsay Dyer, Ahmed Saad, Bernard Finkbeiner, Maryam Booshehrian, Zahid Hossain, Tom Torsney-Weir, Hamid Younesy, Alireza Ghane, Mike Phillips, Nima Aghdaii, Ehsan Shokrgozar, Andrea Tagliasacchi.

- My friends at SFU and in the Vancouver area, in particular, Pradeep Ramana, Yonas Weldesselassie, Omer Aziz, Sana Omer, Aamir Malik.

Above all, I am deeply indebted to the support that my family provided during this long journey. In particular, I would like to thank my wife for her support and companionship and my mother for encouraging and believing in me. Last but not least, I would not have been here today without the nurturing that my grandfather provided during the early and formative years of my life.

Contents

List of Tables

List of Figures

Glossary

BCC body-centered cubic.

CC Cartesian cubic.

CT computed tomography.

DFT discrete Fourier transform.

DST discrete sine transform.

DTFT discrete-time Fourier transform.

FCC face-centered cubic.

FFT fast Fourier transform.

FIR finite impulse response.

GPU graphics processing unit.

IIR infinite impulse response.

MDFT multidimensional discrete Fourier transform.

MDST multidimensional discrete sine transform.

ML Marschner and Lobb.

OP orthogonal projection.

PDE partial differential equation.

PSF point spread function.

RMS root mean square.

Notations

$\mathbb{R}^s$	is the s-dimensional Euclidean space.	1
$\mathbb{Z}$	is the set of integers.	2
$\mathbb{Z}^s$	is the Cartesian power of the set of integers, $\mathbb{Z}^s := \underbrace{\mathbb{Z} \times \mathbb{Z} \times \cdots \times \mathbb{Z}}_{s}$.	2
$\hat{f}(\boldsymbol{\omega})$	denotes the Fourier transform of the function $f(\boldsymbol{x})$.	2
$\leftrightarrow$	indicates continuous or discrete-time Fourier transform pairs.	8
$\mathcal{L}$	denotes a lattice generated by the matrix $\mathbf{L}$.	2
$\mathcal{L}^\circ$	denotes the lattice that is dual to $\mathcal{L}$ and is generated by the matrix $\mathbf{L}^{-\mathsf{T}}$.	3
$L_2(\mathbb{R}^s)$	is the space of square integrable functions in $\mathbb{R}^s$.	4
$g^\star(\boldsymbol{x})$	is the complex conjugate of $g(\boldsymbol{x})$.	4
$\|f\|$	denotes the L_2-norm of the function f.	4
$\|c\|_{l_2(\mathbb{Z}^s)}$	denotes the l_2-norm of the sequence c.	5
$\langle f, g \rangle$	is the L_2-inner product between f and g.	4
$O(x^k)$	Big O notation for asymptotics. $f(x) = O(x^k)$ if and only if $\limsup_{x\to 0} \left\| \frac{f(x)}{x^k} \right\| < \infty$.	5
$\mathbb{V}(\mathcal{L}_h, \varphi)$	is the approximation space spanned by the lattice shifts of the kernel φ scaled by h.	4
$D^{\boldsymbol{\alpha}}$	denotes the operator that gives the partial derivative of order $\left\|\boldsymbol{\alpha}\right\|$, where $\boldsymbol{\alpha}$ is a multi-index.	5
$W_2^k(\mathbb{R}^s)$	denotes the Sobolev space formed by those functions in $L_2(\mathbb{R}^s)$ whose weak derivates upto order k are also in $L_2(\mathbb{R}^s)$.	5
$\Pi_k(\mathbb{R}^s)$	is the vector space of all multivariate polynomials in $\mathbb{R}^s$ of degree at most k.	5

$\big(f * g\big)(\boldsymbol{x})$	is the continuous convolution of the functions f and g. It is defined as $\big(f * g\big)(\boldsymbol{x}) := \int_{\mathbb{R}^s} f(\boldsymbol{y}) g(\boldsymbol{x} - \boldsymbol{y}) d\boldsymbol{y}$.	8
$\big(f * g\big)[\boldsymbol{n}]$	is the discrete convolution of the sequences f and g. It is defined as $\big(f * g\big)[\boldsymbol{n}] := \sum_{\boldsymbol{l} \in \mathbb{Z}^s} f[\boldsymbol{l}] g[\boldsymbol{n} - \boldsymbol{l}]$.	12
$\overline{\varphi}(\boldsymbol{x})$	is the flipped version of φ, $\overline{\varphi}(\boldsymbol{x}) := \varphi(-\boldsymbol{x})$.	8
$a_{\varphi}[\boldsymbol{n}]$	is the autocorrelation sequence of the kernel φ with respect to a lattice $\mathcal{L}$.	8
$\delta_{\boldsymbol{n}-\boldsymbol{l}}$	is Kronecker's delta function. $\delta_{\boldsymbol{n}-\boldsymbol{l}} = 1$ if $\boldsymbol{n} = \boldsymbol{l}$ and 0 otherwise.	9
$\mathring{\varphi}$	is the biorthogonal dual of the kernel φ. $\hat{\mathring{\varphi}}(\boldsymbol{\omega}) = \frac{\hat{\varphi}(\boldsymbol{\omega})}{\hat{A}_{\varphi}(\boldsymbol{\omega})}$.	9
$\big(\mathsf{P}_{\mathbb{V}(\mathcal{L}_h, \varphi)} f\big)$	denotes the orthogonal projection of f onto the space $\mathbb{V}(\mathcal{L}_h, \varphi)$.	9
$\mathbb{1}_{\mathcal{P}}(\boldsymbol{x})$	is the indicator function of the set $\mathcal{P}$, i.e. $\mathbb{1}_{\mathcal{P}}(\boldsymbol{x}) = 1$ when $\boldsymbol{x} \in \mathcal{P}$ and 0 otherwise.	11
$M_{\boldsymbol{\Xi}}(\boldsymbol{x})$	denotes the box spline formed by the direction vectors in the $s \times n$ matrix $\boldsymbol{\Xi}$.	19
$\beta^k(x)$	denotes the centered B-spline of degree k.	19
$B^k(\boldsymbol{x})$	is the tensor product of 1D centered B-splines of degree k.	19
$\vartheta^k(\boldsymbol{x})$	is the rhombic dodecahedral box spline that provides an approximation order of k on the BCC lattice.	35
$\boldsymbol{\nabla} f$	is the gradient of the function f.	31
$\partial_i f$	denotes the partial derivative with respect to x_i, i.e. $\partial_i f := \frac{\partial f}{\partial x_i}$.	32
$\partial_{\vec{\boldsymbol{u}}} f$	denotes the directional derivative of f in the direction $\vec{\boldsymbol{u}}$, i.e. $\partial_{\vec{\boldsymbol{u}}} f := \boldsymbol{\nabla} f \cdot \vec{\boldsymbol{u}}$.	48
	The following additional symbols are first defined in Chapter 4.	65
$\big\langle f_1, f_2 \big\rangle_{\mathcal{P}^s}$	is the L_2-inner product between f_1 and f_2 over the domain $\mathcal{P}^s := [-1, 1]^s$.	66
ΔV	is the Laplacian of V where $\Delta := \sum_{i=1}^{s} \frac{\partial^2}{\partial x_i^2}$.	66
$\tilde{\rho}[\boldsymbol{m}]$	for $\boldsymbol{m} \in \mathbb{Z}_+^s$, denotes the Fourier sine series coefficients of a rectangularly periodic and odd function ρ.	67
$\hat{\rho}[\boldsymbol{m}]$	for $\boldsymbol{m} \in \mathbb{Z}^s$, denotes the Fourier series coefficients of a rectangularly periodic function ρ.	67
$\Leftrightarrow$	denotes Fourier sine series pairs, i.e. $\rho(\boldsymbol{x}) \Leftrightarrow \tilde{\rho}[\boldsymbol{m}]$.	68
$\circledast$	denotes the periodic convolution operation.	73

Chapter 1

Background

The numerical approximation of continuous real-world phenomena is a fundamental task that is at the heart of many scientific disciplines. Given a function $f(\boldsymbol{x})$ ($\boldsymbol{x} \in \mathbb{R}^s$) that models some real-world phenomenon, we often wish to find an approximation $f_{\mathsf{app}}(\boldsymbol{x})$ that, in some quantifiable way, can be brought arbitrarily close to f so that it gives us a faithful representation of the underlying continuous phenomenon. Typical examples include speech or audio signals ($s = 1$), a photograph or image ($s = 2$), or medical scans such as magnetic resonance imaging or computed tomography ($s = 3$). In this chapter, we are concerned with approximations that are obtained from a set of discrete measurements of f so that they can be efficiently manipulated and processed by a digital computer. We are specially interested in discrete measurements that lie on a point lattice. This not only facilitates mathematical analysis but also expedites many acquisition and processing tasks. Of notable importance to us is the three-dimensional case ($s = 3$) where, we shall argue that the BCC lattice is a much better alternative to the ubiquitous CC lattice for the representation of 3D scalar functions.

1.1 Approximation in Shift Invariant Spaces

We first look at the approximation problem in the context of bandlimited functions and later explore a more general Hilbert-space formulation that encapsulates the bandlimited function model.

1.1.1 Reconstruction of Bandlimited Functions

One of the seminal results is attributed to Shannon, Nyquist and Whittaker, commonly referred to as the Nyquist-Shannon sampling theorem [Sha49]. In 1D, this theorem tells us that exact recovery of f is possible as long as f is bandlimited and is point sampled at a rate strictly greater than twice the Nyquist rate. Recall that the Fourier transform $\hat{f}(\boldsymbol{\omega})$ ($\boldsymbol{\omega} \in \mathbb{R}^s$) of an s-dimensional function f is defined as

$$\hat{f}(\boldsymbol{\omega}) := \int_{\mathbb{R}^s} f(\boldsymbol{x}) \exp(-2\pi\imath\boldsymbol{\omega}^\mathsf{T}\boldsymbol{x}) d\boldsymbol{x}. \tag{1.1}$$

When $s = 1$, a spectrum $\hat{f}(\omega)$ is bandlimited if it is completely supported inside the interval $[-\omega_0, \omega_0]$. If this is the case, then f can be completely recovered from regularly spaced samples $f(hk)$ ($k \in \mathbb{Z}$) according to the interpolation formula

$$f(x) = \sum_{k\in\mathbb{Z}} f(hk)\,\mathrm{sinc}(\frac{x}{h} - k), \tag{1.2}$$

provided that the sampling rate $h < \frac{1}{2\omega_0}$. We shall not give a proof of Shannon's theorem here. However, we would like to mention that a key ingredient is Poisson's summation formula which tells us that sampling in the spatial domain is tantamount to a periodization of the spectrum in the Fourier domain [Pin02]. The higher the sampling rate, the further apart the Fourier spectrum replica will be. Thus, at the Nyquist rate, the replica have no overlap (no aliasing) which makes exact recovery possible. The Fourier transform of the interpolator $\mathrm{sinc}(x) := \frac{\sin(\pi x)}{\pi x}$ is a rectangular pulse that is unity when $\omega \in (-\frac{1}{2}, \frac{1}{2})$ and zero otherwise. Thus, the recovery formula (1.2) can be seen as carving out the spectrum $\hat{f}(\omega)$ by suppressing all the replica.

Shannon's sampling theorem can be extended to higher dimensions via a simple tensor product provided that the relevant function of interest $f(\boldsymbol{x})$ has a spectrum $\hat{f}(\boldsymbol{\omega})$ that is completely supported inside a rectangular region in the Fourier domain. This is equivalent to using the s-dimensional integer lattice $\mathbb{Z}^s$ as the sampling pattern in the spatial domain. A more general version of the theorem that handles non-rectangular sampling lattices was first proposed by Miyakawa [Miy59] (in Japanese) and later brought to the attention of the Anglosphere by Peterson and Middleton [PM62]. Recall that a sampling lattice $\mathcal{L}$ is a set of points in $\mathbb{R}^s$ that forms a group under addition. It is generated by a matrix $\mathbf{L}$ and is defined as

$$\mathcal{L} := \{\mathbf{L}\boldsymbol{n} : \boldsymbol{n} \in \mathbb{Z}^s\}. \tag{1.3}$$

The matrix $\mathbf{L}$ is usually an $s \times s$ non-singular matrix and is not unique. Sampling a function f on $\mathcal{L}$ amounts to a periodization of $\hat{f}$ on the dual lattice $\mathcal{L}^\circ$ that is generated by $\mathbf{L}^{-\mathsf{T}}$. A multi-dimensional version of the Poisson summation formula [DM84, Sec. 1.4.2] makes this operation precise.

$$\sum_{\boldsymbol{n}\in\mathbb{Z}^s} f(\mathbf{L}\boldsymbol{n}) \exp(-2\pi\imath\boldsymbol{\omega}^\mathsf{T}\mathbf{L}\boldsymbol{n}) = \frac{1}{\left|\det(\mathbf{L})\right|} \sum_{\boldsymbol{m}\in\mathbb{Z}^s} \hat{f}(\boldsymbol{\omega} - \mathbf{L}^{-\mathsf{T}}\boldsymbol{m}) \tag{1.4}$$

This formula tells us that, if the spectrum $\hat{f}$ is compactly supported and can be tightly enclosed inside the Voronoi cell around $\boldsymbol{\omega} = \mathbf{0}$ of the dual lattice $\mathcal{L}^\circ$, then f can be completely recovered from its spatial lattice samples $f(\mathbf{L}\boldsymbol{n})$. The multi-dimensional analog of the interpolation formula (1.2) is

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{n}\in\mathbb{Z}^s} f(\mathbf{L}\boldsymbol{n}) \operatorname{sinc}_{\mathcal{L}}(\boldsymbol{x} - \mathbf{L}\boldsymbol{n}), \tag{1.5}$$

where $\operatorname{sinc}_{\mathcal{L}}(\boldsymbol{x})$ is the interpolator whose Fourier transform is the indicator function of the Voronoi cell of $\mathcal{L}^\circ$ around $\boldsymbol{\omega} = \mathbf{0}$.

If the compact support of $\hat{f}$ has a non-rectangular shape, then the most efficient sampling lattice $\mathcal{L}$ is the one whose dual $\mathcal{L}^\circ$ most tightly packs the spectrum replica of $\hat{f}$ [PM62]. In many practical applications, one typically assumes that there is no directional bias and the spectrum $\hat{f}$ is radially bandlimited, i.e. $\hat{f}(\boldsymbol{\omega}) = 0$ whenever $\boldsymbol{\omega} > \omega_0$. For such a function, the most efficient sampling lattice is therefore the dual of the densest hypersphere packing lattice in $\mathbb{R}^s$. Petersen and Middleton [PM62] provided the generating matrices for optimal sampling lattices up to $s = 7$. In particular, for $s = 2$, the densest circle packing lattice is the hexagonal lattice. Its dual, a rotated hexagonal lattice is therefore the optimal sampling lattice. For $s = 3$, the densest sphere packing lattice is the FCC lattice and its dual, the BCC lattice provides the most optimal sampling. A complete characterization of the known hypersphere packing lattices is provided by Conway and Sloan [CS99]. For the more general case of arbitrary polygonal shape Fourier spectra, Lu *et al.* have recently proposed a search algorithm that can find the most efficient alias-free sampling lattices [LDL09].

1.1.2 Hilbert-space Formulation

Bandlimited recovery formulas such as (1.2) and (1.5) are rarely used in practice for a number of reasons. Firstly, functions encountered in practice are seldom bandlimited and therefore need to be appropriately filtered with an anti-aliasing prefilter in order to ensure

that the interpolation operation recovers the closest bandlimited approximation. Secondly, the painstakingly slow decay of the sinc function prohibits practical use. It is much more desirable to use reconstruction kernels that are compactly supported so that an approximation can be readily computed. The ubiquity of linear interpolation is indisputable. It is efficient and can easily provide better quality approximations simply by increasing the sampling rate. It is therefore advantageous to adopt a more general Hilbert space formulation that in principle behaves like the interpolation formula (1.5) but allows for the inclusion of more compact kernels.

We assume that our function of interest f lies in $L_2(\mathbb{R}^s)$, the space of square integrable functions in $\mathbb{R}^s$. We denote the L_2-norm of f as

$$\|f\| := \left(\int_{\mathbb{R}^s} \left| f(\boldsymbol{x}) \right|^2 d\boldsymbol{x} \right)^{\frac{1}{2}} < \infty. \tag{1.6}$$

$L_2(\mathbb{R}^s)$ is a Hilbert space with the associated inner product given by

$$\left\langle f, g \right\rangle := \int_{\mathbb{R}^s} f(\boldsymbol{x}) g^\star(\boldsymbol{x}) d\boldsymbol{x}, \tag{1.7}$$

where $g^\star(\boldsymbol{x})$ is the complex conjugate of $g(\boldsymbol{x})$. With this definition of the inner product, a function f and its Fourier transform $\hat{f}$ are isometric i.e., $\|f\|^2 = \left\langle f, f \right\rangle = \left\langle \hat{f}, \hat{f} \right\rangle = \|\hat{f}\|^2$.

We generalize the bandlimited formulation described earlier to shift-invariant spaces, spaces that are spanned by the lattice shifts of a kernel $\varphi(\boldsymbol{x})$. Unless otherwise stated, we assume that $\mathcal{L}$ is normalized so that $\left|\det(\mathbf{L})\right| = 1$. In order to control the rate of sampling, we employ an isotropic sampling parameter h to refine the lattice. We denote the resulting scaled lattice as $\mathcal{L}_h := \{h\mathbf{L}\boldsymbol{n} : \boldsymbol{n} \in \mathbb{Z}^s\}$. A shift-invariant approximation space is then defined as

$$\mathbb{V}(\mathcal{L}_h, \varphi) := \left\{ g(\boldsymbol{x}) = \sum_{\boldsymbol{n} \in \mathbb{Z}^s} c[\boldsymbol{n}] \varphi(\frac{\boldsymbol{x}}{h} - \mathbf{L}\boldsymbol{n}) : c \in l_2(\mathbb{Z}^s) \right\}. \tag{1.8}$$

In order to ensure that any function $g \in \mathbb{V}(\mathcal{L}_h, \varphi)$ has a unique representation in terms of a finite energy coefficient sequence $c[\cdot]$, the set of functions $\{\varphi_{h,\boldsymbol{n}}\}_{\boldsymbol{n} \in \mathbb{Z}^s}$ where $\varphi_{h,\boldsymbol{n}}(\boldsymbol{x}) := \varphi(\frac{\boldsymbol{x}}{h} - \mathbf{L}\boldsymbol{n})$, must be linearly independent and form a basis for $\mathbb{V}(\mathcal{L}_h, \varphi)$. An important concept is that of a *Riesz* basis, which is a basis that is not necessarily orthogonal but exhibits properties that are similar to those found in orthogonal bases [Kre89]. Formally, the lattice translates $\{\varphi_{h,\boldsymbol{n}}\}_{\boldsymbol{n} \in \mathbb{Z}^s}$ form a Riesz basis if there exist finite positive constants A and B such that

$$\forall\, c[\boldsymbol{n}] \in l_2(\mathbb{Z}^s),\ A\|c\|^2_{l_2(\mathbb{Z}^s)} \leq \| \sum_{\boldsymbol{n} \in \mathbb{Z}^s} c[\boldsymbol{n}] \varphi_{h,\boldsymbol{n}}(\boldsymbol{x}) \|^2 \leq B\|c\|^2_{l_2(\mathbb{Z}^s)}, \tag{1.9}$$

where $\|c\|_{l_2(\mathbb{Z}^s)} := \left(\sum_{\boldsymbol{n}\in\mathbb{Z}^s} \left|c[\boldsymbol{n}]\right|^2\right)^{\frac{1}{2}}$ is the l_2-norm of the coefficient sequence c. This is an important requirement when seeking an approximation $f_{\mathsf{app}} \in \mathbb{V}(\mathcal{L}_h, \varphi)$. Besides ensuring that $\mathbb{V}(\mathcal{L}_h, \varphi) \subset L_2(\mathbb{R}^s)$, it guarantees that a biorthogonal basis of $\mathbb{V}(\mathcal{L}_h, \varphi)$ exists so that f can be orthogonally projected to $\mathbb{V}(\mathcal{L}_h, \varphi)$. The basis is orthonormal when $A = B = 1$ and there is norm equivalence between the L_2-norm of a function $g \in \mathbb{V}(\mathcal{L}_h, \varphi)$ and the l_2-norm of its expansion coefficients. Indeed, the interpolator $\mathrm{sinc}_{\mathcal{L}}$ introduced in the previous section forms an orthonormal basis for the space of bandlimited functions, the sample values serving as the expansion coefficients.

1.1.3 Approximation Order and Strang-Fix Conditions

The bandlimited model introduced earlier aims at perfectly recovering a bandlimited function from its sample values. The shift invariant model on the other hand relaxes this requirement and provides a way to quantify how far off an approximation f_{app} is in terms of the L_2-distance $\|f - f_{\mathsf{app}}\|$. For an integer k, we say that the space $\mathbb{V}(\mathcal{L}_h, \varphi)$ provides an *approximation order* of k if for each sufficiently smooth function f, there exists a constant $C > 0$ such that

$$\|f - f_{\mathsf{app}}\| \leq Ch^k \;\text{ as }\; h \to 0, \;\text{ or equivalently }\; \|f - f_{\mathsf{app}}\| = O(h^k). \tag{1.10}$$

The smoothness requirement on f is made precise by requiring that f belongs to the Sobolev space $W_2^k(\mathbb{R}^s)$, the space of those functions in $L_2(\mathbb{R}^s)$ whose *weak* partial derivatives up to order k are also in $L_2(\mathbb{R}^s)$. It is defined as

$$W_2^k(\mathbb{R}^s) := \left\{u \in L_2(\mathbb{R}^s) : D^{\boldsymbol{\alpha}} u \in L_2(\mathbb{R}^s) \;\; \forall \left|\boldsymbol{\alpha}\right| \leq k\right\}, \tag{1.11}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_s)$ is a multi-index, $\left|\boldsymbol{\alpha}\right|$ denotes its l_1-norm, and $D^{\boldsymbol{\alpha}} u := \frac{\partial^{\left|\boldsymbol{\alpha}\right|} u}{\partial x_1^{\alpha_1} \cdots \partial x_s^{\alpha_s}}$ is a partial derivative of order $\left|\boldsymbol{\alpha}\right|$ in the weak sense. Equivalently, $f \in W_2^k(\mathbb{R}^s)$ if and only if $\int_{\mathbb{R}^s} \left|\omega_1^{\alpha_1} \cdots \omega_s^{\alpha_s} \hat{f}(\boldsymbol{\omega})\right|^2 d\boldsymbol{\omega} < \infty$, $\forall \left|\boldsymbol{\alpha}\right| \leq k$. Thus, the Fourier transform $\hat{f}(\boldsymbol{\omega})$ must decay faster than any polynomial up to degree k, i.e. $\lim_{\|\boldsymbol{\omega}\|\to\infty} \left|\omega_1^{\alpha_1} \cdots \omega_s^{\alpha_s} \hat{f}(\boldsymbol{\omega})\right| = 0$, $\forall \left|\boldsymbol{\alpha}\right| \leq k$. Note that when f is differentiable in the conventional sense, its weak derivative coincides with the strong derivative $D^{\boldsymbol{\alpha}} f$.

There is an important connection between the approximation order k and $\Pi_{k-1}(\mathbb{R}^s)$, the vector space of all multivariate polynomials of degree at most $k-1$. In the univariate setting $s = 1$, this connection was first established by Schoenberg [Sch46] who is regarded

as the father of splines. It was later formalized by Strang and Fix [SF71] in the context of the univariate finite element method. It is now commonly referred to as the *Strang-Fix* conditions.

Strang-Fix Conditions of Order k: *In the multivariate setting of approximation in $\mathbb{V}(\mathcal{L}_h, \varphi)$, the following statements are equivalent.*

$$\hat{\varphi}(\mathbf{0}) = 1, \quad \textit{and} \quad D^{\boldsymbol{\alpha}}\hat{\varphi}(\boldsymbol{\beta}) = 0 \ \textit{for} \ \left|\boldsymbol{\alpha}\right| \leq k-1 \ \textit{and} \ \forall \boldsymbol{\beta} \in \mathcal{L}^{\circ} \backslash \{\mathbf{0}\}. \tag{1.12a}$$

$$\mathbb{V}(\mathcal{L}, \varphi) \ \textit{contains} \ \Pi_{k-1}(\mathbb{R}^s). \tag{1.12b}$$

A proof of (1.12b) $\implies$ (1.12a) is included in the chapter notes.

If, in addition to satisfying the Riesz basis condition introduced earlier, the kernel φ also satisfies the Strang-Fix conditions of order k, then for any $f \in W_2^k(\mathbb{R}^s)$, there exists a minimum error approximation $f_{\mathsf{app}} \in \mathbb{V}(\mathcal{L}_h, \varphi)$ such that $\|f - f_{\mathsf{app}}\| = O(h^k)$. We do not give a proof of this here and refer the reader to recent surveys on approximation in shift-invariant spaces such as Jia [Jia95] or Jetter and Plonka [JP01]. Another important survey ($s = 1$) geared towards the signal processing audience is due to Unser [Uns00].

The Strang-Fix conditions tell us that the space $\mathbb{V}(\mathcal{L}, \varphi)$ can reproduce any polynomial of degree at most $k-1$. According to the Weirstrass theorem, polynomials are dense in the space of continuous functions. Thus, any function that is sufficiently smooth can be approximated in $\mathbb{V}(\mathcal{L}_h, \varphi)$ to an arbitrary degree of accuracy, the rate of decay of the L_2-error being governed by the Strang-Fix conditions. The minimum requirement for a convergent approximation scheme is imposed by the Strang-Fix conditions of order 1, often termed as the *partition of unity* which states that the lattice translates of φ must reproduce unity, i.e.

$$\sum_{\boldsymbol{n} \in \mathbb{Z}^s} \varphi(\boldsymbol{x} - \mathbf{L}\boldsymbol{n}) = 1, \quad \text{or equivalently,} \quad \hat{\varphi}(\mathbf{0}) = 1 \text{ and } \hat{\varphi}(\boldsymbol{\beta}) = 0 \ \ \forall \boldsymbol{\beta} \in \mathcal{L}^{\circ} \backslash \{\mathbf{0}\}. \tag{1.13}$$

This condition can also be directly derived in the spatial domain by considering the Taylor expansion of the reconstruction sum [MMMY97b].

The Strang-Fix conditions also play an important role in wavelet analysis where, in addition to satisfying these conditions, an admissible scaling function must also satisfy a self-similarity relationship that ensures that a nested hierarchy of approximation spaces can be constructed. These self-similar scaling functions are often constructed using a subdivision scheme that starts with a compactly supported *mask* defined on $\mathcal{L}$. An iterative

procedure is then employed to refine the mask until it converges to the admissible kernel φ (see e.g. Jia [Jia98] and references therein). In this work, we do not focus on wavelets or multiresolution analysis and therefore, do not require that φ satisfies any self-similarity relations.

1.1.4 Goals

There are many admissible kernels that are widely used in practice, the most familiar being the B-splines [deB01] ($s = 1$) and their tensor-product extensions ($s > 1$). In fact, the use of splines is so common that shift-invariant spaces are also frequently known as *spline-like* spaces. Most effort on multivariate spline-like approximation spaces has focused on the integer lattice $\mathbb{Z}^s$ where a 1D kernel can be employed via a simple tensor-product extension. Non-separable kernels have also been proposed, the most notable example being the *box splines* [dHR93].

Recently, the BCC lattice, owing to its sampling optimality, has gained attention specially in the fields of computer graphics and visualization. Justification for using the BCC sampling lattice usually follows the bandlimited model which argues that a BCC lattice can capture the same isotropically bandlimited spectrum as the CC lattice, with 30% fewer samples [TMG01]. However, this is only true if the appropriate $\mathrm{sinc}_{\mathcal{L}}$ function is used as the reconstruction kernel. Most practical approximation and interpolation schemes on the BCC lattice however, use more localized kernels such as box splines. How relevant is the bandlimited argument in this context? What can we say about the L_2-error of such schemes as compared to similar schemes used on the CC lattice? In this chapter, we review some of the recently proposed BCC reconstruction kernels and compare them with their CC counterparts in light of the theoretical analysis tools introduced earlier. In Section 1.2, we introduce some quantitative analysis tools that can be used to compare the approximation quality of interpolative and quasi-interpolative schemes while Section 1.3 compares such schemes on the CC and BCC lattices.

1.2 Quantitative Error Analysis

We have seen that the Strang-Fix conditions characterize the approximation characteristics of the orthogonal projection of f onto the desired space $\mathbb{V}(\mathcal{L}_h, \varphi)$. Such an approximation scheme requires that the function measurements be made with respect to a dual

basis (Section 1.2.1). In many practical problems of interest, one is often not able to modify the measurement procedure. The measurements are made using some sort of a point spread function (PSF) that cannot be altered. Typically, the PSF is assumed to be Dirac's delta distribution $\delta(\boldsymbol{x})$ and what we get as a starting point is a sequence of sample values $f(h\boldsymbol{L}\boldsymbol{n})$. The subsequent approximation schemes are sub-optimal and employ either interpolative or quasi-interpolative models (Section 1.2.3). How does the L_2-error behave for such schemes as compared to the error predicted by the Strang-Fix conditions? We take a closer look at this question in Section 1.2.4. However, before proceeding further, we need to introduce some additional notations and definitions.

The discrete-time Fourier transform (DTFT) [DM84] of a sequence $c[\cdot]$ associated with $\mathcal{L}$ is defined as

$$\hat{C}(\boldsymbol{\omega}) := \sum_{\boldsymbol{n}\in\mathbb{Z}^s} c[\boldsymbol{n}] \exp(-2\pi\imath\boldsymbol{\omega}^{\mathsf{T}}\mathbf{L}\boldsymbol{n}). \tag{1.14}$$

It is periodic with respect to the dual lattice $\mathcal{L}^\circ$, i.e. $\hat{C}(\boldsymbol{\omega}+\mathbf{L}^{-\mathsf{T}}\boldsymbol{m}) = \hat{C}(\boldsymbol{\omega})$, for $\boldsymbol{m}\in\mathbb{Z}^s$. In order to distinguish between the DTFT and the Fourier transform, we shall use a lower-case symbol to denote a discrete sequence in the spatial domain, and the corresponding upper-case symbol to denote its DTFT. We shall also make use of the symbol $\leftrightarrow$ interchangeably to indicate transform pairs. Thus, for a function f, we have the Fourier transform pairs $f(\boldsymbol{x}) \leftrightarrow \hat{f}(\boldsymbol{\omega})$, and for a discrete sequence c, we have the DTFT pairs $c[\boldsymbol{n}] \leftrightarrow \hat{C}(\boldsymbol{\omega})$. Usually, when there is no room for ambiguity, we shall also omit the parentheses $(\cdot)$ and the square brackets $[\cdot]$.

The autocorrelation sequence of φ with respect to the lattice $\mathcal{L}$ is defined as

$$a_\varphi[\boldsymbol{n}] := \langle \varphi, \varphi_{\boldsymbol{n}} \rangle = \big(\varphi * \overline{\varphi}\big)(\mathbf{L}\boldsymbol{n}), \tag{1.15}$$

where $*$ represents the continuous convolution operation and $\overline{\varphi}(\boldsymbol{x}) := \varphi(-\boldsymbol{x})$. The Riesz basis condition (1.9) has a simple characterization in the Fourier domain in terms of $\hat{A}_\varphi$ [AU94] given by

$$A \le \hat{A}_\varphi(\boldsymbol{\omega}) \le B. \tag{1.16}$$

1.2.1 Minimum Error Approximation

If the lattice translates $\{\varphi_{h,\boldsymbol{n}}\}_{\boldsymbol{n}\in\mathbb{Z}^s}$ form a Riesz basis, then there exists a dual kernel $\mathring{\varphi} \in \mathbb{V}(\mathcal{L},\varphi)$ such that the shifts $\{\frac{1}{h^s}\mathring{\varphi}_{h,\boldsymbol{n}}\}_{\boldsymbol{n}\in\mathbb{Z}^s}$ form a basis that is biorthonormal. In other words,

$\left\langle \varphi_{h,\boldsymbol{n}}, \frac{1}{h^s}\mathring{\varphi}_{h,\boldsymbol{l}} \right\rangle = \delta_{\boldsymbol{n}-\boldsymbol{l}}$, where $\delta_{\boldsymbol{n}-\boldsymbol{l}}$ is Kronecker's delta function. Using this biorthonormality relationship and the fact that $\mathring{\varphi} \in \mathbb{V}(\mathcal{L}, \varphi)$, it can be shown that

$$\mathring{\varphi}(\boldsymbol{x}) = \sum_{\boldsymbol{n}\in\mathbb{Z}^s} a_{\varphi}^{-1}[\boldsymbol{n}]\varphi(\boldsymbol{x} - \mathbf{L}\boldsymbol{n}), \tag{1.17}$$

where the sequence a_{φ}^{-1} is related to the autocorrelation sequence according to $a_{\varphi}^{-1}[\boldsymbol{n}] \leftrightarrow 1/\hat{A}_{\varphi}(\boldsymbol{\omega})$ [Uns00]. From this, we can easily infer that $\mathring{\varphi}(\boldsymbol{x}) \leftrightarrow \frac{\hat{\varphi}(\boldsymbol{\omega})}{\hat{A}_{\varphi}(\boldsymbol{\omega})}$.

Orthogonal Projection: The orthogonal projection of f onto $\mathbb{V}(\mathcal{L}_h, \varphi)$ is obtained by taking inner products with the duals $\mathring{\varphi}_{h,\boldsymbol{n}}$ [Uns00]. In particular, the orthogonal projection is given by

$$\left(\mathsf{P}_{\mathbb{V}(\mathcal{L}_h,\varphi)} f\right)(\boldsymbol{x}) := \sum_{\boldsymbol{n}} c[\boldsymbol{n}]\varphi(\frac{\boldsymbol{x}}{h} - \mathbf{L}\boldsymbol{n}), \quad \text{where} \quad c[\boldsymbol{n}] = \left\langle f, \frac{1}{h^s}\mathring{\varphi}_{h,\boldsymbol{n}} \right\rangle. \tag{1.18}$$

According to the Hilbert projection theorem [Kre89], an approximation $f_{\mathsf{app}} \in \mathbb{V}(\mathcal{L}_h, \varphi)$ of f minimizes the L_2-error $\|f - f_{\mathsf{app}}\|$ if and only if $f_{\mathsf{app}} = \mathsf{P}_{\mathbb{V}(\mathcal{L}_h,\varphi)} f$.

If the function f is sufficiently smooth, i.e. $f \in W_2^k(\mathbb{R}^s)$, and φ is compactly supported and satisfies the Strang-Fix conditions of order k, then $\|f - \left(\mathsf{P}_{\mathbb{V}(\mathcal{L}_h,\varphi)} f\right)\| = O(h^k)$ as $h \to 0$ [JP01, Theorem 2.3.12]. For a convergence rate that is $O(h^k)$, the Strang-Fix conditions are equivalent to saying that φ has vanishing moments up to order $k-1$ [Uns96]. This suggests that the asymptotic error is dominated by the k-th order moments. For $s = 1$, a tight bound can be obtained in terms of the derivatives of $\hat{\varphi}$. In particular, we have the following error bound.

$$\|f - \mathsf{P}_{\mathbb{V}(\mathbb{Z}_h,\varphi)} f\| \leq C_{\varphi} \|D^k f\| h^k \;\text{ as }\; h \to 0, \tag{1.19}$$

where the constant C_{φ} is given by

$$C_{\varphi} = \frac{1}{k!}\sqrt{\sum_{n\neq 0} \left|D^k \hat{\varphi}(n)\right|^2} \tag{1.20}$$

[Uns96, Eq. 17]. For $s > 1$, the convergence is determined by a combination of the various k-th order moments. We shall revisit this issue in Section 1.2.4.

1.2.2 Approximation of Bandlimited Functions

It is also useful to revisit the problem of bandlimited recovery from the perspective of the space $\mathbb{V}(\mathcal{L}_h, \mathrm{sinc}_{\mathcal{L}})$. It is easily shown that $\hat{A}_{\mathrm{sinc}_{\mathcal{L}}}(\boldsymbol{\omega}) = 1$, which is what we would expect

since the shifts $\{\mathrm{sinc}_{\mathcal{L}}(\cdot - \mathbf{L}\boldsymbol{n})\}_{\boldsymbol{n}\in\mathbb{Z}^s}$ form an orthonormal basis. If f is bandlimited, then there exists a scale h_0 such that $\left(\mathsf{P}_{\mathbb{V}(\mathcal{L}_h,\mathrm{sinc}_{\mathcal{L}})}f\right) = f$ whenever $h < h_0$, and we have perfect reconstruction. Moreover, when $h < h_0$, the inner products simply yield the sample values i.e., $\left\langle f, \frac{1}{h^s}\mathrm{sinc}_{\mathcal{L}}(\frac{\cdot}{h} - \mathbf{L}\boldsymbol{n})\right\rangle = f(h\mathbf{L}\boldsymbol{n})$, and we can safely use Dirac's delta $\delta(\boldsymbol{x})$ instead of the dual $\frac{1}{h^s}\mathrm{sinc}_{\mathcal{L}}(\frac{\boldsymbol{x}}{h})$ as a measurement device. On the other hand, if f is not bandlimited, then the same inner products serve the role of an ideal low pass prefilter that is applied to f and the resulting approximation $\left(\mathsf{P}_{\mathbb{V}(\mathcal{L}_h,\mathrm{sinc}_{\mathcal{L}})}f\right)$ is the closest bandlimited approximation of f in $\mathbb{V}(\mathcal{L}_h, \mathrm{sinc}_{\mathcal{L}})$, the error of which behaves as $O(h^k)$ where k is the maximum Sobolev regularity exponent of f [JP01, Lemma 2.3.2].

Another interesting case that has received a great deal of attention in signal processing and computer graphics is the use of a compact generator φ to approximate a bandlimited function f in the shift-invariant space $\mathbb{V}(\mathcal{L}_h, \varphi)$. Traditionally, this problem has been investigated in the univariate setting where the errors are classified as either *prealiasing* or *postaliasing* artifacts [ML94, MN88, Key81]. Prealiasing occurs when either the sampling rate is not sufficient or when the function of interest is not appropriately prefiltered. On the other hand, postaliasing refers to the fact that the Fourier transform of a compact reconstruction kernel $\hat{\varphi}$ has a slow rate of decay and picks up aliases of the spectrum in addition to the central copy.

In the multivariate shift-invariant setting, we can quantitatively analyze these phenomena in terms of a frequency error kernel that yields the L_2-error $\|f - \mathsf{P}_{\mathbb{V}(\mathcal{L}_h,\varphi)}f\|$. Without loss of generality, let us assume that $h = 1$ and that f is bandlimited such that $\hat{f}$ is completely supported inside $\mathcal{V}_{\mathcal{L}^\circ}$, the Voronoi cell of $\mathcal{L}^\circ$ centered around $\boldsymbol{\omega} = \mathbf{0}$. Using Parseval's relation [Pin02] and the fact that $\mathsf{P}_{\mathbb{V}(\mathcal{L},\varphi)}f$ is the orthogonal projection of f onto $\mathbb{V}(\mathcal{L}, \varphi)$, we have

$$\|f - \mathsf{P}_{\mathbb{V}(\mathcal{L},\varphi)}f\|^2 = \|f\|^2 - \|\mathsf{P}_{\mathbb{V}(\mathcal{L},\varphi)}f\|^2 = \|\hat{f}\|^2 - \|\widehat{\mathsf{P}_{\mathbb{V}(\mathcal{L},\varphi)}f}\|^2.$$

Taking the Fourier transform of (1.18), we obtain

$$(\widehat{\mathsf{P}_{\mathbb{V}(\mathcal{L},\varphi)}f})(\boldsymbol{\omega}) = \frac{\hat{\varphi}(\boldsymbol{\omega})}{\hat{A}_{\varphi}(\boldsymbol{\omega})}\hat{F}_{\varphi}(\boldsymbol{\omega}), \text{ where the periodic function } \hat{F}_{\varphi}(\boldsymbol{\omega}) := \sum_{\boldsymbol{r}\in\mathcal{L}^\circ} \hat{f}(\boldsymbol{\omega}-\boldsymbol{r})\hat{\varphi}^\star(\boldsymbol{\omega}-\boldsymbol{r}).$$

We can now express the L_2-error in terms of an integral over $\mathcal{V}_{\mathcal{L}^\circ}$.

$$\begin{aligned}\|f - \mathsf{P}_{\mathbb{V}(\mathcal{L},\varphi)}\|^2 &= \int_{\mathbb{R}^s} \Big(\big|\hat{f}(\boldsymbol{\omega})\big|^2 - \big|\frac{\hat{\varphi}(\boldsymbol{\omega})}{\hat{A}_\varphi(\boldsymbol{\omega})} \hat{F}_\varphi(\boldsymbol{\omega})\big|^2 \Big) d\boldsymbol{\omega} \\ &= \int_{\mathcal{V}_{\mathcal{L}^\circ}} \Big(\big|\hat{f}(\boldsymbol{\omega})\big|^2 - \big|\frac{\hat{F}_\varphi(\boldsymbol{\omega})}{\hat{A}_\varphi(\boldsymbol{\omega})}\big|^2 \sum_{\boldsymbol{r}\in\mathcal{L}^\circ} \big|\hat{\varphi}(\boldsymbol{\omega}-\boldsymbol{r})\big|^2 \Big) d\boldsymbol{\omega} \\ &= \int_{\mathcal{V}_{\mathcal{L}^\circ}} \big|\hat{f}(\boldsymbol{\omega})\big|^2 \Big(1 - \frac{\big|\hat{\varphi}(\boldsymbol{\omega})\big|^2}{\hat{A}_\varphi(\omega)}\Big) d\boldsymbol{\omega} \quad = \int_{\mathbb{R}^s} \big|\hat{f}(\boldsymbol{\omega})\big|^2 \Big(1 - \frac{\big|\hat{\varphi}(\boldsymbol{\omega})\big|^2}{\hat{A}_\varphi(\omega)}\Big) d\boldsymbol{\omega}.\end{aligned} \tag{1.21}$$

Here we have used the fact that $\sum_{\boldsymbol{r}\in\mathcal{L}^\circ} \big|\hat{\varphi}(\boldsymbol{\omega}-\boldsymbol{r})\big|^2 = \hat{A}_\varphi(\boldsymbol{\omega})$, and since f is bandlimited, $\big|\hat{F}_\varphi(\boldsymbol{\omega})\big|^2 = \big|\hat{f}(\boldsymbol{\omega})\hat{\varphi}(\boldsymbol{\omega})\big|^2$ inside $\mathcal{V}_{\mathcal{L}^\circ}$. This formula was initially proposed by de Boor *et al.* in the context of the integer lattice $\mathbb{Z}^s$ [dDR94, Theorem 2.20]. Here we have generalized it to arbitrary sampling lattices. Lifting the restriction on h, we obtain the following result for the asymptotic L_2-error [JP01, Theorem 2.3.10].

$$\|f - \mathsf{P}_{\mathbb{V}(\mathcal{L}_h,\varphi)} f\|^2 = \int_{\mathbb{R}^s} \big|\hat{f}(\boldsymbol{\omega})\big|^2 E_{\mathsf{min}}(h\boldsymbol{\omega})\, d\boldsymbol{\omega} = O(h^k), \;\text{ as }\; h \to 0. \tag{1.22}$$

The minimum error kernel is defined as

$$E_{\mathsf{min}}(\boldsymbol{\omega}) := 1 - \frac{\big|\hat{\varphi}(\boldsymbol{\omega})\big|^2}{\hat{A}_\varphi(\boldsymbol{\omega})}. \tag{1.23}$$

Although this error kernel is derived from the perspective of bandlimited functions, it is actually more general. It can also be used to predict the L_2-error for bandlimited functions that are undersampled as well as non-bandlimited functions as explained in Section 1.2.4. It is also valid for both compact and non-compact generators φ. Indeed, it is easy to check that when $\varphi = \mathrm{sinc}_{\mathcal{L}}$, $E_{\mathsf{min}}(\boldsymbol{\omega}) = 1 - \mathbb{1}_{\mathcal{V}_{\mathcal{L}^\circ}}(\boldsymbol{\omega})$, which completely vanishes inside $\mathcal{V}_{\mathcal{L}^\circ}$.

1.2.3 Interpolation and Quasi-interpolation

In many practical cases of interest, f is only known through its point samples on $\mathcal{L}_h$. In such cases, the orthogonal projection cannot be realized and one aims to obtain an oblique projection that is optimal in an asymptotic sense, i.e. it has the same order of decay as the orthogonal projection, albeit, with a higher constant. The two models we are going to consider here are commonly referred to as *interpolation* and *quasi-interpolation*. They both achieve an $O(h^k)$ rate of decay where k is the approximation order provided by the reconstruction kernel φ.

The approximation formula used to obtain f_{app} is similar to the one used for the orthogonal projection (1.18) except for the addition of a correction step that attempts to compensate for the deviation from orthogonality. Denoting $f[\boldsymbol{n}] = f(h\mathbf{L}\boldsymbol{n})$ as the point samples of f, the oblique projection is given by

$$f_{\mathsf{app}}(\boldsymbol{x}) = \sum_{\boldsymbol{n}} (f * q)[\boldsymbol{n}]\varphi(\frac{\boldsymbol{x}}{h} - \mathbf{L}\boldsymbol{n}), \tag{1.24}$$

where q is a correction filter that is applied to the measurements via a discrete convolution operation (denoted by $*$). A useful notion for oblique projections is that of consistency which imposes the requirement that both f and its oblique projection f_{app} be consistent with respect to point sampling measurements [Uns00]. In other words,

$$f[\boldsymbol{m}] = f(h\mathbf{L}\boldsymbol{m}) = f_{\mathsf{app}}(h\mathbf{L}\boldsymbol{m}) = \sum_{\boldsymbol{n}\in\mathbb{Z}^s} (f * q)[\boldsymbol{n}]\varphi(\mathbf{L}(\boldsymbol{m} - \boldsymbol{n})). \tag{1.25}$$

In the signal processing literature, this requirement is a special case of *consistent sampling* [Uns00]. Both interpolation and quasi-interpolation can be cast into this framework. If the reconstruction kernel φ satisfies the consistency requirement (1.25) for any $f \in W_2^k(\mathbb{R}^s)$, we have an *interpolation* scheme. On the other hand, if this requirement is only satisfied for $f \in \Pi_{k-1}(\mathbb{R}^s)$, we have a *quasi-interpolation* scheme.

For an admissible kernel φ, an interpolation filter can be obtained by solving the linear system formed by the consistency requirement (1.25). An easier alternative is to analyze this requirement in the Fourier domain to obtain the correction filter [Uns00]. Denoting $p_\varphi[\boldsymbol{n}] := \varphi(\mathbf{L}\boldsymbol{n})$ as the sampled sequence of φ, the DTFT of (1.25) yields

$$\hat{F}(\boldsymbol{\omega}) = \hat{F}(\boldsymbol{\omega})\hat{Q}(\boldsymbol{\omega})\hat{P}_\varphi(\boldsymbol{\omega}). \tag{1.26}$$

From here, we infer that

$$\hat{Q}(\boldsymbol{\omega}) = 1/\hat{P}_\varphi(\boldsymbol{\omega}). \tag{1.27}$$

Using the inverse sequence $p_\varphi^{-1}[\boldsymbol{n}] \leftrightarrow 1/\hat{P}_\varphi(\boldsymbol{\omega})$, we can define an equivalent kernel as $\varphi^{\mathsf{eq}}(\boldsymbol{x}) := \sum_{\boldsymbol{n}} p_\varphi^{-1}[\boldsymbol{n}]\varphi(\boldsymbol{x} - \mathbf{L}\boldsymbol{n})$. By construction, this kernel is interpolating i.e. $\varphi^{\mathsf{eq}}(\mathbf{L}\boldsymbol{n}) = \delta_{\mathbf{0}}$.

For a quasi-interpolation scheme, the consistency requirement (1.25) is actually a strong requirement. Since φ satisfies the Strang-Fix conditions of order k, it reproduces all polynomials in Π_{k-1}. This implies that f_{app} must be exact, i.e.

$$\forall f \in \Pi_{k-1}(\mathbb{R}^s), \quad f_{\mathsf{app}} = \sum_{\boldsymbol{n}\in\mathbb{Z}^s} (f * q)[\boldsymbol{n}]\varphi(\boldsymbol{x} - \mathbf{L}\boldsymbol{n}) = f(\boldsymbol{x}). \tag{1.28}$$

This imposes some constraints on the admissibility of the quasi-interpolation filter q. In particular, $\left|\hat{Q}(\boldsymbol{\omega})\right|^2$ must be bounded from above and below [AU94, Prop. 6] (see also the Riesz basis condition (1.16)). For an admissible kernel φ, the interpolation filter p_φ^{-1} introduced earlier is also a quasi-interpolation filter.

The asymptotic decay of the L_2-error of these sub-optimal approximation schemes has been well studied (see e.g. [CD90, UD97, BU99b] and references therein). We present here some of the core approximation theoretic ideas that attempt to quantify the lack of biorthogonality between q and φ. These ideas can also be succinctly captured in terms of a Fourier error kernel which is similar to the bandlimited error kernel (1.23). We shall introduce this error kernel formulation in the next section.

Intuitively, an oblique projection and an orthogonal projection will asymptotically behave similarly as long as they are indistinguishable for any $f \in \Pi_{k-1}(\mathbb{R}^s)$, where k is the approximation order provided by φ. This is the same as saying that the continuous moments of $\mathring{\varphi}$ match the discrete moments of $\overline{q}$ up to order $(k-1)$. This concept goes by the name of *quasi-biorthonormality*.

Quasi-biorthonormality: *The oblique projection correction filter q and the reconstruction kernel φ are said to be quasi-biorthonormal of order k if for all monomials $\boldsymbol{x}^{\boldsymbol{\alpha}} := x_1^{\alpha_1} \cdots x_s^{\alpha_s}$ with $\left|\boldsymbol{\alpha}\right| \leq (k-1)$,*

$$\int_{\mathbb{R}^s} \boldsymbol{x}^{\boldsymbol{\alpha}} \mathring{\varphi}(\boldsymbol{x}) d\boldsymbol{x} = \sum_{\boldsymbol{n}} (\mathbf{L}\boldsymbol{n})^{\boldsymbol{\alpha}} \overline{q}[\boldsymbol{n}]. \tag{1.29a}$$

This condition has the following equivalent formulation in the Fourier domain [BU99b, Lemma 2].

$$\hat{Q}^\star(\boldsymbol{\omega}) \hat{\varphi}(\boldsymbol{\omega} + \mathbf{L}^{-\mathsf{T}}\boldsymbol{n}) = \delta_{\boldsymbol{n}} + O(\|\boldsymbol{\omega}\|^k), \quad \forall \boldsymbol{n} \in \mathbb{Z}^s. \tag{1.29b}$$

If there is order k quasi-biorthonormality between q and φ, then for any $f \in W_2^k(\mathbb{R}^s)$, the approximation $f_{\mathsf{app}} \in \mathbb{V}(\mathcal{L}_h, \varphi)$ obtained via (1.24) satisfies $\|f - f_{\mathsf{app}}\| = O(h^k)$, as $h \to 0$ [BU99b, Theorem 1]. In the univariate setting, a tight bound exists and is related to the k-th order derivatives of $\hat{\varphi}$ [UD97, Prop. 5.1]. Particularly, we have

$$\|f - f_{\mathsf{app}}\| \leq C_\varphi^{\mathsf{Q}} \|D^k f\| h^k \;\text{ as }\; h \to 0, \quad \text{where} \quad C_\varphi^{\mathsf{Q}} = \frac{1}{k!} \sqrt{\sum_{n \in \mathbb{Z}} \left|D^k \hat{\varphi}(n)\right|^2}. \tag{1.30}$$

Note that the asymptotic constant C_φ^{Q} is the same as the minimum error constant C_φ (1.20) except for the additional contribution due to the $n = 0$ term.

1.2.4 Fourier Error Kernel

Most of the ideas discussed so far can be succinctly expressed in terms of a Fourier domain error kernel proposed by Blu and Unser [BU99a,BU99b] in the univariate setting. We present the multivariate extension of this kernel here and remark on the equivalence between the asymptotic error behaviour predicted by this kernel and the results presented in the previous sections.

The error kernel $E(\boldsymbol{\omega})$ is composed of two additive terms, a minimum orthogonal projection error term and a residue error term that comes into play when the approximation is sub-optimal. It is defined as

$$E(\boldsymbol{\omega}) := \underbrace{1 - \frac{\left|\hat{\varphi}(\boldsymbol{\omega})\right|^2}{\hat{A}_\varphi(\boldsymbol{\omega})}}_{E_{\mathsf{min}}(\boldsymbol{\omega})} + \underbrace{\hat{A}_\varphi(\boldsymbol{\omega})\left|\hat{Q}(\boldsymbol{\omega}) - \hat{\mathring{\varphi}}(\boldsymbol{\omega})\right|^2}_{E_{\mathsf{res}}(\boldsymbol{\omega})}. \tag{1.31}$$

Denoting $f^{\boldsymbol{\xi}}(\boldsymbol{x}) := f(\boldsymbol{x} - \boldsymbol{\xi})$ as a shifted version of f, the error kernel can be used to obtain the L_2-error between $f^{\boldsymbol{\xi}}$ and its approximation $f^{\boldsymbol{\xi}}_{\mathsf{app}} \in \mathbb{V}(\mathcal{L}_h, \varphi)$, averaged over all possible shifts $\boldsymbol{\xi} \in \mathbb{R}^s$. In particular [BU99a, Theorem 2],

$$\frac{1}{h^s}\int_{\mathcal{V}_{\mathcal{L}_h}} \|f^{\boldsymbol{\xi}} - f^{\boldsymbol{\xi}}_{\mathsf{app}}\|^2 d\boldsymbol{\xi} = \int_{\mathbb{R}^s} \left|\hat{f}(\boldsymbol{\omega})\right|^2 E(h\boldsymbol{\omega}) d\boldsymbol{\omega}. \tag{1.32}$$

Observe that $E_{\mathsf{min}}(\boldsymbol{\omega})$ also appears earlier in (1.22) in the context of bandlimited functions. Here, it appears again in a more general context. All that is required is that f be no more than continuous, which is satisfied whenever $f \in W_2^r(\mathbb{R}^s)$ with $r > \frac{1}{2}$. For an orthogonal projection, the residue term vanishes and the average minimum error is quantified by $E_{\mathsf{min}}(\boldsymbol{\omega})$. On the other hand, for interpolative and quasi-interpolative approximation scenarios there is additional error that is quantified by $E_{\mathsf{res}}(\boldsymbol{\omega})$.

Besides quantifying the average error, the kernel $E(\boldsymbol{\omega})$ can also be used to obtain an upper bound for the asymptotic error. Indeed, the Strang-Fix conditions (1.12) and quasi-biorthonormality (1.29) have equivalent formulations in terms of E_{min} and E_{res} respectively.

For the Strang-Fix conditions, we have the following equivalence.

1. *φ satisfies the Strang-Fix conditions of order k.*
2. $E_{\mathsf{min}}(\boldsymbol{\omega}) = O(\|\boldsymbol{\omega}\|^{2k})$.

Quasi-birorthonormality can be expressed in terms of a similar equivalence.

1. *φ and q are quasi-birorthonormal of order k.*

2. *$E_{\mathsf{res}}(\boldsymbol{\omega}) = O(\|\boldsymbol{\omega}\|^{2k})$.*

Furthermore, if φ satisfies the Strang-Fix conditions of order k, and there is order k quasi-biorthonormality between q and φ, then the asymptotic error is bounded by the order $2k$ coefficients of the Taylor developments of $E(\boldsymbol{\omega})$ around $\boldsymbol{\omega} = \mathbf{0}$ [BU99a, Sec. IV]. In particular,

$$\|f - f_{\mathsf{app}}\|^2 \leq h^{2k} \sum_{|\boldsymbol{\alpha}|=2k} \frac{\big(D^{\boldsymbol{\alpha}} E\big)(\mathbf{0})}{\alpha_1! \cdots \alpha_s!} \big(\textstyle\int_{\mathbb{R}^s} \omega_1^{\alpha_1} \cdots \omega_s^{\alpha^s} \big|\hat{f}(\boldsymbol{\omega})\big|^2 d\boldsymbol{\omega} \big), \quad \text{as} \quad h \to 0, \tag{1.33}$$

provided that $f \in W_2^k(\mathbb{R}^s)$. By restricting this result to the univariate setting, we can also obtain the asymptotic constants C_φ (1.20) and C_φ^{Q} (1.30) [BU99a]. For the minimum approximation scenario, it can be shown that

$$\frac{\big(D^{2k} E_{\mathsf{min}}\big)(0)}{(2k)!} = \frac{1}{(k!)^2} \sum_{n \neq 0} \big|D^k \hat{\varphi}(n)\big|^2 = C_\varphi^2, \tag{1.34}$$

whereas for the interpolation and quasi-interpolation scenarios, we have

$$\frac{\big(D^{2k} E\big)(0)}{(2k)!} = \frac{1}{(k!)^2} \sum_{n \in \mathbb{Z}} \big|D^k \hat{\varphi}(n)\big|^2 = (C_\varphi^{\mathsf{Q}})^2. \tag{1.35}$$

In the univariate setting, this error kernel has been used to obtain splines that have maximal approximation order with minimum support [BTU01], improve the quality of linear interpolation by a simple shift of the kernel [BTU04], and to design quasi-interpolating prefilters [CBU05]. It has also been extended to handle the quantitative error analysis of periodic functions [JBU02] and derivatives [CM11]. It is an attractive tool both from an analysis and design perspective. In the following section, we shall make use of it to compare the approximation characteristics of shift-invariant spaces associated with the three-dimensional cubic lattices.

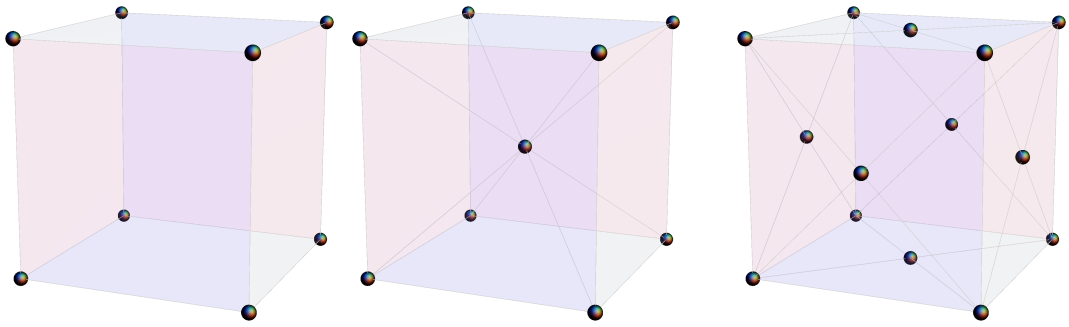
1.3 Shift-invariant Spaces on the Cubic Lattices

1.3.1 Three-dimensional Cubic Lattices

The (normalized) three-dimensional cubic lattices, namely, the CC lattice $\mathbb{Z}^3$, the BCC lattice $\mathcal{B}$, and the FCC lattice $\mathcal{F}$ are generated by the matrices

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \frac{1}{\sqrt[3]{4}} \begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix}, \quad \text{and} \quad \mathbf{F} = \frac{1}{\sqrt[3]{2}} \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{1.36}$$

respectively. The BCC lattice, isotropically scaled by a factor of $\sqrt[3]{4}$, is a sub-lattice of the CC lattice that is obtained by retaining all those points whose coordinates have the same parity, i.e. they are either all odd or they are all even. The resulting lattice is four times less dense. In a similar fashion, the FCC lattice, scaled by a factor of $\sqrt[3]{2}$, is a sub-lattice of the CC lattice that satisfies the checkerboard property [CS99], i.e. it contains all those points whose coordinate sum is even. The resulting lattice is therefore half as dense. These lattices are also important in crystallography and solid state physics where they are commonly known as the cubic *Bravais* lattices and describe the arrangement of crystal systems [PB11]. The term 'cubic' stems from the fact that these lattices can all be obtained from a tessellation of cubes. The CC lattice consists of the vertices of the cubes, the scaled BCC lattice has an additional point in the center of each cube in addition to the vertices, whereas the scaled FCC lattice consists of the six face centers of each cube in addition to the vertices as shown in Fig. 1.1.



(a) CC (b) BCC (c) FCC

Figure 1.1: The three-dimensional cubic lattices.

1.3.2 Sphere Packing

The sphere packing density of the cubic lattices is important from an approximation theoretic perspective as discussed earlier. When projecting a function to the cardinal space $\mathbb{V}(\mathcal{L}, \mathrm{sinc}_{\mathcal{L}})$, the error is quantified by (1.23) with the kernel being $E_{\mathsf{min}}(\boldsymbol{\omega}) = 1 - \mathbb{1}_{\mathcal{V}_{\mathcal{L}^\circ}}(\boldsymbol{\omega})$. This kernel vanishes inside the Voronoi cell $\mathcal{V}_{\mathcal{L}^\circ}$, which implies that the only error contribution is due to the out-of-band portion of the spectrum. Thus, the optimal lattice $\mathcal{L}$ is the one that captures as much of the spectrum within $\mathcal{V}_{\mathcal{L}^\circ}$ (also known as the *Brillouin* zone [PB11]) as possible. When there are no preferred directions, the minimum error will therefore be achieved by the lattice that maximizes the packing efficiency i.e., the ratio of the volume of the inscribed sphere to the volume of the Brillouin zone.

Lattice $\mathcal{L}$ (generator)	**Dual** $\mathcal{L}^\circ$ (generator)	**Voronoi cell** $\mathcal{V}_{\mathcal{L}^\circ}$ (edge length a)	**Inscribing radius**	**Efficiency**
CC: **I**	CC: **I**	Cube: $a = 1$	$\frac{a}{2}$	$\frac{\pi}{6} \approx 0.52$
BCC: **B**	FCC: **F**	Rhombic dodecahedron: $a = \sqrt[3]{\frac{9}{16\sqrt{3}}}$	$\sqrt{\frac{2}{3}}a$ [Wik12a]	$\frac{\pi}{3\sqrt{2}} \approx 0.74$
FCC: **F**	BCC: **B**	Truncated octahedron: $a = \sqrt[3]{\frac{1}{8\sqrt{2}}}$	$\frac{\sqrt{6}}{2}a$ [Wik12b]	$\frac{\sqrt{3}\pi}{8} \approx 0.68$

Table 1.1: Packing efficiency of the three cubic lattices. The generating matrices are normalized so that each lattice has a unit volume Voronoi cell. The volume of a rhombic dodecahedron of edge length a is $\frac{16}{9}\sqrt{3}a^3$ [Wik12a], while that of a truncated octahedron of edge length a is $8\sqrt{2}a^3$ [Wik12b].

As listed in Table 1.1, the self-dual CC lattice has the worst packing efficiency of the three cubic lattices. The most efficient packing lattice is the FCC lattice which implies that its dual, the BCC lattice is the most efficient sampling lattice. A more pictorial account can be found in Entezari's PhD thesis [Ent07].

1.3.3 Spline-like Spaces on the Cubic Lattices

Unlike the cardinal space $\mathbb{V}(\mathcal{L}_h, \mathrm{sinc}_{\mathcal{L}})$, spline-like spaces are spanned by the dilation and shifts of compact spline-like generators that are easy to work with computationally since they are built by piecing polynomials together. Such spaces are usually studied from the perspective of the Cartesian lattice $\mathbb{Z}^s$, even though it has a poor packing efficiency. The univariate B-splines [UAE93] are a popular choice since they can be easily extended to the Cartesian lattice $\mathbb{Z}^s$ via a separable tensor product. Non-separable spline kernels, albeit

on the Cartesian lattice, have also been investigated (see e.g. [Chu88, dHR93, Wan01] and references therein). One notable exception is the hex-splines [VBU$^+$04, CVDV07] that reside on the two-dimensional hexagonal lattice and are built by the successive convolutions of the indicator function of the Voronoi cell of the hexagonal lattice.

There has been some recent progress towards the development of spline-like generators on the non-Cartesian cubic lattices. Entezari *et al.* proposed second and fourth order box splines for function approximation on the BCC lattice [EDM04]. They later derived closed from polynomial representations [EVM08], proposed quasi-interpolants [EMK09] and implemented their efficient approximation algorithm on the GPU [FEVM10]. Kim *et al.* investigated box-spline generators on the FCC lattice [KEP08], and later generalized their construction scheme to the (non-Cartesian) *root* lattices in arbitrary dimension [KP10, KP11]. The idea of successively convolving the indicator function of the Voronoi cell has also been investigated. Preliminary empirical studies on the BCC lattice were presented by Csébfalvi [Csé08a]. Mirzargar and Entezari formalized this notion and proposed *Voronoi* splines [ME10], as well as their associated quasi-interpolatants [ME11]. Another line of research has focused on using the tensor-product B-splines on the BCC lattice [Csé10, DC10].

It is now well-known that spline like spaces on the non-Cartesian cubic lattices outperform their Cartesian counterparts. Additionally, reconstruction kernels on the non-Cartesian lattices also tend to be more compact while providing the same order of accuracy as compared to the CC lattice. However, comparisons of the various approximation spaces on the cubic lattices have largely been qualitative [MES$^+$11, DC11] and, to the best of our knowledge, asymptotic bounds such as (1.20), (1.30), and (1.33) have not been investigated. In the remainder of this chapter, we focus on establishing such bounds since they provide a comparison that is more relevant in practical settings than the idealized packing efficiency (Table 1.1).

1.3.4 Review of Box Splines

Many of the approximation spaces associated with the cubic lattices are spanned by box spline generators. Consequently, we briefly review the topic of box splines here. For additional details, we refer the reader to the comprehensive text by de Boor *et al.* [dHR93].

The box splines are very useful reconstruction functions that are well suited for designing approximation spaces on arbitrary sampling lattices. They can be constructed to satisfy the

Strang-Fix relations (1.12), and have attractive mathematical properties similar to the B-splines. Associated with a box spline in $\mathbb{R}^s$ is an $s \times n$ $(n \geq s)$ matrix $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1\ \boldsymbol{\xi}_2\ \cdots\ \boldsymbol{\xi}_n]$ consisting of direction vectors $\boldsymbol{\xi}_i$. We denote this box spline as $M_{\boldsymbol{\Xi}}(\boldsymbol{x})$. It is obtained by successively convolving line segments along the direction vectors contained in $\boldsymbol{\Xi}$. The simplest box spline is obtained by choosing $n = s$ linearly independent direction vectors and is the indicator function of the parallelepiped formed by these direction vectors. Successive directional convolutions are defined inductively as

$$M_{[\boldsymbol{\Xi}\ \boldsymbol{\xi}]}(\boldsymbol{x}) := \int_0^1 M_{\Xi}(\boldsymbol{x} - t\boldsymbol{\xi})dt. \tag{1.37}$$

The box splines have a simple Fourier domain representation given by

$$\hat{M_{\boldsymbol{\Xi}}}(\boldsymbol{\omega}) = \prod_{i=1}^{n} \frac{1 - \exp(-2\pi\imath {\boldsymbol{\xi}_i}^{\mathsf{T}}\boldsymbol{\omega})}{2\pi\imath {\boldsymbol{\xi}_i}^{\mathsf{T}}\boldsymbol{\omega}}. \tag{1.38}$$

The support of $M_{\boldsymbol{\Xi}}$ consists of all the points contained within the polytope formed by taking the Minkowski sum of the direction vectors in $\boldsymbol{\Xi}$. This implies that $M_{\boldsymbol{\Xi}}$ is centered at the point $c_{\boldsymbol{\Xi}} := \sum_{i=1}^{n} \frac{1}{2}\boldsymbol{\xi}_i$. When the box spline $M_{\boldsymbol{\Xi}}$ is used in conjunction with the Cartesian lattice $\mathbb{Z}^s$, the smoothness and approximation order of the space $\mathbb{V}(\mathbb{Z}_h^s, M_{\boldsymbol{\Xi}})$ can be readily obtained simply by inspecting the columns of $\boldsymbol{\Xi}$ [dHR93].

We now restrict attention to the case $s = 3$ and summarize the key features of some box spline spaces associated with the cubic lattices.

Tensor-product B-splines

The non-centered tensor product B-splines are a special case of the box splines obtained by choosing $\mathbf{I}^k := [\mathbf{I}\ \cdots\ \mathbf{I}]$ (k repetitions) as the direction matrix. A closed form representation in terms of the centered B-splines $\beta^k(x)$ is given by

$$M_{\mathbf{I}^k}(\boldsymbol{x}) = \prod_{i=1}^{3} \beta^k(x_i + \tfrac{k+1}{2}), \quad \text{where} \quad \beta^k(x) := \sum_{j=0}^{k+1} \frac{(-1)^j}{k!} \binom{k+1}{j} \left(x + \frac{k+1}{2} - j\right)_+^k, \tag{1.39}$$

and $(x)_+^k := \max(0, x)^k$ [UAE93].

We shall be making extensive use of the trivariate centered tensor-product B-splines and denote them as $B^k(\boldsymbol{x}) := \prod_{i=1}^{3} \beta^k(x_i)$, where k is the polynomial degree of the univariate centered B-spline $\beta^k(x)$. They are commonly used in conjunction with the CC lattice to approximate functions in the space $\mathbb{V}(\mathbb{Z}_h^3, B^k)$, providing an approximation order of $(k + 1)$ [BTU01]. In fact, they are precisely the Voronoi splines of the CC lattice [ME10].

Rhombic Dodecahedral Box Splines

As the name suggests, these box splines are constructed by successive convolutions of a fundamental box spline that is completely supported within a rhombic dodecahedron formed by the Voronoi relevant neighbors of the BCC lattice [EDM04, EVM08] (Fig. 1.2). The fundamental four-directional box spline is built from the direction matrix

$$\boldsymbol{\Theta} := \begin{bmatrix} \boldsymbol{\theta}_1 & \boldsymbol{\theta}_2 & \boldsymbol{\theta}_3 & \boldsymbol{\theta}_4 \end{bmatrix} = \frac{1}{\sqrt[3]{4}} \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & 1 & -1 \end{bmatrix}, \tag{1.40}$$

and consists of linear polynomial pieces. When used in conjunction with the BCC lattice to approximate in the space $\mathbb{V}(\mathcal{B}_h, M_{\boldsymbol{\Theta}})$, it provides a second-order approximation. The approximation order can be successively increased by convolving this box spline with itself. Thus, $\boldsymbol{\Theta}^k$, the matrix obtained by appending k copies of $\boldsymbol{\Theta}$, gives rise to a box spline that has an approximation order of $2k$. In particular, $M_{\boldsymbol{\Theta}^2}$ consists of quintic polynomial pieces and provides a fourth order approximation on the BCC lattice.

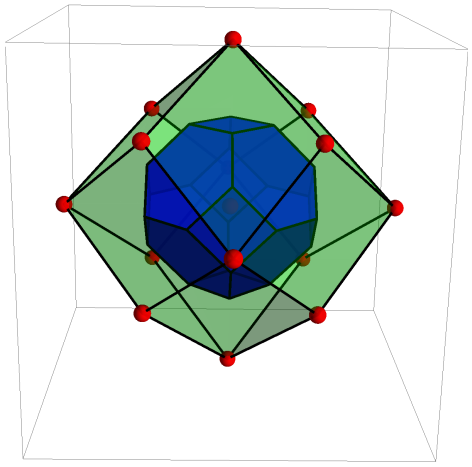


Figure 1.2: The Voronoi cell of the BCC lattice is a truncated octahedron (blue). The Voronoi relevant neighbors form a rhombic dodecahedron (green) that is also the support of the linear box spline.

Voronoi Splines on the FCC Lattice

The Voronoi cell of the FCC lattice, isotropically scaled by a factor of $\sqrt[3]{2}$, is a rhombic dodecahedron having a volume of 2. It can be split up into four parallelepipeds by choosing any three linearly independent vectors (scaled by $\frac{1}{2}$) from the direction matrix $\boldsymbol{\Theta}$. The first-order Voronoi spline on the scaled FCC lattice can therefore be expressed as a linear combination of four three-directional box splines [ME10]. It is given by

$$V_0^{\mathcal{F}} = \frac{1}{4}\big(M_{\frac{1}{2}[\boldsymbol{\theta}_1\ \boldsymbol{\theta}_2\ \boldsymbol{\theta}_3]} + M_{\frac{1}{2}[\boldsymbol{\theta}_1\ \boldsymbol{\theta}_2\ \boldsymbol{\theta}_4]} + M_{\frac{1}{2}[\boldsymbol{\theta}_1\ \boldsymbol{\theta}_3\ \boldsymbol{\theta}_4]} + M_{\frac{1}{2}[\boldsymbol{\theta}_2\ \boldsymbol{\theta}_3\ \boldsymbol{\theta}_4]}\big), \tag{1.41}$$

where the leading normalization factor ensures that $\int_{\mathcal{V}_{\mathcal{F}}} V_0^{\mathcal{F}}(\boldsymbol{x})d\boldsymbol{x} = 1$. Higher order Voronoi splines are given by the recursive convolution relationship

$$V_k^{\mathcal{F}}(\boldsymbol{x}) = \big(V_{k-1}^{\mathcal{F}} * V_0^{\mathcal{F}}\big)(\boldsymbol{x}), \tag{1.42}$$

which ensures that each successive convolution increases the approximation order by one. Thus, $V_k^{\mathcal{F}}$ has an approximation order of $k+1$ on the FCC lattice. By substituting the box spline pieces (1.41) into the convolution relationship (1.42), one can also obtain the constituent box splines of the higher order Voronoi spline $V_k^{\mathcal{F}}$.

Voronoi splines on the BCC lattice can be similarly characterized in terms of a six-directional box spline. However, the construction is slightly more involved and relies on breaking up a truncated octahedron into sixteen parallelepipeds [ME10].

1.3.5 Comparison of Approximation Spaces

We can now use the analysis tools introduced earlier to quantitatively compare the approximation spaces associated with the cubic lattices. We focus on the orthogonal projection case since it gives us a sense of the minimum error achievable by a lattice.

Error for Radially Bandlimited Functions

Without loss of generality, let us assume that the function of interest is radially bandlimited with its spectrum completely contained in the unit diameter sphere $\mathcal{S} := \{\boldsymbol{\omega} \in \mathbb{R}^3 : \|\boldsymbol{\omega}\| \leq \frac{1}{2}\}$. Using (1.22), the squared L_2-error at scale h is bounded according to

$$\|f - f_{\mathsf{app}}\|^2 = \int_{\mathcal{S}} \big|\hat{f}(\boldsymbol{\omega})\big|^2 E_{\mathsf{min}}(h\boldsymbol{\omega})d\boldsymbol{\omega} \leq \big\|\big|\hat{f}\big|^2\big\|_\infty \eta(h), \tag{1.43}$$

where $\eta(h) := \int_{\mathcal{S}} E_{\mathsf{min}}(h\boldsymbol{\omega})d\boldsymbol{\omega}$. The measure $\eta(h)$ tells us how good a space is in approximating functions that are radially bandlimited. It also measures the exact approximation error for the three-dimensional *jinc* function $f(\boldsymbol{x}) = 2\sqrt{2}\frac{J_{3/2}(\pi\|\boldsymbol{x}\|)}{\|\boldsymbol{x}\|^{3/2}}$, where $J_{3/2}$ is the Bessel function of the first kind of order 3/2. The Fourier transform of this function is a unit diameter sphere [Miz73].

Using this definition of $\eta(h)$ on the CC lattice, if f were to be point-sampled, it would be critically sampled when $h = 1$, undersampled when $h > 1$ and oversampled when $h < 1$. Note that $\eta(h)$ measures the orthogonal projection error which attempts to quantify the minimum error achievable. On the other hand, measures such as prealiasing and postaliasing that compare $\hat{\varphi}$ with $\mathbb{1}_{V_{\mathcal{L}}^\circ}$ are sub-optimal since they do not take the autocorrelation function A_φ into consideration. Thus, $\eta(h)$ veritably reflects the approximation capabilities of a space.

Asymptotic Error for Isotropic Functions

When f is an isotropic function that belongs to $W_2^k(\mathbb{R}^3)$, the rate of decay of the error as $h \to 0$ can be obtained from the $2k$ order terms of the Taylor developments of $E_{\mathsf{min}}(\boldsymbol{\omega})$ around $\boldsymbol{\omega} = \mathbf{0}$. According to (1.33), the mean squared asymptotic error is dominated by the $2k$ order terms where k is the approximation order. For an isotropic function, we can use spherical coordinates to rearrange the right hand side of (1.33) to obtain

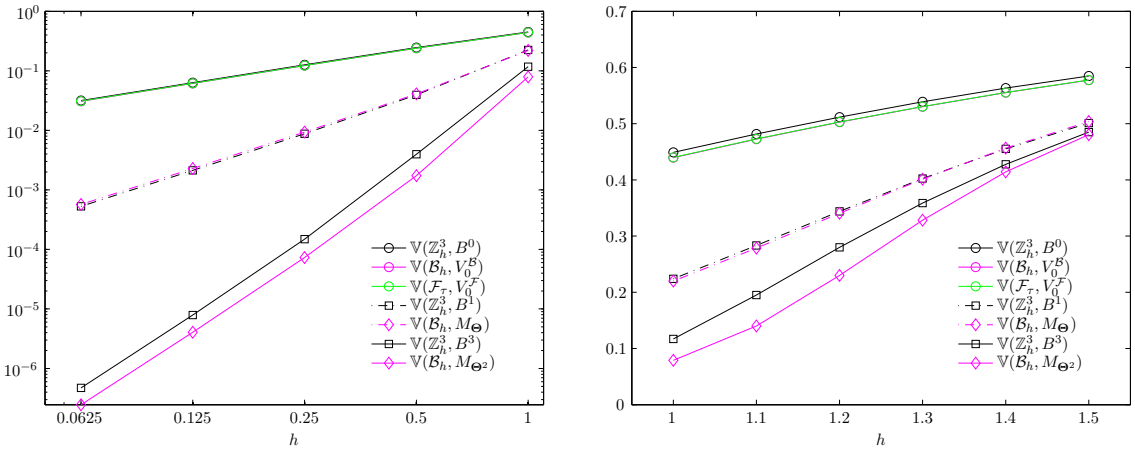
$$\|f - f_{\mathsf{app}}\|^2 \leq K_f h^{2k} \kappa. \tag{1.44}$$

Here, the constant $K_f := \int_0^\infty \left|\hat{f}(r)\right|^2 r^{2(k+1)} dr < \infty$ and $\kappa := \int_0^{2\pi}\int_0^{\pi} C_{\mathsf{op}}(\theta,\phi)\sin\theta\, d\theta d\phi$, where the spherical function $C_{\mathsf{op}}(\theta,\phi)$ is obtained from the Taylor developments of $E_{\mathsf{min}}(\boldsymbol{\omega})$ according to

$$C_{\mathsf{op}}(\theta,\phi) = \sum_{|\boldsymbol{\alpha}|=2k} \frac{\left(D^{\boldsymbol{\alpha}} E_{\mathsf{min}}\right)(\mathbf{0})}{\alpha_1!\alpha_2!\alpha_3!} (\sin\theta\cos\phi)^{\alpha_1}(\sin\theta\sin\phi)^{\alpha_2}(\cos\theta)^{\alpha_3}. \tag{1.45}$$

Thus, the constant κ provides an upper bound for the asymptotic error when approximating isotropic functions that belong to $W_2^k(\mathbb{R}^3)$.

The main ingredient for both the measures $\eta(h)$ and κ is the minimum error kernel $E_{\mathsf{min}}(\boldsymbol{\omega})$ which can be evaluated in closed form for a variety of spaces of interest. Particularly, we need the Fourier transform of the generator φ and the DTFT of its autocorrelation sequence (1.15). For the box splines, the Fourier transform is given by (1.38) while the



(a) Oversampled (b) Undersampled

Figure 1.3: The variation of $\sqrt{\eta(h)}$ for a variety of approximation spaces on the cubic lattices. For the first-order Voronoi spline on FCC, $\tau = \frac{h}{\sqrt[3]{2}}$. For each lattice, $\eta(h)$ is computed by numerically integrating the closed form expression for $E_{\mathsf{min}}(\boldsymbol{\omega})$. Data for the critically sampled case ($h = 1$) is also presented in Table 1.2.

autocorrelation sequence can be evaluated using the recursive algorithm of de Boor [deB93]. Owing to the algebraic complexity of the calculations involved, we resort to using a computer algebra system such as *Mathematica* [Wol10].

Table 1.2: Comparison of approximation spaces associated with the cubic lattices

k	Space	Supp.	$\frac{0.4489}{\sqrt{\eta(1)}}$	$\sqrt{\kappa}$	$\frac{1}{2\pi\sqrt{6}}/\sqrt{\kappa}$
1	$\mathbb{V}(\mathbb{Z}_h^3, B^0)$	1	1.000	$\frac{1}{2\pi\sqrt{6}}$	1.00
	$\mathbb{V}(\mathcal{B}_\sigma, V_0^{\mathcal{B}})$	1	1.021	$\frac{\sqrt{19/3}}{32\pi\sqrt[3]{2}}$	1.03
	$\mathbb{V}(\mathcal{F}_\tau, V_0^{\mathcal{F}})$	1	1.020	$\frac{1}{4\pi\sqrt[3]{2}}$	1.03
2	$\mathbb{V}(\mathbb{Z}_h^3, B^1)$	8	2.005	$\frac{1}{20\pi\sqrt{6}}$	10.00
	$\mathbb{V}(\mathcal{B}_\sigma, M_{\boldsymbol{\Theta}})$	4	2.041	$\frac{\sqrt{11}}{120\pi\sqrt[3]{2}}$	9.31
4	$\mathbb{V}(\mathbb{Z}_h^3, B^3)$	64	3.836	$\frac{1}{720\pi\sqrt{14}}$	549.91
	$\mathbb{V}(\mathcal{B}_\sigma, M_{\boldsymbol{\Theta}^2})$	32	5.689	$\frac{\sqrt{\frac{467}{5}}}{80640\pi\sqrt[3]{2}}$	675.92

Figure 1.3 plots $\sqrt{\eta(h)}$ for a number of sampling rates in the oversampled and undersampled regimes. As expected, the oversampled regime reveals that the Voronoi spline spaces $\mathbb{V}(\mathbb{Z}_h^3, B^0)$, $\mathbb{V}(\mathcal{B}_h, V_0^{\mathcal{B}})$, and $\mathbb{V}(\mathcal{F}_\tau, V_0^{\mathcal{F}})$ exhibit a first-order decay; the spaces $\mathbb{V}(\mathbb{Z}_h^3, B^1)$ and $\mathbb{V}(\mathcal{B}_h, M_{\boldsymbol{\Theta}})$ have a second-order decay; and the spaces $\mathbb{V}(\mathbb{Z}_h^3, B^3)$ and $\mathbb{V}(\mathcal{B}_h, M_{\boldsymbol{\Theta}^2})$ have a

fourth-order decay. A better sense of the rate of decay can also be obtained from the constant $\sqrt{\kappa}$ which can be evaluated in closed form (Table 1.2). Asymptotically, the non-Cartesian Voronoi spline spaces are marginally better that the Cartesian Voronoi spline space. A similar trend is observed in the undersampled regime as well. Although not demonstrated here, we expect that higher-order Voronoi spline spaces would only amplify this improvement.

The second-order trilinear B-spline space $\mathbb{V}(\mathbb{Z}_h^3, B^1)$ is asymptotically a little better than the linear box spline space $\mathbb{V}(\mathcal{B}_h, M_{\boldsymbol{\Theta}})$. However, this trend is reversed in the neighborhood around $h = 1$ where the BCC lattice outperforms owing to its greater isotropy. This is quite remarkable indeed since the linear box spline reconstruction, owing to its smaller support as compared to the trilinear B-spline, only needs to access half as many coefficients. When $k = 4$, the BCC box spline space $\mathbb{V}(\mathcal{B}_h, M_{\boldsymbol{\Theta}^2})$ has a distinct advantage over the Cartesian space $\mathbb{V}(\mathbb{Z}_h^3, B^3)$. Again, this is noteworthy since a quintic box spline approximation needs to access half as many coefficients as its Cartesian counterpart. Furthermore, the improvement exists in the undersampled and oversampled regimes alike.

The advantage of the BCC lattice over the CC lattice has also been demonstrated empirically for the critically sampled case [EDM04, EVM08, FEVM10]. However, to the best of our knowledge, quantitative measures such as the ones listed in Table 1.2 have not been explored before.

Although the BCC box spline spaces are spanned by more compact kernels, this does not lead to faster evaluations in practice owing to the non-separability of the polynomial pieces. In fact, the separable tensor-product B-splines lend themselves to highly optimized software and GPU implementations that are more efficient as compared to their BCC counterparts [FEVM10, Csé10].

1.4 Summary of Contributions

The substantial gain in approximation quality offered by the BCC lattice warrants that acquisition and processing techniques be closely examined so that researchers and practitioners have a viable alternative to the ubiquitous Cartesian lattice. This dissertation focuses on the processing realm and makes the following contributions.

Gradient Estimation. Our primary motivation for investigating this problem comes from volume visualization where, the (normalized) gradient of a function is used to simulate the interaction of light with a three-dimensional volume. We cast this problem into the framework of shift-invariant spaces. In Chapter 2, we use the lattice samples of f to approximate the three orthogonal gradient components independently in the same shift-invariant space $\mathbb{V}(\mathcal{L}_h, \varphi)$. The coefficients are obtained by convolving the samples with infinite impulse response (IIR) derivative filters. Since these filters are not compactly supported, they have to be applied in a preprocessing step, and the resulting coefficients have to be stored in a gradient volume, thereby increasing the storage overhead. In order to mitigate this overhead, Chapter 3 examines the problem in light of the quantitative analysis tools introduced in Section 1.2. This results in a modified framework where the gradient components — which are no longer required to be orthogonal — are approximated in separate spaces that are spanned by shifted versions of a symmetric generator.

The Poisson Equation. In Chapter 4, we investigate the use of shift-invariant spaces in solving inverse problems. In particular, we examine Poisson's equation within a rectangular domain with homogeneous Dirichlet boundary conditions. We analyze the problem in the Fourier domain and identify the solution operator that needs to be discretized. One of our main contributions is a reformulation of the Fourier error kernel of Blu and Unser [BU99a] (see (1.31) on page 14) that allows us to quantify the error incurred when approximating a derived quantity.

Discrete Fourier Transform on BCC. The FFT is a fundamental data processing tool that extends to the CC lattice in a simple tensor product manner. In contrast, a discrete Fourier transform (DFT) on the BCC lattice is non-separable. However, as we demonstrate in Chapter 5, it can be made separable by choosing a rectangular sampling window in the spatial domain. Choosing such a window amounts to a rectangular sampling of the replicated spectra in the Fourier domain. We exploit the geometry of the dual FCC lattice and identify suitable rectangular regions in the Fourier domain so that the FFT can be used to evaluate the DFT as well as the DST. This allows us to efficiently implement discrete convolution operations on the BCC lattice in the Fourier domain.

1.5 Notes

1.5.1 Derivation of the Strang-Fix Conditions

The Strang-Fix conditions (1.12) are usually studied from the perspective of the integer lattice $\mathbb{Z}^s$ (see e.g. [deB87, Jia98] for derivations). In order to make our presentation self-contained, we show how to extend these conditions to arbitrary sampling lattices. For a (normalized) sampling lattice $\mathcal{L}$, the Strang-Fix conditions state that:

$$\begin{aligned} &\mathbb{V}(\mathcal{L},\varphi) \ \text{ contains } \ \Pi_{k-1}(\mathbb{R}^s) \\ \Longrightarrow\ &\hat{\varphi}(\mathbf{0}) = 1, \quad \text{and} \quad D^{\boldsymbol{\alpha}}\hat{\varphi}(\boldsymbol{\beta}) = 0 \ \text{ for } \ \big|\boldsymbol{\alpha}\big| \le k-1 \ \text{ and } \ \forall \boldsymbol{\beta} \in \mathcal{L}^\circ \backslash \{\mathbf{0}\}. \end{aligned}$$

Proof. $\mathbb{V}(\mathcal{L},\varphi)$ contains $\Pi_{k-1}(\mathbb{R}^s)$ means that, for any monomial $\boldsymbol{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1}\cdots x_s^{\alpha_s}$ with $\big|\boldsymbol{\alpha}\big| \le (k-1)$,

$$\sum_{\boldsymbol{n}\in\mathbb{Z}^s} \varphi(\boldsymbol{x}-\mathbf{L}\boldsymbol{n})(\mathbf{L}\boldsymbol{n})^{\boldsymbol{\alpha}} = \boldsymbol{x}^{\boldsymbol{\alpha}}, \ \ \forall \boldsymbol{x} \in \mathbb{R}^s. \tag{1.46}$$

For a fixed $\boldsymbol{x} \in \mathbb{R}^s$, define the function $\psi(\boldsymbol{y}) := \varphi(\boldsymbol{x}-\boldsymbol{y})\boldsymbol{y}^{\boldsymbol{\alpha}}$. Taking its Fourier transform, we obtain $\hat{\psi}(\boldsymbol{\omega}) = \big(\frac{\imath}{2\pi}\big)^{|\boldsymbol{\alpha}|} D^{\boldsymbol{\alpha}}\left(\hat{\varphi}(-\boldsymbol{\omega})\exp\big(-2\pi\imath\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{x}\big)\right)$.

Using the Poisson summation formula (1.4) on the left hand side of (1.46), we get

$$\begin{aligned} &\sum_{\boldsymbol{n}\in\mathbb{Z}^s} \varphi(\boldsymbol{x}-\mathbf{L}\boldsymbol{n})(\mathbf{L}\boldsymbol{n})^{\boldsymbol{\alpha}} = \sum_{\boldsymbol{n}\in\mathbb{Z}^s} \psi(\mathbf{L}\boldsymbol{n}) \\ &= \sum_{\boldsymbol{m}\in\mathbb{Z}^s} \hat{\psi}(-\mathbf{L}^{-\mathsf{T}}\boldsymbol{m}) \\ &= \big(\frac{\imath}{2\pi}\big)^{|\boldsymbol{\alpha}|} \sum_{\boldsymbol{m}\in\mathbb{Z}^s} \exp\big(2\pi\imath\boldsymbol{m}^{\mathsf{T}}\mathbf{L}^{-1}\boldsymbol{x}\big)\Big(D^{\boldsymbol{\alpha}}\hat{\varphi}(\mathbf{L}^{-\mathsf{T}}\boldsymbol{m}) + \big(\frac{2\pi}{\imath}\big)^{|\boldsymbol{\alpha}|}\hat{\varphi}(\mathbf{L}^{-\mathsf{T}}\boldsymbol{m})\boldsymbol{x}^{\boldsymbol{\alpha}}\Big) \\ &= \boldsymbol{x}^{\boldsymbol{\alpha}} \ \text{ provided that, } \ \hat{\varphi}(\mathbf{0}) = 1 \ \text{ and } \ D^{\boldsymbol{\alpha}}\hat{\varphi}(\mathbf{L}^{-\mathsf{T}}\boldsymbol{m}) = 0 \ \text{ for } \ \big|\boldsymbol{\alpha}\big| \le (k-1) \ \text{ and } \ \forall \boldsymbol{m} \ne \mathbf{0}. \end{aligned}$$

□

1.5.2 Relationship between Pre/Post-aliasing and the Error Kernel

Prealiasing (smoothing) and postaliasing error metrics are commonly used in computer graphics and visualization to quantify the deviation of a reconstruction kernel from the ideal [ML94, MN88]. Here, we demonstrate that the Fourier error kernel (1.31) conveniently encapsulates these error measures.

Let f be a bandlimited function and let $\hat{f}$ be completely contained inside $\mathcal{V}_{\mathcal{L}^\circ}$. Suppose that f has been critically point-sampled and is approximated so that $f_{\mathsf{app}} \in \mathbb{V}(\mathcal{L},\varphi)$.

From Section 1.2.4, we know that the error $\|f - f_{\mathsf{app}}\|^2$ can be quantified using (1.31) with $\hat{Q}(\boldsymbol{\omega}) = 1$, i.e.

$$\|f - f_{\mathsf{app}}\|^2 = \int_{\mathcal{V}_{\mathcal{L}^\circ}} \big|\hat{f}(\boldsymbol{\omega})\big|^2 E(\boldsymbol{\omega}) d\boldsymbol{\omega},$$

$$\text{where} \quad E(\boldsymbol{\omega}) = 1 - \frac{\big|\hat{\varphi}(\boldsymbol{\omega})\big|^2}{\hat{A}_\varphi(\boldsymbol{\omega})} + \hat{A}_\varphi(\boldsymbol{\omega})\big|1 - \frac{\hat{\varphi}(\boldsymbol{\omega})}{\hat{A}_\varphi(\boldsymbol{\omega})}\big|^2. \tag{1.47}$$

After simplifying (1.47), we obtain

$$\begin{aligned} E(\boldsymbol{\omega}) &= 1 - 2\hat{\varphi}(\boldsymbol{\omega}) + \hat{A}_\varphi(\boldsymbol{\omega}) \\ &= 1 - 2\hat{\varphi}(\boldsymbol{\omega}) + \sum_{\boldsymbol{r} \in \mathcal{L}^\circ} \big|\hat{\varphi}(\boldsymbol{\omega} - \boldsymbol{r})\big|^2 \\ &= 1 - 2\hat{\varphi}(\boldsymbol{\omega}) + \big|\hat{\varphi}(\boldsymbol{\omega})\big|^2 + \sum_{\boldsymbol{r} \in \mathcal{L}^\circ \setminus \{\boldsymbol{0}\}} \big|\hat{\varphi}(\boldsymbol{\omega} - \boldsymbol{r})\big|^2 \\ &= \big|1 - \hat{\varphi}(\boldsymbol{\omega})\big|^2 + \sum_{\boldsymbol{r} \in \mathcal{L}^\circ \setminus \{\boldsymbol{0}\}} \big|\hat{\varphi}(\boldsymbol{\omega} - \boldsymbol{r})\big|^2. \end{aligned}$$

The error is now given by

$$\|f - f_{\mathsf{app}}\|^2 = \int_{\mathcal{V}_{\mathcal{L}^\circ}} \big|\hat{f}(\boldsymbol{\omega})\big|^2 \Big(\underbrace{\big|1 - \hat{\varphi}(\boldsymbol{\omega})\big|^2}_{\text{smoothing}} + \underbrace{\sum_{\boldsymbol{r} \in \mathcal{L}^\circ \setminus \{\boldsymbol{0}\}} \big|\hat{\varphi}(\boldsymbol{\omega} - \boldsymbol{r})\big|^2}_{\text{postaliasing}} \Big) d\boldsymbol{\omega}.$$

Smoothing and postaliasing are immediately recognizable.

Chapter 2

Toward High-Quality Gradient Estimation

2.1 Motivation

Volumetric data, typically given on a discrete lattice, is perceived as a continuous data type and therefore requires the algorithms working on them to model the data as if it were given in a continuous domain. Hence, interpolation and reconstruction are the key aspects of any volumetric manipulation and have a tremendous impact on the quality and efficiency of the underlying visualization task. While there has been a large body of work on interpolation and reconstruction filter design, in many tasks we also need secondary information of the volumetric data, such as histograms for data exploration [KD98], gradients for shading [MMK+98] or higher order gradients for illustrative rendering [KWTM03] and feature detection [KTW06]. One could simply just take an interpolation kernel and consider its analytical derivative as a proper derivative filter. However, this approach is not only computationally inefficient, it also unnecessarily constrains the conditions on accuracy and smoothness. Hence, a separate design of gradient estimation schemes can lead to much better results. Therefore, we will consider the design of gradient estimation schemes in this chapter. In particular, we consider the idea of approximation spaces introduced in the previous chapter (see Section 1.1.2 on page 3), and seek to approximate derivatives in a prescribed shift-invariant space. Our focus is on improved shading, with the assumption that poor gradient estimation may overshadow the performance of a superior scalar data

reconstruction filter, particularly when perceptual metrics are employed. For this reason, we postulate that it is just as important to improve shading as it is to improve the interpolation of the underlying function.

2.2 Related Work

In rendering, we are often concerned with the smoothness of the reconstruction. Towards this end, Möller *et al.* [MMMY97b] provide a general filter design scheme that extends a purely numerical approach based on a Taylor series expansion by incorporating smoothness constraints. In a different work [MMK+98], they contrast two possible approaches to gradient estimation - using a combination of discrete derivative filter with a continuous interpolation filter or simply computing the derivative of the interpolation filter. While in the former case, one has much better control over smoothness and accuracy, the latter case is simply more attractive since it creates the exact gradient of the interpolated function. Here we argue that for rendering applications, it is not paramount to compute the exact derivative of the interpolated function, but it is preferable to compute the derivative that is closest to the *true* underlying function. This allows us to incorporate smoothness constraints as well.

Despite these fundamental insights on function reconstruction, not much work has focused on derivative reconstruction, and to the best of our knowledge, no work has been done on designing proper derivative filters for arbitrary lattices. While the idea of shift-invariant spaces has been exploited in the visualization community to demonstrate the tremendous impact prefiltering can have on both image quality as well as reconstruction accuracy [FAVM09, Csé08b], it remains to be seen how effective prefiltering is for derivative reconstruction. Similarly, for volume visualization tasks, the BCC lattice has been shown to outperform the CC lattice [TMG01, NM02]. However, it is not clear whether the advantages of the BCC lattice extend to gradient reconstruction as well. This chapter addresses this issue too.

Outside the visualization community, there has been some progress in the way of gradient estimation on regular lattices. Brekelmans *et al.* [BDHDH08] derive discrete filters for gradient estimation using Lagrange polynomials on a CC lattice. They also compare different discrete derivative filters in the presence of noise [BDHDH05]. Sun *et al.* [SKZC03] develop a fourth order gradient estimation scheme for the hexagonal lattice in 2D only.

However, assuming a Lagrange polynomial fit for the gradient estimation and using a different basis, for example, tricubic B-splines for data reconstruction may not yield the most optimal surface shading. We address this in our method (Section 2.3 on page 30) where we take the reconstruction kernel into account and derive discrete derivative filters that are optimized for it.

One track of research has focused on designing digital derivative filters in the Fourier domain. These techniques inherently assume an underlying band-limited signal. Most methods have focused on designing filters in 1D where derivative reconstruction corresponds to a multiplication with a unit slope ramp in the frequency domain [BML96, Gos94]. The ideal discrete derivative filter in that case is the IIR sinc' sampled at the grid points. The continuous derivative can then be recovered by using the sinc as an interpolation kernel on the filtered signal. However, most methods seek to recover the derivative at the grid points only for which, a digital filtering solution suffices. Because of the slow decay, sinc' is rarely used in practice and many approximations have been proposed. These approximations proceed by appropriately choosing a design criterion in the frequency domain and then optimizing it to yield either IIR or finite impulse response (FIR) filters in the spatial domain. For example, Dutta Roy and Kumar [DK93] design 1D FIR filters that are maximally linear over a specified frequency band and therefore attempt to match the unit slope ramp as closely as possible within the band. Farid and Simoncelli [FS04] choose the rotation invariance of the gradient operator in higher dimensions as an optimality criterion to design separable FIR filters. We are unaware of any Fourier domain techniques that design non-separable derivative filters for arbitrary sampling lattices in higher dimensions.

In comparison to Fourier domain techniques, our proposed method is different for two main reasons. Firstly, we are interested in volume visualization as a primary application and for that, we need to accurately estimate derivatives everywhere and not just at the sample points. Secondly, we are not tied to the band-limitedness assumption, this allows us to employ approximation theoretic techniques in our design methodology.

2.3 Orthogonal Projections

While Chapter 1 presented an overview of function approximation from lattice measurements in $\mathbb{R}^s$, here we shall focus on the dimension $s = 3$ and restrict attention to real-valued, measurable functions that belong to the Hilbert space $L_2(\mathbb{R}^3)$. However, our design methodology

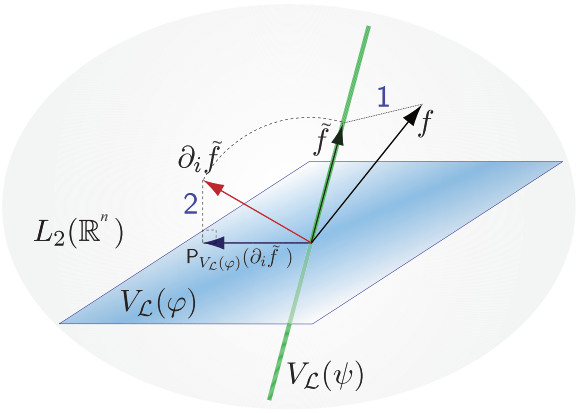


Figure 2.1: An illustration of the two-stage gradient estimation framework.

can be easily generalized to any dimension s.

2.3.1 Gradient Approximation

Typically in visualization applications, a sequence consisting of the sample values of an unknown function f is already given to us which makes the orthogonal projection (see (1.18) on page 9) unrealizable. Our goal is to accurately estimate $\boldsymbol{\nabla} f$, the gradient of f from the given sampled sequence. This can be accomplished using a two-stage procedure as suggested by Unser [Uns95]. In the first stage, we approximate f in an auxiliary approximation space $\mathbb{V}(\mathcal{L}_h, \psi)$ and in the second stage, we orthogonally project the gradient of the approximation of f onto a target approximation space $\mathbb{V}(\mathcal{L}_h, \varphi)$ (Figure 2.1). The generating functions ψ and φ can be chosen according to the needs of the application. When accuracy is of prime importance, ψ should be chosen to have a higher approximation order as compared to φ. On the other hand, when visual quality and efficiency are important, ψ can be chosen so that it has comparable smoothness properties. Note that $\mathbb{V}(\mathcal{L}_h, \psi)$ is an intermediate approximation space which, as we shall soon see, governs the order and size of the discrete derivative filter that is to be applied to the samples of f. The final gradient approximation lies in the target space $\mathbb{V}(\mathcal{L}_h, \varphi)$ where φ plays the role of a reconstruction kernel.

Let $f[\boldsymbol{n}] = f(h\mathbf{L}\boldsymbol{n})$ be the given sampled sequence and let $f_{\mathsf{app}}(\boldsymbol{x}) = \sum_{\boldsymbol{n}} c_1[\boldsymbol{n}]\psi_{h,\boldsymbol{n}}(\boldsymbol{x})$ denote the first-stage approximation of f. Since we do not have any knowledge of the

underlying function f other than its sample values, we consider a design based on consistency, i.e. the coefficient sequence c_1 should be such that the first-stage approximation should be able to exactly interpolate f at the sample locations. In other words, $f_{\mathsf{app}}(h\mathbf{L}\boldsymbol{n}) = \sum_{\boldsymbol{n}} c_1[\boldsymbol{n}]\psi_{h,\boldsymbol{n}}(h\mathbf{L}\boldsymbol{n}) = f[\boldsymbol{n}]$. If the basis functions $\psi_{h,\boldsymbol{n}}$ are interpolating (i.e. $\psi_{h,\boldsymbol{n}}(h\mathbf{L}\boldsymbol{m}) = \delta_{\boldsymbol{n}-\boldsymbol{m}}$), the coefficient sequence c_1 is exactly equivalent to the sampled sequence f. However, if the basis functions are not interpolating, c_1 can be obtained from the sampled sequence f by applying a suitable interpolation prefilter (Section 1.2.3). Denoting this prefilter as $p_1[\cdot] \leftrightarrow \hat{P}_1()$, the first-stage approximation can be written as

$$f_{\mathsf{app}}(\boldsymbol{x}) = \sum_{\boldsymbol{n}} c_1[\boldsymbol{n}]\psi_{h,\boldsymbol{n}}(\boldsymbol{x}) = \sum_{\boldsymbol{n}} (f * p_1)\,[\boldsymbol{n}]\,\psi_{h,\boldsymbol{n}}(\boldsymbol{x}). \tag{2.1}$$

Recall that the prefilter p_1 is given in the Fourier domain (see (1.27) on page 12 and (1.14) on page 8) by

$$\hat{P}_1(\boldsymbol{\omega}) = \frac{1}{\sum_{\boldsymbol{n}} \psi(\mathbf{L}\boldsymbol{n}) \exp\left(-2\pi\imath\boldsymbol{\omega}^{\mathsf{T}}\mathbf{L}\boldsymbol{n}\right)}. \tag{2.2}$$

As discussed earlier, this prefiltering step not only makes the basis functions interpolating, it is also necessary to utilize the full approximation power of the generating function [BU99a].

In the second-stage, we project $\boldsymbol{\nabla} f_{\mathsf{app}}$ onto the target space $\mathbb{V}(\mathcal{L}_h, \varphi)$. This is tantamount to performing three orthogonal projections, one for each component of the gradient. Let $\partial_i f$ denote the partial derivative $\frac{\partial f}{\partial x_i}$, $i \in \{1, 2, 3\}$. Applying the orthogonal projection (1.18) to the gradient of the first-stage approximation (2.1), the second-stage approximation of $\partial_i f$ is given by

$$\begin{aligned} f^i_{\mathsf{app}}(\boldsymbol{x}) &:= \left(\mathsf{P}_{\mathbb{V}(\mathcal{L}_h,\varphi)}\partial_i f_{\mathsf{app}}\right)(\boldsymbol{x}) = \sum_{\boldsymbol{n}} \left\langle \partial_i f_{\mathsf{app}}, \mathring{\varphi}_{h,\boldsymbol{n}} \right\rangle \varphi_{h,\boldsymbol{k}}(\boldsymbol{x}) \\ &= \sum_{\boldsymbol{n},\boldsymbol{m}} c_1[\boldsymbol{m}] \left\langle \partial_i \psi_{h,\boldsymbol{m}}, \mathring{\varphi}_{h,\boldsymbol{n}} \right\rangle \varphi_{h,\boldsymbol{k}}(\boldsymbol{x}) = \sum_{\boldsymbol{n},\boldsymbol{m}} c_1[\boldsymbol{m}] \left\langle \partial_i \psi_h, \mathring{\varphi}_{h,\boldsymbol{n}-\boldsymbol{m}} \right\rangle \varphi_{h,\boldsymbol{k}}(\boldsymbol{x}) \\ &= \sum_{\boldsymbol{n}} (c_1 * \mathring{d}_i)[\boldsymbol{n}] \varphi_{h,\boldsymbol{n}}(\boldsymbol{x}), \end{aligned} \tag{2.3}$$

where $\mathring{d}_i$ is a digital derivative filter given by the inner product

$$\mathring{d}_i[\boldsymbol{n}] := \left\langle \partial_i \psi_h, \mathring{\varphi}_{h,\boldsymbol{n}} \right\rangle. \tag{2.4}$$

2.3.2 Examples

Once ψ and φ have been chosen, the remaining key step in the above scheme is the evaluation of the inner product (2.4) that yields the discrete derivative filter $\mathring{d}_i$. Here we focus on the CC

and BCC lattices and show how the 1D B-splines can be used to design derivative filters that implement the above two-stage approximation scheme. In particular, on the CC lattice, we work with generating functions formed by tensor-product B-splines (Section 1.3.4) and for the BCC lattice, we employ the rhombic dodecahedral box splines (Section 1.3.4) introduced by Entezari *et al.* [EVM08].

Let $\beta^k(x)$ be the centered 1D B-spline of degree k (see (1.39) on page 19) and let $\beta^k_{\triangleright}(x) := \beta^k(x - \frac{k+1}{2})$ be the corresponding non-centered B-spline that is supported in the interval $[0, k+1]$. A very useful property of the B-splines that we shall exploit is that their derivatives can be expressed in terms of lower degree B-splines [UAE93]. In particular,

$$\begin{aligned} \acute{\beta^k}(x) &:= \frac{d\beta^k(x)}{dx} = \beta^{k-1}(x+\tfrac{1}{2}) - \beta^{k-1}(x-\tfrac{1}{2}), \text{ and} \\ \acute{\beta^k_{\triangleright}}(x) &:= \frac{d\beta^k_{\triangleright}(x)}{dx} = \beta^{k-1}_{\triangleright}(x) - \beta^{k-1}_{\triangleright}(x-1). \end{aligned} \tag{2.5}$$

Before delving into the specifics of the CC and BCC lattices, we mention the convolution interpretation of (2.4) that will aid us in our design. Owing to the shift-invariance of the basis functions, the inner product in (2.4) can be seen as a sampled convolution, i.e.

$$\big\langle \partial_i \psi_h, \mathring{\varphi}_{h,\boldsymbol{n}} \big\rangle = (\partial_i \psi_h * \overline{\mathring{\varphi}}_h)(\boldsymbol{x})\big|_{\boldsymbol{x}=h\mathbf{L}\boldsymbol{n}} = \tfrac{1}{h}(\partial_i \psi * \overline{\mathring{\varphi}})(\boldsymbol{x})\big|_{\boldsymbol{x}=\mathbf{L}\boldsymbol{n}}.$$

The latter convolution can be expressed in one of three equivalent forms, $(\partial_i \psi * \overline{\mathring{\varphi}}) = (\psi * \partial_i \overline{\mathring{\varphi}}) = \partial_i(\psi * \overline{\mathring{\varphi}})$, which can be easily verified in the Fourier domain. Furthermore, when $\overline{\mathring{\varphi}}$ is symmetric (i.e. $\overline{\mathring{\varphi}} = \mathring{\varphi}$), as is the case with tensor-product centered B-splines on CC and the rhombic dodecahedral box splines on BCC, this convolution can be simplified by expanding the dual $\mathring{\varphi}$ in terms of the primal basis functions $\{\varphi(\cdot - \mathbf{L}\boldsymbol{n})\}_{\boldsymbol{n}\in\mathbb{Z}^3}$ as given in (1.17) on page 9. The digital derivative filter in (2.4) can then be written as $\mathring{d}_i = (d_i * a_\varphi^{-1})$, where d_i is obtained from the primal φ through

$$d_i[\boldsymbol{n}] := \frac{1}{h}\big(\partial_i(\psi * \varphi)\big)(\boldsymbol{x})\big|_{\boldsymbol{x}=\mathbf{L}\boldsymbol{n}}, \tag{2.6}$$

and $a_\varphi^{-1}[\cdot] \leftrightarrow \frac{1}{\hat{A}_\varphi(\cdot)}$ is obtained from the autocorrelation sequence of φ ((1.15) on page 8).

The orthogonal projection scheme (2.3) that approximates the first partial derivative can now be compactly written as

$$f^i_{\mathsf{app}}(\boldsymbol{x}) = \sum_{\boldsymbol{n}} (p_1 * f * d_i * a_\varphi^{-1})[\boldsymbol{n}]\varphi_{h,\boldsymbol{n}}(\boldsymbol{x}). \tag{2.7}$$

Observe that d_i is an FIR filter whereas p_1 and a_φ^{-1} are IIR filters. In our implementation (Section 2.5), we perform the filtering in the Fourier domain and therefore, do not need to explicitly compute the impulse response of the filters p_1 and a_φ^{-1}. Instead, we make use of the sampled sequence $\psi(\mathbf{L}\boldsymbol{n})$ and the autocorrelation sequence $a_\varphi[\cdot]$ to implement the convolution in the Fourier domain via a simple division.

CC Lattice

We use order $(k+1)$ tensor-product trivariate B-splines $B^k(\boldsymbol{x})$ to design the various digital filters outlined in the previous section. Owing to their separability, the problem boils down to finding the tensor product of 1D filters as shown below.

We choose $\mathbb{V}(\mathbb{Z}_h^3, B^k)$ and $\mathbb{V}(\mathbb{Z}_h^3, B^l)$ as the approximation spaces for the first and second stages respectively. The first stage prefilter is given by the samples of B^k at the lattice sites as given in (2.2). Since the B-splines are symmetric, we use (2.6) to obtain the digital derivative filter. Owing to the fact that the convolution of two B-splines yields another B-spline of a higher degree [UAE93], the CC derivative filter takes the form

$$\begin{aligned} d_i[\boldsymbol{n}] &= \tfrac{1}{h}\big(\partial_i(B^k * B^l)\big)(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{n}} = \tfrac{1}{h}\big(\partial_i B^{k+l+1}\big)(\boldsymbol{n}) \\ &= \tfrac{1}{h}\beta^{k+\acute{l}+1}(n_i)\prod_{j\neq i}\beta^{k+l+1}(n_j). \end{aligned} \tag{2.8}$$

This filter only needs to be evaluated once, derivative filters for other directions are given by appropriate permutations. For instance, if $d_1[\boldsymbol{n}]$ is known, d_2 and d_3 are given by

$$d_2[\boldsymbol{n}] = d_1[n_2, n_1, n_3], \quad \text{and} \quad d_3[\boldsymbol{n}] = d_1[n_3, n_2, n_1] \tag{2.9}$$

respectively.

Lastly, the auto-correlation sequence $a_{B^l}[\boldsymbol{n}]$ is obtained by sampling $(B^l * B^l) = B^{2l+1}$ at the integers. Thanks to separability, it is conveniently given by $a_{B^l}[n_1, n_2, n_3] = \prod_{i=1}^3 \beta^{2l+1}(n_i)$.

BCC Lattice

In this section, we use a scaled version of the BCC lattice generated by the matrix

$$\mathbf{H} := \sqrt[3]{4}\mathbf{B} = \begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix}, \tag{2.10}$$

where $\mathbf{B}$ is as defined in (1.36) on page 16. We denote the generated scaled lattice as $\mathcal{H}$. Recall that with this scaling, the BCC lattice is a sublattice of $\mathbb{Z}^3$ obtained by retaining those points that have the same parity (coordinates are either all odd or all even). The Voronoi cell of each lattice site of $\mathcal{H}$ is a truncated octahedron having a volume of 4.

The four-directional rhombic dodecahedral box splines proposed by Entezari *et al.* [EVM08] (Section 1.3.4 on page 20), generate bases that satisfy the Strang-Fix relations (1.12). Box splines, in general, have various equivalent definitions that make use of the generating direction vectors either in the spatial domain or in the Fourier domain [dHR93]. Here, we follow a somewhat different approach by using the projection interpretation of the rhombic dodecahedral box splines as it allows us to easily extend the B-spline framework to BCC.

A rhombic dodecahedral box spline can also be constructed by projecting a 4D tensor-product B-spline — dilated by a factor of 2 — along the antipodal axis of the supporting tesseract [EVM08]. Let $\boldsymbol{y}$ be the 4D column vector $\boldsymbol{y} := (\boldsymbol{x}, t)^\mathsf{T} = (x_1, x_2, x_3, t)^\mathsf{T}$ and $\mathbf{R}$ be the 4D rotation matrix

$$\mathbf{R} := \frac{1}{2}\,[\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \boldsymbol{r}_4] = \frac{1}{2}\begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

that rotates 4D space so that the antipodal axis of the tesseract is parallel to the t-axis [EVM08]. We can now define the rhombic dodecahedral box spline of order k (for even k) as the projection of a 4D tensor-product function consisting of dilated non-centered B-splines of degree $(k/2-1)$. We write this as

$$\vartheta^k(\boldsymbol{x}) := \frac{1}{4}\int_{t=0}^{2k}\prod_{j=1}^{4}\beta_{\triangleright}^{\frac{k}{2}-1}\left(\frac{1}{4}\boldsymbol{r}_j^\mathsf{T}\boldsymbol{y}\right)dt, \tag{2.11}$$

where $k \in 2\mathbb{Z}_+$, the length of the antipodal diagonal is $2k$, and the leading factor of $1/4$ ensures that the box splines are normalized to have an integral of 4 (volume of the Voronoi cell of $\mathcal{H}$) over their support. This definition of the rhombic dodecahedral box splines is related to the previous definition (see Section 1.3.4) according to the scaling relationship

$$\vartheta^k(\boldsymbol{x}) = M_{\boldsymbol{\Theta}^{k/2}}\left(\frac{\boldsymbol{x}}{\sqrt[3]{4}}\right). \tag{2.12}$$

In contrast to the tensor-product B-splines, k is the approximation order of the box spline rather than the degree of the constituent B-splines. The support of ϑ^k is a rhombic dodecahedron that has its 14 vertices at the lattice sites $(\pm k/2, \pm k/2, \pm k/2)$, $(\pm k, 0, 0)$, $(0, \pm k, 0)$

Table 2.1: Orthogonal projection derivative filters. The filters are grouped according to the approximation order of the first and second stage basis functions. Sizes for the compact (FIR) filter d_i in (2.7), in terms of the number of non-zero filter weights, are shown. The filters are named according to the degree of the polynomials that make up the basis functions; on CC, l-trilinear, c-tricubic and q-triquintic; and on BCC, L-linear, Q-quintic and N-nonic

(a) CC

		φ	
		B^1	B^3
ψ	B^1	ll 18	
	B^3	cl 100	cc 294
	B^5	ql same as cc	qc 648

(b) BCC

		φ	
		ϑ^2	ϑ^4
ψ	ϑ^2	LL 10	
	ϑ^4	QL 52	QQ 150
	ϑ^6	NL same as QQ	NQ 328

and $(0, 0, \pm k)$. Since the B-splines are piecewise polynomials, the integral in (2.11) can be analytically evaluated for arbitrary $\boldsymbol{x} \in \mathbb{R}^3$.

Analogous to the CC case, let us choose the first and second-stage approximation spaces as $\mathbb{V}(\mathcal{H}_h, \vartheta^k)$ and $\mathbb{V}(\mathcal{H}_h, \vartheta^l)$ respectivly. The first-stage prefilter is related to the samples of ϑ^k through (2.2). Like a tensor-product B-spline, ϑ^k is symmetric and can be represented as a convolution of lower order box splines. The BCC derivative filter (2.6) therefore becomes

$$d_i[\boldsymbol{n}] = \tfrac{1}{h}\big(\partial_i(\vartheta^k * \vartheta^l)\big)(\boldsymbol{x})\big|_{\boldsymbol{x}=\mathbf{H}\boldsymbol{n}} = \tfrac{1}{h}\big(\partial_i \vartheta^{k+l}\big)(\mathbf{H}\boldsymbol{n}),$$

which, after using (2.11) and the derivative relation in (2.5), simplifies to

$$d_i[\boldsymbol{n}] = \tfrac{1}{8h}\int_{t=0}^{4(\tilde{k}+1)} \sum_{j=0}^{4} \mathbf{R}_{ji} \beta_{\triangleright}^{\tilde{k}}\left(\tfrac{1}{4}\boldsymbol{r}_j^{\mathsf{T}}\left[\begin{smallmatrix}\mathbf{H}\boldsymbol{n}\\ t\end{smallmatrix}\right]\right) \prod_{m\neq j} \beta_{\triangleright}^{\tilde{k}}\left(\tfrac{1}{4}\boldsymbol{r}_m^{\mathsf{T}}\left[\begin{smallmatrix}\mathbf{H}\boldsymbol{n}\\ t\end{smallmatrix}\right]\right) dt, \tag{2.13}$$

where $\tilde{k} := \frac{k+l}{2} - 1$. The integrand above is also a piecewise polynomial and can be analytically evaluated. It is easy to verify that the permutation relation (2.9) is also applicable here.

Finally, the auto-correlation sequence $a_{\vartheta^l}[\cdot]$ is obtained by sampling the box spline $(\vartheta^l * \vartheta^l) = \vartheta^{2l}$ at the lattice sites of $\mathcal{H}$.

2.4 Applicability Analysis

When working with signals that are either bandlimited or sufficiently over-sampled, the quality of a filter can be characterized in the Fourier domain in terms of its deviation from

the ideal filter. On the Cartesian lattice, it is a common practice to design one-dimensional filters and then extend them to higher dimensions via a tensor product. In this section, we compare the frequency behavior of some of our derivative filters for the CC lattice. To simplify the analysis, we focus on those filters that are to be used in combination with the cubic B-spline (see Table 2.1). A more thorough analysis in terms of a modified Fourier error kernel similar to the scalar error kernel discussed in Section 1.2.4) is presented in Chapter 3.

Derivatives on the higher dimensional Cartesian lattice are usually computed by using a one-dimensional derivative filter along a canonical direction. For such a scenario, it is instructive to compare the frequency profiles of the filters restricted to a univariate setting. Figure 2.2 shows the response of the 1D versions of our orthogonal projection (OP) filters *qc* and *cc* along with the responses of some other digital derivative filters that are combined with the prefiltered cubic B-spline. Dutta Roy and Kumar's filter that is designed to be maximally linear in the pass-band is inferior to the OP filters which have the best pass-band behavior. This is to be expected since the OP filters are optimitized for the space spanned by the cubic B-spline. However, the gain provided by the OP filters comes at the expense of a deteriorated post-aliasing. Estimating the analytical derivative of the reconstructed function via second-order central differencing with an ϵ-step (ϵ-*cd*), fairs similarly in the pass-band but has the worst post-aliasing performance. This attests the fact that taking the analytical derivative of the reconstructed function may not be the best choice.

Since non-separable lattices such as the BCC lattice have been shown to improve scalar reconstruction quality [TMG01, EDM04, MSE+07], we believe that the OP framework when extended to the BCC lattice should improve gradient reconstruction quality as well.

2.5 Results and Discussion

We followed the recipe presented in Section 2.3 to design gradient estimation filters of different orders for both the CC and BCC lattices. A summary of our filters is given in Table 2.1 on page 36. These filters are not compactly supported and therefore not suitable for on-the-fly computations. Practical implementation is feasible when gradients are pre-computed, for example, as three separate gradient volumes (one for each component) with each having the same number of elements as the data volume itself. If all the four volumes (including the data volume) fit into memory then using these filters will not only yield the most accurate gradients but will also be efficient; as for every sample we just need to perform

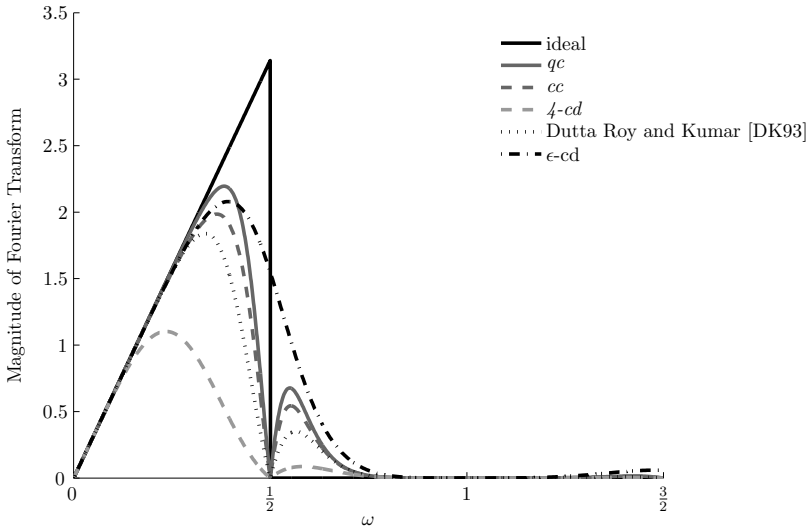


Figure 2.2: Frequency response of various 1D derivative filters used in combination with the cubic B-spline. The fourth-order FIR filter (4 non-zero weights) *4-cd* [HAM11] (also see Section 2.6) and the 14 non-zero weight filter of Dutta Roy and Kumar [DK93] (designed to be maximally linear in the pass band) are combined with the prefiltered cubic B-spline. ϵ-*cd* is the analytical derivative of the prefiltered cubic B-spline.

four interpolations (one for the data and three for the gradient). Data streaming techniques can be employed when storage space is not the limiting factor but RAM is.

In order to efficiently implement the discrete convolution operation in a preprocessing stage, we employed the multidimensional discrete Fourier transform (MDFT) to yield a gradient volume. On the CC lattice, the MDFT can be evaluated using a tensor product FFT [FJ05]. The MDFT on the BCC lattice is non-separable, but can still be efficiently evaluated using the FFT as explained in Chapter 5. We also used the MDFT to prefilter scalar data when interpolating with either the tricubic B-splines on CC or the quintic box spline on BCC in order to fully exploit the approximation power. The cost of this prefiltering step is negligible as compared to the cost of the subsequent rendering operations.

2.5.1 Implementation

We evaluated ray-casting [HJ05] integrals in an iso-surface rendering (ISR) mode. A given iso-surface is extracted along a ray in the volume using a linear bisection technique and once the iso-surface is found, the gradient is estimated at that point and shaded accordingly. Only scalar interpolation is performed during the iso-surface extraction stage. Keeping

the underlying interpolation filter the same allows us to investigate how the quality of the rendered images changes as a result of different gradient estimation schemes.

We implemented our volume ray-caster as a single threaded application and ran all our experiments on an Intel Core$^{\text{TM}}$2 Duo (2.40GHz on each core) machine with 4GB RAM running Linux. We also optimized our codes using the following compiler (GCC version 4.4.1) optimization flags:
`-march=core2 -O6 -ffast-math -funroll-all-loops -ftree-vectorize`.

2.5.2 Synthetic Data

We used the popular synthetic function proposed by ML [ML94]. This function has a form that allows one to correctly point sample it so as to ensure that there is no aliasing of the spectrum in the frequency domain. In practice however, one may have little to no knowledge of the underlying frequency content of a sampled signal, in which case, a reconstruction that attempts to minimize the L_2 norm of the error is a more desirable one. Furthermore, isosurfaces of the ML function are not closed manifolds and error is introduced near the boundaries of the sampling window since data outside the window needs to be fetched to accurately reconstruct the function or its gradient. For these reasons, we also employed an appropriately modified version of the ML function so that the isosurfaces are closed manifolds that radiate spherically outwards with increasing isovalue (Figure 2.3). The resulting function can be written in Cartesian coordinates as

$$f_{\mathsf{test}}(\boldsymbol{x}) := \gamma\|\boldsymbol{x}\| - \alpha\cos(2\pi f_m \frac{x_3}{\|\boldsymbol{x}\|}), \tag{2.14}$$

where $\boldsymbol{x} \in \mathbb{R}^3$ ($\boldsymbol{x} \neq 0$) and γ, α and f_m are positive real parameters. The cosine frequency modulation form akin to the ML function can be obtained by expressing the above equation in spherical coordinates. As $\boldsymbol{x} \to \mathbf{0}$, the oscillation frequency of this function tends to infinity. Thus, for any finite sampling rate, one can always choose an isosurface that would be a demanding test for any reconstruction filter.

We point sampled both the ML function and f_{test} within a $(-1, 1)^3$ window on CC and BCC lattices. For the ML function, we used the parameters given in [ML94] and sampled the function on a $41 \times 41 \times 41$ CC grid and on an equivalent $32 \times 32 \times 64$ BCC grid. For f_{test}, we used the parameters shown in Figure 2.3 and sampled it on CC and BCC grid sizes of $101 \times 101 \times 101$ and $80 \times 80 \times 160$ respectively.

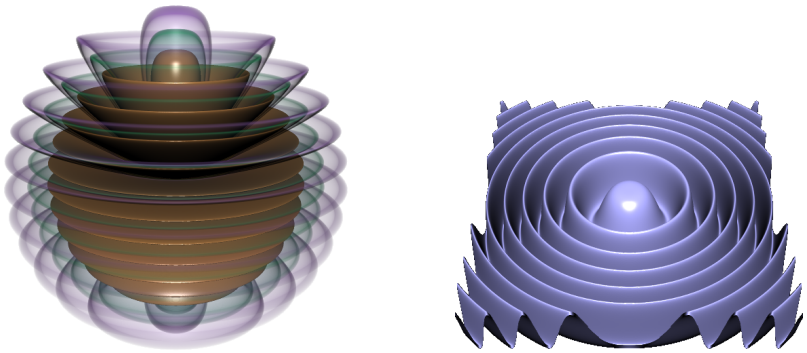
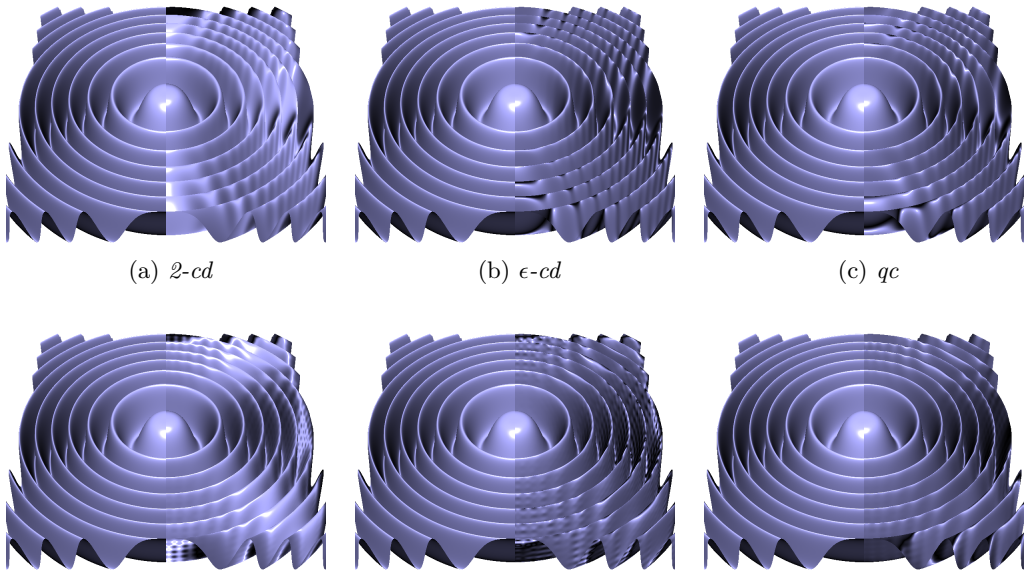


Figure 2.3: Isosurfaces of the unsampled synthetic functions. Left, the modified test function, $\alpha = 0.25$, $\gamma = 2$ and $f_m = 6$, showing the isovalues 0.4 (rendered opaque), 0.5 (green) and 0.6 (purple). Right, the ML test function, isovalue = 0.5 and other parameters as given in [ML94].

We performed ISR experiments using both test functions. For the ML function, we chose an isovalue of 0.5 whereas for f_{test}, we chose an isovalue of 0.4. We used the analytic form of the functions to compute isosurface intersections and used the sampled versions solely for normal estimation. This ensures that the underlying shape of the isosurface is the same for both lattice types. For the OP filters, the stored gradient volume was used to interpolate gradients at non-grid points. The interpolation filter used is governed by the basis function used in the second stage as indicated by the second letter of the filter name (Table 2.1).

Figure 2.4 shows the isosurface of the ML function shaded with different gradient estimation schemes. The second-order compact filters *2-cd* and *BCD* stand for second-order central differencing (on CC) and box central differencing (on BCC) [HAM11] (see also Section 2.6), and are combined with the prefiltered tricubic B-spline and the prefiltered quintic box spline respectively. Since these filters are compact, they are implemented on the fly without the need to precompute a gradient volume. Since they are second-order filters, they do not fully exploit the approximation capabilities of the reconstruction space and therefore, exhibit undesirable artefacts. The terms ϵ-*cd* (on CC) and ϵ-*CD* (on BCC) refer to estimating the gradient locally at the point of intersection by computing the gradient of the interpolated function using central differences. The superiority of the BCC lattice is clearly evident; the BCC filters *BCD* and *NQ* do a better job at reconstructing the normals than their CC counterparts *2-cd* and *qc*. Additionally, we observe that ϵ central differencing yields better normal estimates as compared to the second order filters *2-cd* and *BCD*. However, ϵ-*CD* on BCC gives rise to rippling artifacts which are absent in ϵ-*cd* on CC. The OP



(a) *2-cd* (b) ϵ-*cd* (c) *qc*

(d) *BCD* (e) ϵ-*CD* (f) *NQ*

Figure 2.4: Isosurface of the ML function shaded using different normal estimation schemes; top row, CC, and bottom row, BCC. The analytic form of the ML function is used to compute the isosurface and the sampled data is used for normal estimation. To facilitate comparison, the left half of each image shows the truth. For (b) and (e), $\epsilon = 0.003$.

filters *qc* and *NQ* outperform the other filters in their respective categories. With *NQ*, the artifacts introduced by ϵ-*CD* are removed and the appearance of the isosurface is closest to the truth.

We also used the test functions to quantify the performance of the filters and measured the RMS l_2-norm of the difference between the true gradient and the estimated gradient, as well as the RMS angular deviation from the truth, on the visible isosurface. The numerical results are shown Table 2.2 and some of the isosurface renderings of f_{test} along with the error distributions are shown in Figure 2.5. Our numerical results corroborate the fact that the advantages of BCC sampling extend to gradient reconstruction as well. The BCC filters yield significantly lower RMS error values as compared to their CC cousins. We observe the same trend quantitatively that we qualitatively saw in Figure 2.4; epsilon central differencing is better than the second-order filters, and the high quality OP filters are comparable in accuracy to epsilon central differencing. On the BCC side however, orthogonal projection seems to have a clear advantage which is further substantiated by the corresponding images

Table 2.2: RMS length of the error vector (l) and RMS angular deviation (θ in degrees) on the visible isosurface. The comparison is performed on the 0.4 isosurface of f_{test} and the 0.5 isosurface of ML. For f_{test}, $\epsilon = 0.005$ and for ML, $\epsilon = 0.003$. The compact fourth-order filters *4-cd* and *OPT16* [HAM11] are combined with the prefiltered tricubic B-spline and the prefiltered quintic box spline respectively. All images were rendered at a resolution of 800×800 pixels.

(a) CC

	f_{test}		ML	
	l	θ	l	θ
4-cd	17.9	31.8	2.79	44.1
ϵ-cd	11.9	22.4	1.52	25.7
ll	18.6	34.1	2.76	32.4
cl	17.6	32.2	2.20	29.8
ql	17.4	31.6	2.09	29.9
cc	16.1	27.7	1.78	23.8
qc	15.9	27.6	1.69	24.5

(b) BCC

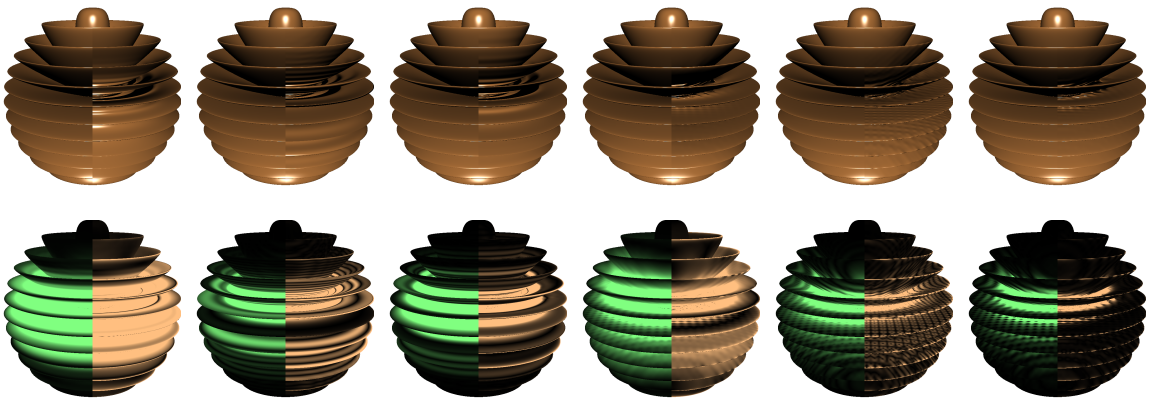
	f_{test}		ML	
	l	θ	l	θ
OPT16	13.7	17.0	2.42	28.7
ϵ-CD	9.8	12.0	1.61	26.1
LL	15.1	22.9	2.50	29.3
QL	13.1	22.8	1.91	28.0
NL	12.5	22.9	1.80	29.1
QQ	10.5	13.0	1.38	19.4
NQ	9.7	12.5	1.29	21.9

in Figure 2.5. We see no rippling artifacts in *QQ* and the error image is mostly black.

2.5.3 Real Data

To assess the practical impact of our filters on the visualization of volumetric data, we rendered isosurface images of the carp and bunny data sets in ISR mode. The original CC data sets have grid sizes of $512 \times 512 \times 512$ and $512 \times 512 \times 361$ respectively. Figure 2.6 depicts the original high resolution carp's skull reconstructed with prefiltered tricubic B-spline interpolation and shaded using the OP filter *cc* in conjunction with the tricubic B-spline. Figure 2.7 illustrates the impact of various fourth-order schemes on a downsampled version of the data set. The top row shows the carp's skull reconstructed using prefiltered tricubic B-spline scalar interpolation on a downsampled CC grid and shaded using three different gradient estimation schemes. The second row analogously shows the results for a downsampled BCC grid. The images follow the same trend as that in Figure 2.4. The rippling artifacts that we observed in the case *ϵ-CD* earlier, can be seen here as well. As before, the OP filters (*qc* and *NQ*) reveal the lost features by strongly enhancing contrast and high frequency details.

Finally, Figure 2.8 illustrates the result of combining the normal estimation schemes with second-order trilinear interpolation. We used the high resolution CC bunny data set for this



(a) *4-cd* (b) ϵ-*cd* (c) *cc* (d) *OPT16* (e) ϵ-*CD* (f) *QQ*

Figure 2.5: Reconstructed isosurface of f_{test} shaded using different gradient estimation schemes. The top row shows the rendered images as compared to the truth while the bottom row shows the corresponding error images. The left half of an error image illustrates the l_2-norm of the error vector where a value of 15 or more is mapped to the brightest green. The right half illustrates the angular deviation where an angle of 15 degrees or more is mapped to the brightest orange. For (b) and (e), $\epsilon = 0.005$. *OPT16* is a fourth-order finite-differencing filter that makes use of 16 neighbors on the BCC lattice [HAM11].

purpose. In comparison to *2-cd*, the OP filters *ll* and *ql* enhance the details significantly especially in high frequency regions. Subtle features on the surface of the bunny which are smoothed out in the *2-cd* rendition, are more clearly visible. At the same time however, ringing artifacts due to an imperfect CT reconstruction are also appreciably enhanced. This suggests that the higher-order OP filters preserve high frequency details and should therefore be used with caution in the presence of noise.

2.6 Notes

Material presented in this chapter first appeared in our earlier work (see [HAM11]), where we also undertake the problem of designing FIR filters in the spatial domain using a Taylor series framework. The discrete derivative filters of Brekelmans *et al.* [BDHDH08] happen to be particular solutions of the general solution space in this framework.

For comparison purposes, we have presented the experimental results obtained using some of the filters obtained using our Taylor series framework. These FIR filters are suitable for on-the-fly evaluations. However, this gain in efficiency comes at the expense of reduced

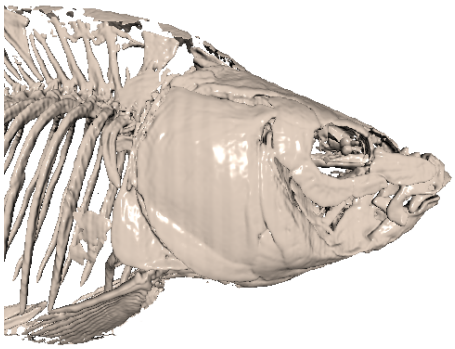
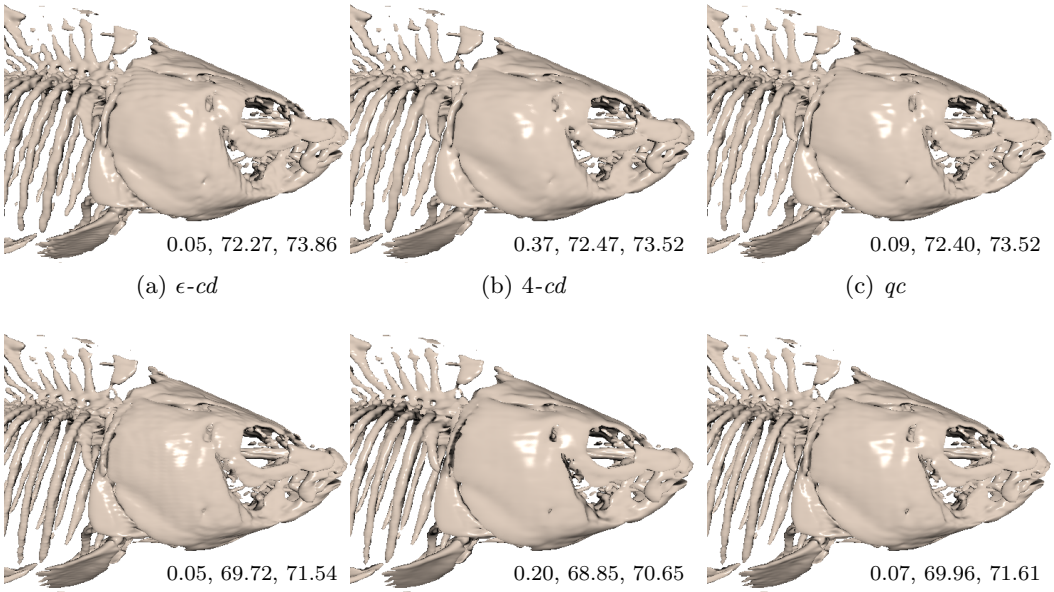


Figure 2.6: An isosurface of the high resolution carp fish data set.

accuracy. The fourth-order filter *4-cd* used in conjunction with the prefiltered tricubic B-spline on the CC lattice has also been investigated by Csébfalvi and Domonkos [CD09].

With compiler optimizations turned on, we observed that quintic box spline evaluation is fairly similar — if not sometimes marginally faster — to that of tricubic B-spline on Intel Core$^{\text{TM}}$2 Duo. However, on AMD Opteron$^{\text{TM}}$, with the same compiler optimizations, we noticed that quintic box spline evaluation is usually marginally slower than tricubic B-spline evaluation. On the other hand, with no compiler optimizations, we observed that this fact is quite the opposite and BCC performs twice faster than CC on both platforms as initially reported by Entezari *et al.* [EDM04].

To access *Mathematica* notebooks that compute the weights for the filters presented in Table 2.1, please see [HAM11, Supplementary Material].



(a) ϵ-*cd* (b) 4-*cd* (c) *qc*

(d) ϵ-*CD* (e) *OPT16* (f) *NQ*

Figure 2.7: Carp data set downsampled to a $160 \times 160 \times 160$ CC grid (a-c) and a $126 \times 126 \times 252$ BCC grid (d-f) and prefiltered appropriately for interpolation filters on the respective grids. Isosurface reconstructed and shaded using tricubic B-spline interpolation on CC and quintic box spline interpolation on BCC. The timing data (in seconds) indicates the normal computation time, the scalar interpolation time and the total render time respectively. All images were rendered at a resolution of 512×512.

(a) *2-cd* (b) *ll* (c) *ql*

Figure 2.8: An isosurface of the high resolution bunny data set. Trilinear interpolation is used for both the scalar data and the gradient.

Chapter 3

Gradient Estimation Revitalized

3.1 Motivation

One of the crucial aspects of volume rendering is the accuracy of normal computation since it will mask the appearance of a bad interpolation filter [MMMY97a]. Even with a good underlying scalar interpolation filter, image quality is very sensitive to the type of normal computation scheme used. Yet, the computation of proper normals remains a very difficult issue, as we saw in the previous chapter where *average* angular errors of 20 to 30 degrees were not uncommon. Therefore, it is crucial to find more *accurate* and *practical* ways of estimating the normal for rendering applications. In this chapter, we shall address both of these competing goals.

Like our approach in Chapter 2, we employ a direct methodology in which a finite-difference-like digital filter is combined with a continuous reconstruction kernel. This is preferable, since all the degrees of freedom can be exploited towards a good estimation of the *true* underlying derivative, without being constrained by the reconstruction of the function itself. Hence, a reconstruction space for the derivative can be specified independent of the way the scalar function is interpolated. Once the reconstruction space for the derivative is chosen, e.g. a spline-like space (Section 1.3.3 on page 17), the scheme which determines a particular continuous function in this space from the available point samples has to be designed. To that end, a relevant viewpoint to analyse reconstruction is offered by approximation theory. We focus on the approximation order of derivative reconstruction schemes that governs how the error behaves asymptotically (Section 1.2). To fully exploit the approximation power of a given reconstruction space, a correction prefilter has to be applied

to the data (see Section 1.2.3 on page 11). The interest of prefiltering for the visualization community has been recognized [Csé08b, FAVM09] but has not been transposed to derivative reconstruction so far. A notable exception is the work of Csébfalvi and Domonkos [CD09] in which they propose FIR derivative prefilters designed to fully exploit the approximation power of the reconstruction space. Our motivation is similar, however, we depart from existing approaches to design IIR prefilters with specific properties, either within the OP framework (Chapter 2) or by designing combinations of interpolation prefilters and finite differences. Our methodology is generic in the sense that it can be deployed on arbitrary lattices. It is now a wide-spread result that the BCC lattice outperforms the CC lattice for visualization tasks [MSE+07, EDM04, NM02, TMG01].

Hence, our main contribution in this chapter is the demonstration that, when reconstructing the gradient continuously in appropriate shift-invariant spaces with specific prefilters, we can obtain normals whose accuracy goes beyond previously known limits. This quality comes without increase in the computational burden! Even more, the efforts for obtaining a good normal estimation provide a good function reconstruction at no additional cost. Incidentally, we break the common belief that the best gradient is obtained by computing the analytic partial derivatives of the reconstructed function.

Our tool of choice is the recently developed extension of the error kernel of Blu and Unser [BU99a] (Section 1.2.4 on page 14) that provides a way to accurately quantify the error between the reconstructed gradient and its true underlying counterpart [CM11].

3.2 Shifted Reconstruction Kernel

The overall quality of the two-stage gradient approximation scheme (Chapter 2) is governed by the approximation properties of the spaces $\mathbb{V}(\mathcal{L}_h, \psi)$ and $\mathbb{V}(\mathcal{L}_h, \varphi)$. In Chapter 2, we considered the case where all the first-stage derivatives $\partial_i f_{\mathsf{app}}$ in the axis aligned directions are orthogonally projected to the same target space $\mathbb{V}(\mathcal{L}_h, \varphi)$ generated by a single reconstruction function φ. The target space is chosen so that it fulfills the regularity and accuracy demands of the application. With the target space fixed, the first-stage auxiliary space should be chosen to have a higher approximation order so that the gradient of the first-stage approximation $\boldsymbol{\nabla} f_{\mathsf{app}}(\boldsymbol{x})$ is close to the true gradient $\boldsymbol{\nabla} f(\boldsymbol{x})$.

Using the same space to approximate the function as well as the gradient components is an attractive design choice from a computational point of view since the same scalar

interpolation routines can be reused to interpolate the gradient as well. In general however, other choices are possible both in terms of the directions one chooses to compute derivatives along, as well as the target space in which each directional derivative is approximated. Towards this end, let $\partial_{\vec{\boldsymbol{u}}} f$ denote the directional derivative of f in the direction $\vec{\boldsymbol{u}}$. Let $\varphi_i(\boldsymbol{x})$ be the reconstruction function of the target space $\mathbb{V}(\mathcal{L}_h, \varphi_i)$ onto which $\partial_{\vec{\boldsymbol{u}}_i} f_{\mathsf{app}}$ is projected. Our goal is to choose each target space $\mathbb{V}(\mathcal{L}_h, \varphi_i)$ such that it minimizes the orthogonal projection L_2 error $\|\partial_{\vec{\boldsymbol{u}}_i} f_{\mathsf{app}} - \big(\mathsf{P}_{\mathbb{V}(\mathcal{L}_h,\varphi_i)} \partial_{\vec{\boldsymbol{u}}_i} f_{\mathsf{app}}\big)\|$. Note that with this modification, the ideal scenario that yields zero error, i.e. when $\varphi_i(\boldsymbol{x})$ is chosen such that $\partial_{\vec{\boldsymbol{u}}_i} f_{\mathsf{app}} \in \mathbb{V}(\mathcal{L}_h, \varphi_i)$, can be easily incorporated in the framework, thus yielding the exact directional derivative $\partial_{\vec{\boldsymbol{u}}_i} f_{\mathsf{app}}$ of the first-stage approximation f_{app}.

The prospect of finding separate reconstruction spaces $\mathbb{V}(\mathcal{L}_h, \varphi_i)$ is an ambitious one and may not be practically all that advantageous as it would require each component of the gradient to be reconstructed with a different reconstruction function $\varphi_i(\boldsymbol{x})$. However, if we choose reconstruction functions from the same family for both the first and second stages, we may be able to exploit the derivative relationships that exist between the two functions. In that case, the problem of finding separate functions $\varphi_i(\boldsymbol{x})$ can be reduced to finding appropriate shifts of a symmetric reconstruction function $\varphi(\boldsymbol{x})$.

The above idea is best explained with an example where we use the order-$(k+1)$ tensor-product B-spline $B^k(\boldsymbol{x})$ (Section 1.3.4 on page 19) as the first stage reconstruction function on the s-dimensional Cartesian lattice $\mathbb{Z}^s$. The first-stage approximation $f_{\mathsf{app}}(\boldsymbol{x})$ lies in the space $\mathbb{V}(\mathbb{Z}^s, B^k)$. Using the derivative relationship of the centered B-splines (see (2.5) on page 33), it is straightforward to show that $\partial_i f_{\mathsf{app}} \in \mathbb{V}(\mathbb{Z}^s, \varphi_i)$, where $\varphi_i(\boldsymbol{x})$ is the ideal second-stage reconstruction function for the i-th partial derivative and is given by

$$\varphi_i(\boldsymbol{x}) = \beta^{k-1}(x_i - \tfrac{1}{2}) \prod_{j \neq i} \beta^k(x_j). \tag{3.1}$$

Instead of the ideal functions $\varphi_i(\boldsymbol{x})$, let us choose the second-stage reconstruction functions to be

$$B_i^m(\boldsymbol{x}) := \beta^m(x_i - \tfrac{1}{2}) \prod_{j \neq i} \beta^m(x_j) = B^m(\boldsymbol{x} - \tfrac{1}{2}\vec{\boldsymbol{e}}_i), \tag{3.2}$$

where $m \leq k$ and $\vec{\boldsymbol{e}}_i$ represents the unit vector in the i-th Cartesian direction. Thus, $B_i^m(\boldsymbol{x})$ is merely a shifted version of the centered function $B^m(\boldsymbol{x})$ (Figure 3.1a). With this choice, we conjecture that we obtain a better approximation scheme as compared to our previous scheme (Section 2.3) that uses the same symmetric function $B^m(\boldsymbol{x})$ to approximate all the

gradient components. The shifts ensure that the reconstruction functions remain close to the ideal. Additionally, they are easy to incorporate into existing interpolation routines as they are cheaply computed from the same symmetric function $B^m(\boldsymbol{x})$ simply by shifting the point at which interpolation is to be perfomed by $-\frac{1}{2}$ in the direction of the derivative.

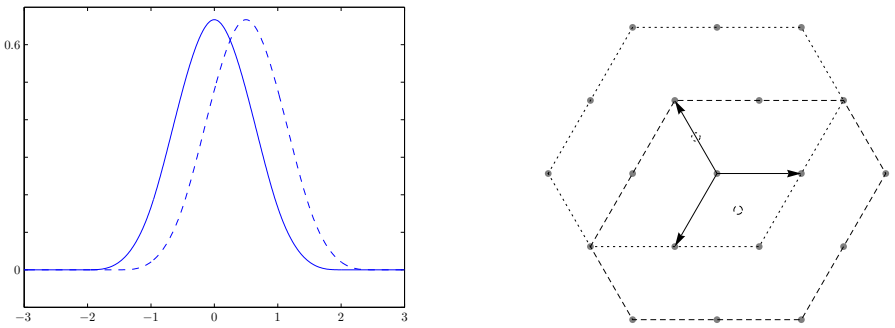


Figure 3.1: (a) 1D illustration of the shifted derivative estimation scheme. Instead of using a centered kernel, e.g. a cubic B-spline $\beta^3(x)$ (solid), to reconstruct the derivative, we propose to use the shifted version $\beta^3(x-\frac{1}{2})$ (dashed). (b) Box splines on the hexagonal lattice are generated by three direction vectors (indicated as arrows). The support of a fourth-order box spline is a hexagon formed by the second-nearest neighbors. The directional derivative of this box spline along a convolution direction is a linear combination of two lower-order box splines that are shifted along the convolution direction as illustrated.

The box spline $M_{\boldsymbol{\Xi}}(\boldsymbol{x})$ (Section 1.3.4 on page 18) also exhibits an analogous derivative relationship [dHR93]. If $\boldsymbol{\xi} \in \boldsymbol{\Xi}$, then the directional derivative $\partial_{\boldsymbol{\xi}} M_{\boldsymbol{\Xi}}$ is given by

$$\partial_{\boldsymbol{\xi}} M_{\boldsymbol{\Xi}}(\boldsymbol{x}) = M_{\boldsymbol{\Xi}\backslash\boldsymbol{\xi}}(\boldsymbol{x}) - M_{\boldsymbol{\Xi}\backslash\boldsymbol{\xi}}(\boldsymbol{x}-\boldsymbol{\xi}), \tag{3.3}$$

where $\boldsymbol{\Xi}\backslash\boldsymbol{\xi}$ is the matrix obtained by removing one occurrence of $\boldsymbol{\xi}$ from $\boldsymbol{\Xi}$. The directional derivative is therefore obtained by the backward difference of two lower-order box splines. If the box spline is symmetric (i.e. $\sum_{\boldsymbol{\eta}\in\boldsymbol{\Xi}} \boldsymbol{\eta} = 0$), then the lower-order box spline thus obtained is shifted in the direction $\boldsymbol{\xi}$ as illustrated in Figure 3.1b for the hexagonal lattice in 2D. Thus, when working with symmetric box splines, we argue that we can obtain a better approximation scheme if instead of approximating a partial derivative in an axis-aligned direction using symmetric box splines, we approximate directional derivatives using the same box spline shifted along the direction of the derivative.

We formalize this notion in the following section by quantitatively analyzing the L_2 error incurred as a result of choosing a shifted second-stage reconstruction function. Hereinafter,

we collectively refer to both of these gradient estimation strategies as the OP framework.

3.3 Fourier-Domain Error Quantification

3.3.1 Scalar and Derivative Error Kernels

In order to quantitatively assess the error behavior of the OP derivative approximation framework, we propose to use the error kernel of Blu and Unser [BU99a] (1.31). Condat and Möller [CM11] have recently extended this result to the Fourier-domain error quantification of one-dimensional derivatives of any order. Their result can be easily applied to arbitrary sampling lattices in higher dimensions to yield the following Fourier-domain derivative error kernel:

$$E^{\boldsymbol{l}}(\boldsymbol{\omega}) := E_{\mathsf{min}}(\boldsymbol{\omega}) + \underbrace{\hat{A}_{\varphi}(\boldsymbol{\omega})\Big|\frac{\hat{D}(\boldsymbol{\omega})}{2\pi\imath\boldsymbol{l}^{\mathsf{T}}\boldsymbol{\omega}} - \hat{\check{\varphi}}^{\star}(\boldsymbol{\omega})\Big|^2}_{E^{\boldsymbol{l}}_{\mathsf{res}}(\boldsymbol{\omega})}. \tag{3.4}$$

where $E_{\mathsf{min}}(\boldsymbol{\omega})$ is as defined in (1.31) on page 14, $\boldsymbol{l}$ is a principal direction of the lattice $\mathcal{L}$ and $\hat{D} \leftrightarrow d$ is a discrete filter that is to be applied to the samples of f to yield the directional derivative approximation:

$$\partial_{\boldsymbol{l}} f(\boldsymbol{x}) \approx f^{\boldsymbol{l}}_{\mathsf{app}}(\boldsymbol{x}) = \sum_{\boldsymbol{n}\in\mathbb{Z}^d} \frac{1}{h}(f * d)[\boldsymbol{n}]\varphi_{h,\boldsymbol{n}}(\boldsymbol{x}). \tag{3.5}$$

The averaged L_2 error $\|\partial_{\boldsymbol{l}} f - f^{\boldsymbol{l}}_{\mathsf{app}}\|$ can be predicted — in a manner similar to (1.32) — according to

$$\epsilon_{f^{\boldsymbol{l}}}(h) := 2\pi\sqrt{\int_{\mathbb{R}^s} \big|\hat{f}(\boldsymbol{\omega})\imath\boldsymbol{l}^{\mathsf{T}}\boldsymbol{\omega}\big|^2 E^{\boldsymbol{l}}(h\boldsymbol{\omega})d\boldsymbol{\omega}}. \tag{3.6}$$

The derivative error kernel (3.4) has an algebraic form that is very similar to the scalar error kernel (1.31) on page 14. It is also bounded below by $E_{\mathsf{min}}(\boldsymbol{\omega})$ which suggests that the derivative approximation error can never be lower than the minimum scalar orthogonal projection error. The term $E^{\boldsymbol{l}}_{\mathsf{res}}(\boldsymbol{\omega})$ can be interpreted as first undoing the directional derivative operation performed by the filter d effectively yielding an approximation of the original function f, and then measuring the deviation from the orthogonal projection.

Since we are dealing with point samples, the minimum error approximation scenario is not realizable. However, for functions that have most of their spectral power contained in the vicinity of $\boldsymbol{\omega} = 0$, we can still achieve a similar asymptotic error behavior if the filter d is chosen appropriately. If φ is a k-th order reconstruction function, then the minimum

error kernel satisfies $E_{\mathsf{min}}(\boldsymbol{\omega}) = O(\|\boldsymbol{\omega}\|^{2k})$ (Section 1.2.4). Thus, in order to ensure that $f^{\boldsymbol{l}}_{\mathsf{app}}$ provides a k-th order approximation of $\partial_{\boldsymbol{l}} f$, we require that $E^{\boldsymbol{l}}_{\mathsf{res}}(\boldsymbol{\omega}) = O(\|\boldsymbol{\omega}\|^{2k})$ [BU99a, CM11]. In other words, the derivative filter d should be chosen so that the approximation scheme matches the orthogonal projection as closely as possible. This boils down to requiring that

$$\frac{\hat{D}(\boldsymbol{\omega})}{2\pi\imath \boldsymbol{l}^{\mathsf{T}}\boldsymbol{\omega}} = \hat{\mathring{\varphi}}^{\star}(\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^{k}), \;\; \text{or equivalently, } \; \hat{D}(\boldsymbol{\omega}) = 2\pi\imath \boldsymbol{l}^{\mathsf{T}}\boldsymbol{\omega}\hat{\mathring{\varphi}}^{\star}(\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^{k+1}). \quad (3.7)$$

3.3.2 Assessment of the two-stage OP framework

We restrict attention to the case where derivatives are taken along the principal lattice directions only. For a symmetric first-stage reconstruction function ψ of the B-spline or box spline type, the directional derivative $\partial_{\boldsymbol{l}_i}\psi$ in the lattice direction $\boldsymbol{l}_i$ ($i \in \{1, 2, \ldots, s\}$) is given by the backward difference of two lower-order spline functions that are centered about the points $\frac{-\boldsymbol{l}_i}{2}$ and $\frac{\boldsymbol{l}_i}{2}$ respectively. We therefore choose the second-stage reconstruction function to be $\varphi_i(\boldsymbol{x}) := \varphi(\boldsymbol{x} - \frac{\boldsymbol{l}_i}{2})$, where φ is a k-th order function of the spline variety that is symmetric about the origin. We then orthogonally project the directional derivative of the first-stage approximation onto the space $\mathbb{V}(\mathcal{L}_h, \varphi_i)$.

It is straightforward to verify that the shift carries over to the duals, i.e.

$$\mathring{\varphi}_i(\boldsymbol{x}) = \mathring{\varphi}(\boldsymbol{x} - \tfrac{\boldsymbol{l}_i}{2}) \quad \leftrightarrow \quad \hat{\mathring{\varphi}}_i(\boldsymbol{\omega}) := \frac{\hat{\varphi}(\boldsymbol{\omega})}{\hat{A}_{\varphi}(\boldsymbol{\omega})} \exp(-\pi\imath \boldsymbol{l}_i{}^{\mathsf{T}}\boldsymbol{\omega}). \quad (3.8)$$

Consequently, the digital derivative filter (2.4), now takes the form

$$\begin{aligned} \mathring{d}_i[\boldsymbol{n}] = \big\langle \partial_{\boldsymbol{l}_i}\psi, \mathring{\varphi}_i(\cdot - \mathbf{L}\boldsymbol{n})\big\rangle &= \big(\partial_{\boldsymbol{l}_i}(\psi * \overline{\mathring{\varphi}_i})\big)(\boldsymbol{x})\big|_{\boldsymbol{x}=\mathbf{L}\boldsymbol{n}} \\ &= \big(\partial_{\boldsymbol{l}_i}(\psi * \mathring{\varphi})\big)(\boldsymbol{x})\big|_{\boldsymbol{x}=\mathbf{L}\boldsymbol{n}+\boldsymbol{l}_i/2}. \end{aligned} \quad (3.9)$$

By expressing the dual $\mathring{\varphi}(\boldsymbol{x})$ in terms of a linear combination of the primal functions $\varphi_{1,\boldsymbol{n}}(\boldsymbol{x})$ (see (1.17) on page 9), we can write this as

$$\begin{gathered} \mathring{d}_i[\boldsymbol{n}] = (\delta_i * a_{\varphi}^{-1})[\boldsymbol{n}], \\ \text{where } \; \delta_i[\boldsymbol{n}] = \big(\partial_{\boldsymbol{l}_i}(\psi * \varphi)\big)(\boldsymbol{x})\big|_{\boldsymbol{x}=\mathbf{L}\boldsymbol{n}+\boldsymbol{l}_i/2} \; \text{ and } \; a_{\varphi}^{-1}[\boldsymbol{n}] \leftrightarrow 1/\hat{A}_{\varphi}(\boldsymbol{\omega}). \end{gathered} \quad (3.10)$$

The combined OP directional derivative filter D_i that is to be applied to the point samples

of f (cf. (2.1) and (2.3)) is then given by

$$\begin{aligned} D_i[\boldsymbol{n}] := (p_1 * \mathring{d}_i)[\boldsymbol{n}] \quad \leftrightarrow \quad \hat{D}_i(\boldsymbol{\omega}) &= \frac{\sum_{\boldsymbol{n}} \mathring{d}_i[\boldsymbol{n}] \exp(-2\pi\imath\boldsymbol{\omega}^{\mathsf{T}}\mathbf{L}\boldsymbol{n})}{\sum_{\boldsymbol{n}} \psi(\mathbf{L}\boldsymbol{n}) \exp(-2\pi\imath\boldsymbol{\omega}^{\mathsf{T}}\mathbf{L}\boldsymbol{n})} \\ &= \frac{\sum_{\boldsymbol{r}\in\mathcal{L}^\circ} \left(\widehat{\partial_{\boldsymbol{l}_i}\psi}(\boldsymbol{\omega}-\boldsymbol{r})\hat{\varphi}(\boldsymbol{\omega}-\boldsymbol{r}) \exp(\pi\imath\boldsymbol{l}_i^{\mathsf{T}}(\boldsymbol{\omega}-\boldsymbol{r}))\right)}{\hat{A}_\varphi(\boldsymbol{\omega}) \sum_{\boldsymbol{r}\in\mathcal{L}^\circ} \hat{\psi}(\boldsymbol{\omega}-\boldsymbol{r})}. \end{aligned} \tag{3.11}$$

We can now state the criterion that needs to be satisfied in order to ensure that the approximation has an order of k.

Proposition 1. *Let ψ be the first-stage generator whose approximation order is at least k. Then the resultant filter D_i satisfies* (3.7) *and provides a k-th order approximation of $\partial_{\boldsymbol{l}_i} f$ when used in conjunction with φ_i. Furthermore, this result holds true irrespective of the second-stage shift.*

The proof of this proposition is deferred till Chapter 4, where we prove a similar proposition (see Proposition 3 on Page 79).

Thus, in order to guarantee a k-th order approximation of the directional derivative, we demand that ψ and φ be the same k-th order reconstruction functions. In this case, the first-stage prefilter p_1 also serves as an interpolation prefilter for a scalar approximation that lies in $\mathbb{V}(\mathcal{L}, \varphi)$. Since all the directional derivative filters d_i have p_1 in common, it only has to be applied once and the resulting data can be used for both scalar interpolation as well as gradient estimation as illustrated in Figure 3.2. Higher quality schemes can be obtained by choosing ψ such that it has an approximation order strictly greater than k. However, we do not discuss such schemes in this chapter.

3.3.3 A Strategy for Designing Practical Filters

Even though the derivative filters that the OP framework yields are asymptotically optimal, they are not advantageous from a practical point of view since they have an infinite impulse response and need to be applied in a preprocessing step resulting in significant storage overhead. The problem of designing derivative filters can be analyzed entirely using the derivative error kernel without resorting to a first-stage auxiliary approximation space. Here, we explore such a strategy that exploits the similarities between the scalar error kernel (1.31) and the derivative error kernel (3.4) to produce separable filters that are practically more advantageous.

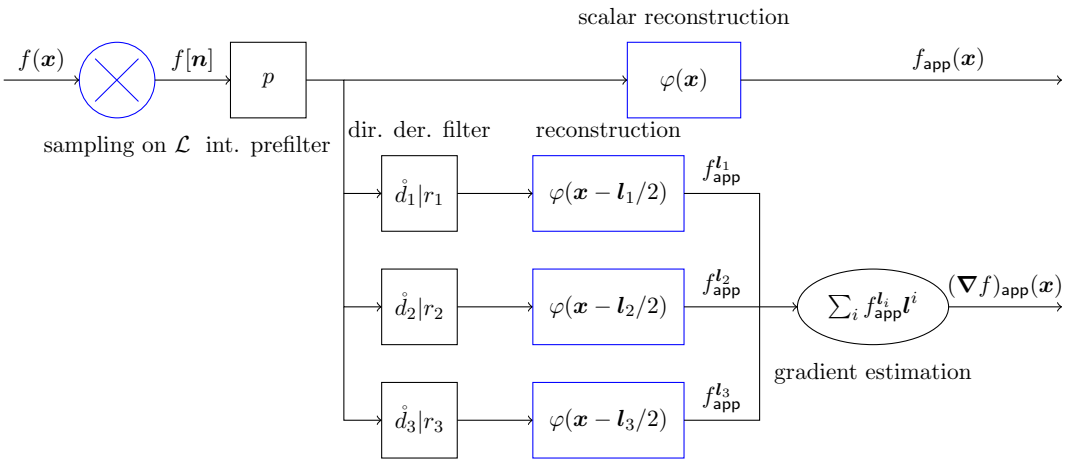


Figure 3.2: Overview of the gradient estimation pipeline in $\mathbb{R}^3$. The sampled data is prefiltered once and can be used for both scalar interpolation as well as gradient estimation. We use the derivative filters $\mathring{d}_1, \mathring{d}_2$ and $\mathring{d}_3$ in the OP framework while r_1, r_2 and r_3 are FIR filters used for on-the-fly derivative estimation.

In order for a directional derivative filter D_i to be asymptotically optimal, it must satisfy the optimality criterion (3.7). This is tantamount to requiring that the Taylor-series expansion of the frequency response $\hat{D}_i(\boldsymbol{\omega})$ match that of the function $2\pi\imath\boldsymbol{l}_i{}^{\mathsf{T}}\boldsymbol{\omega}\hat{\mathring{\varphi}}^{\star}(\boldsymbol{\omega})$ up to order $k+1$ where k is the desired approximation order. Additionally, it is practically desirable that D_i be factorable according to $d_i[\boldsymbol{n}] = (p * r_i)[\boldsymbol{n}] \leftrightarrow \hat{D}_i(\boldsymbol{\omega}) = \hat{P}(\boldsymbol{\omega})\hat{R}_i(\boldsymbol{\omega})$, where r_i depends on the direction of the derivative while p has no such dependence and can be applied once in a preprocessing stage for all the directions in a manner akin to the OP framework (Figure 3.2). With these design criteria, (3.7) can be written as

$$\begin{aligned}\hat{D}_i(\boldsymbol{\omega}) = \hat{P}(\boldsymbol{\omega})\hat{R}_i(\boldsymbol{\omega}) &= \hat{\mathring{\varphi}}^{\star}_i(\boldsymbol{\omega})2\pi\imath\boldsymbol{l}_i{}^{\mathsf{T}}\boldsymbol{\omega} + O(\|\boldsymbol{\omega}\|^{k+1}) \\ &= \big(\hat{\mathring{\varphi}}(\boldsymbol{\omega})\big)\big(2\pi\imath\boldsymbol{l}_i{}^{\mathsf{T}}\boldsymbol{\omega}\exp(\pi\imath\boldsymbol{l}_i{}^{\mathsf{T}}\boldsymbol{\omega})\big) + O(\|\boldsymbol{\omega}\|^{k+1}).\end{aligned} \tag{3.12}$$

It is obvious that if p satisfies

$$\hat{P}(\boldsymbol{\omega}) = \hat{\mathring{\varphi}}(\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^{k}), \tag{3.13}$$

and r_i satisfies

$$\hat{R}_i(\boldsymbol{\omega}) = 2\pi\imath\boldsymbol{l}_i{}^{\mathsf{T}}\boldsymbol{\omega}\exp(\pi\imath\boldsymbol{l}_i{}^{\mathsf{T}}\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^{k+1}), \tag{3.14}$$

then the combined filter d_i satisfies (3.12) as well as the optimality criterion (3.7). The directional dependence due to the derivative and the shift are completely reflected in the response of the derivative filter r_i making the prefilter p directionally independent.

An inspection of the scalar residue term $E_{\mathsf{res}}(\boldsymbol{\omega})$ in (1.31) on page 14 reveals that if we use the symmetric function φ to approximate f in the space $\mathbb{V}(\mathcal{L}_h, \varphi)$, then p also provides an asymptotically optimal k-th order approximation of f, i.e. $\hat{P}(\boldsymbol{\omega}) = \hat{\mathring{\varphi}}(\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^k)$ or equivalently, $E_{\mathsf{res}}(\boldsymbol{\omega}) = O(\|\boldsymbol{\omega}\|^{2k})$. An interpolation prefilter that attempts to exactly interpolate the sample values (given by (2.2) on page 32 with $\psi = \varphi$) satisfies this condition (Section 1.2.3). Such a prefilter is usually employed anyway to approximate the scalar function. Combining it with a derivative filter r_i that satisfies $\hat{R}_i(\boldsymbol{\omega}) = 2\pi\imath \boldsymbol{l}_i^{\mathsf{T}}\boldsymbol{\omega} \exp(\pi\imath \boldsymbol{l}_i^{\mathsf{T}}\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^{k+1})$ will therefore guarantee a k-th order approximation. Higher quality quasi-interpolation prefilters are also possible [CVDV07] and are a topic of future research.

As for the directional component r_i, observe that the substitution $\nu = \boldsymbol{l}_i^{\mathsf{T}}\boldsymbol{\omega}$ converts the multi-dimensional Taylor expansion of the term $\left(2\pi\imath \boldsymbol{l}_i^{\mathsf{T}}\boldsymbol{\omega} \exp(\pi\imath \boldsymbol{l}_i^{\mathsf{T}}\boldsymbol{\omega})\right)$ into a one-dimensional expansion of $\left(2\pi\imath\nu \exp(\pi\imath\nu)\right)$ in the variable ν. Therefore, it suffices to design derivative filters in 1D and then extend them to higher dimensions by simply applying the filter along the lattice direction $\boldsymbol{l}_i$. This is an attractive solution for our design goals as we are interested in keeping the impulse response of r_i as short as possible so that it can be employed on the fly. The resulting overall filtering pipeline is the same as that obtained through the OP framework as shown in Figure 3.2.

3.3.4 Discussion

Error Behavior in 1D

We illustrate the error behavior of the two scenarios considered above with a 1D example where the centered reconstruction function is chosen to be a 4-th order cubic B-spline $\beta^3(x)$. For the OP scenario, the first-stage is also taken to be the cubic B-spline ($\psi(x) = \beta^3(x)$) and the derivative is then projected to a second-stage centered cubic B-spline ($\varphi(x) = \beta^3(x)$) and a shifted cubic B-spline ($\varphi(x) = \beta^3(x - \frac{1}{2})$) yielding the filters *cc* and *cc-s* respectively.

For the practical scenario, the FIR derivative filter is obtained by equating Taylor coefficients upto and including terms of order 4 as explained in Section 3.4.1. The case without the shift is termed *pFIR* while the one with the shift is termed *pFIR-s*, where p refers to the scalar prefilter.

As shown in Figure 3.3, using a shifted reconstruction function leads to better error

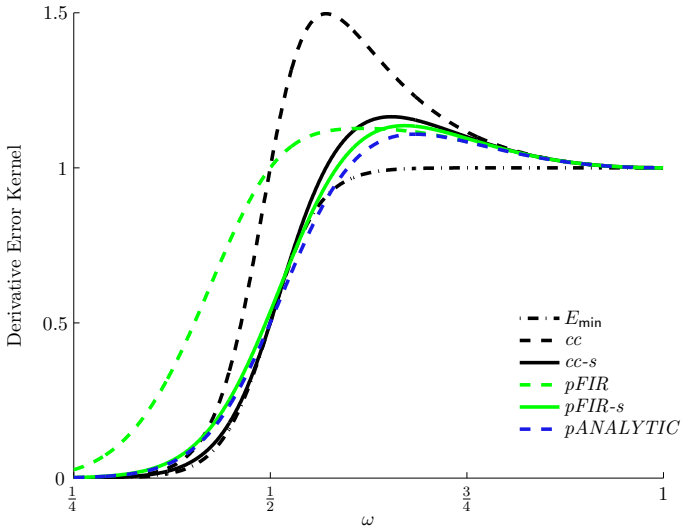


Figure 3.3: The derivative error kernel for various derivative reconstruction schemes designed for the cubic B-spline.

behavior across the board. The error kernel for the OP filter *cc-s* closely follows the minimum error kernel for the cubic B-spline while *cc* departs significantly around $\omega = \pi$ suggesting that the use of this filter would lead to corruption of high frequency content. Using a shifted reconstruction function has a more dramatic impact on the FIR filters as can be clearly seen from the corresponding error kernels. In comparison to *pFIR*, *pFIR-s* vastly improves the error response making it comparable to the OP filter *cc-s*.

Finally, we show that simply computing the analytic derivative of the scalar approximation (*pANALYTIC*) is not the best possible choice. The error kernel for this scheme departs from the minimum sooner as compared to the OP schemes. This should not come as a surprise since the quality is constrained by the approximation characteristics of the scalar approximation.

Gradient Reconstruction

So far, we have only discussed how to accurately reconstruct directional derivatives. The problem of combining the different directional derivatives to estimate the function gradient deserves some attention.

The column vectors of the generating matrix $\mathbf{L}$ of lattice $\mathcal{L}$ define a basis for $\mathbb{R}^s$ that is not necessarily orthogonal. The gradient of a function is coordinate-system independent and

can be conveniently expressed in a dual (contravariant) basis according to

$$\boldsymbol{\nabla} f(\boldsymbol{x}) = \sum\nolimits_{i=1}^{d} (\partial_{\boldsymbol{l}_i} f)(\boldsymbol{x})\, \boldsymbol{l}^i, \tag{3.15}$$

where the dual vectors $\boldsymbol{l}^i$ are column vectors of the matrix $\mathbf{L}^{-\mathsf{T}}$ [You92]. Thus, if the directional derivatives in the principal lattice directions are approximately known, they can be easily combined to yield an approximation of the function gradient.

3.4 Experimental Validation

In order to validate our proposed shifted schemes, we consider various 4-th order gradient estimation filters to be used in conjunction with the tricubic B-spline on the CC lattice and the quintic box spline on the BCC lattice. Both of these reconstruction functions are known to have an approximation order of 4. We refer the reader to Section 1.3.3 for their definitions.

3.4.1 Tricubic B-Spline on CC

Recall that the CC lattice $\mathbb{Z}^3$ is generated by the 3×3 identity matrix (see (1.36) on page 16). As we saw in Chapter 2, it is customary to design continuous reconstruction functions and discrete filters in 1D and then extend them to higher dimensions via a simple tensor product. Consequently, the filters can be applied in a separable way.

OP Derivative Filters

We consider a 3D extension of the 1D case presented in Section 3.3.4 and choose the first-stage function to be the centered tricubic B-spline $\psi(\boldsymbol{x}) = B^3(\boldsymbol{x}) = \beta^3(x_1)\beta^3(x_2)\beta^3(x_3)$.

cc: For the unshifted case, the second stage functions $\varphi_i(\boldsymbol{x})$ are all taken to be $B^3(\boldsymbol{x})$ and the components of the gradient in the three principal directions of $\mathbb{Z}^3$ (i.e. the canonical basis) are orthogonally projected to $\mathbb{V}(\mathbb{Z}^3_h, B^3)$. This case has already been considered in the previous chapter (see Section 2.3.2 on page 34 as well as Table 2.1 on page 36). The resulting filters are completely separable and can be obtained by a tensor product of 1D filters. The first-stage prefilter is given by the samples of $\beta^3(x)$, the auto-correlation sequence is obtained by sampling $\beta^7(x)$ while the derivative filter δ_i in (3.10) is given by the samples of $\frac{d\beta^7}{dx}(x)$ in the direction of the derivative and by the samples of $\beta^7(x)$ in the other directions.

cc-s: We introduce a shift in the second stage and choose the reconstruction functions to be $\varphi_i(\boldsymbol{x}) = B_i^3(\boldsymbol{x})$ (cf. (3.2)). The gradient component in the direction $\vec{\boldsymbol{e}}_i$ is then orthogonally projected to $\mathbb{V}(\mathbb{Z}_h^3, B_i^3)$. The first-stage prefilter and the auto-correlation sequence are the same as the unshifted case *cc*. Using (3.10), we see that the derivative filter δ_i is separable and is given by the samples of $\frac{d\beta^7}{dx}(x + \frac{1}{2})$ in the direction $\vec{\boldsymbol{e}}_i$ and by the samples of $\beta^7(x)$ in the other directions.

Since these are IIR filters, they are efficiently applied to the sampled data in the Fourier domain via a tensor product extension of the FFT. The result is stored in a gradient component volume which is later used during rendering for the purpose of gradient reconstruction.

FIR Derivative Filters

For these schemes, the sampled data is first prefiltered using an interpolation prefilter. This is also done in a preprocessing step using the FFT and the resulting filtered data is used for all subsequent operations. Coefficients needed for reconstructing a partial derivative are computed on the fly using a 1D FIR filter that is aligned in the direction of the derivative.

pFIR: For this case, the centered function $B^3(\boldsymbol{x})$ is used to reconstruct gradient components. We choose an antisymmetric 1D derivative filter with weights $[b, a, 0, -a, -b]$. The derivative criterion (3.14) reduces to

$$2\imath(a\sin(2\pi\omega) + b\sin(4\pi\omega)) = 2\pi\imath\omega + O(\omega^5). \tag{3.16}$$

Expanding both sides and equating coefficients, the solution is found to be $a = 2/3$ and $b = -1/12$. This is the same as the filter 4-*cd* filter developed using our spatial domain Taylor-series framework [HAM11] (also see Figure 2.2 on page 38).

pFIR-s: This is analogous to the case *cc-s* in the sense that the shifted function $B_i^3(\boldsymbol{x})$ is used to reconstruct the partial derivative in the direction $\vec{\boldsymbol{e}}_i$. Let us take the unknown filter weights to be $[l_2, l_1, c, r_1, r_2]$. The derivative criterion (3.14) boils down to

$$c + e^{-4\pi\imath\omega}(r_2 + r_1 e^{2\pi\imath\omega} + l_1 e^{6\pi\imath\omega} + l_2 e^{8\pi\imath\omega}) = 2\pi\imath\omega e^{\imath\pi\omega} + O(\omega^5). \tag{3.17}$$

Equating Taylor coefficients on both sides leads to the solution $[-\frac{1}{24}, \frac{9}{8}, -\frac{9}{8}, \frac{1}{24}, 0]$.

For the shifted schemes *pFIR-s* and *cc-s*, we need to reconstruct the i-th partial derivative with the shifted basis function $B_i^3(\boldsymbol{x})$. Reconstructing at the point $\boldsymbol{x}$ using the shifted basis function is equivalent to reconstructing at the point $\boldsymbol{y} = \boldsymbol{x} - \vec{\boldsymbol{e}}_i/2$ with respect to the centered basis function. Therefore, we simply translate the point of interest by $-\vec{\boldsymbol{e}}_i/2$ and

use the code that implements the centered interpolation scheme. Alternatively, this can also be regarded as a shift of the underlying grid by $\vec{\boldsymbol{e}}_i/2$.

3.4.2 Quintic Box Spline on BCC

We use a scaled version of the BCC lattice generated by the matrix $\mathbf{H} = [\boldsymbol{h}_1\, \boldsymbol{h}_2\, \boldsymbol{h}_3]$, where $\mathbf{H}$ is as given in (2.10) on page 34. For the definition of the 4-th order quintic box spline $\vartheta^4(\boldsymbol{x})$, please consult (2.11) and (2.12).

OP Derivative Filters

Like the CC lattice, we consider two different cases. For both cases, the first-stage function is the quintic box spline, i.e. $\psi(\boldsymbol{x}) = \vartheta^4(\boldsymbol{x})$.

QQ: The partial derivatives of $\vartheta^4(\boldsymbol{x})$ in the canonical directions $\vec{\boldsymbol{e}}_i$ are orthogonally projected to the same target space $\mathbb{V}(\mathcal{H}_h, \vartheta^4)$. This case has also been explored in the previous chapter (see Section 2.3.2 on page 34 as well as Table 2.1 on page 36). The constituent filters are non-separable. In particular, the first-stage prefilter is given by the samples of $\vartheta^4(\boldsymbol{x})$ at the lattice sites of $\mathcal{H}$ while the auto-correlation sequence is obtained by sampling the box-spline $\vartheta^8(\boldsymbol{x})$. Finally, the derivative filter δ_i in (3.10) is obtained by sampling the partial derivative $\partial_i \vartheta^8(\boldsymbol{x})$ at the lattice sites.

QQ-s: For the shifted case, we take the second stage function to be $\varphi_i(\boldsymbol{x}) = \vartheta_i^4(\boldsymbol{x}) := \vartheta^4(\boldsymbol{x} - \frac{1}{2}\boldsymbol{h}_i)$ and orthogonally project the first stage derivative in the direction $\boldsymbol{h}_i$ to the target space $\mathbb{V}(\mathcal{H}_h, \vartheta_i^4)$. The first-stage prefilter and the auto-correlation sequence are unaffected by the shift and are the same as QQ. The weights of the directional derivative filter $\delta_i[\boldsymbol{n}]$ are obtained by evaluating $\partial_{\boldsymbol{h}_i} \vartheta^8(\boldsymbol{x})$ at the sites $\boldsymbol{x} = \mathbf{H}\boldsymbol{n} + \frac{1}{2}\boldsymbol{h}_i$. The three approximated directional derivatives are combined according to (3.15) to yield an estimate of the gradient.

Like the CC case, these IIR filters are also applied to the sampled data in a preprocessing step using our BCC MDFT (Chapter 5).

FIR Derivative Filters

This pipeline proceeds in a manner akin to the CC case above. The sampled data is first prefiltered for use with the quintic box spline. This is implemented in a preprocessing step using the BCC MDFT. The prefiltered data is then used to evaluate derivatives on the fly. We distinguish between two filter types.

P-OPT26: We use our previously derived error optimal 26 weight filter [HAM11] to compute derivatives in the canonical directions. The components are all reconstructed with the centered quintic box spline.

P-FIR-s: We apply the 4 weight shifted FIR filter derived above (cf. (3.17)) to the prefiltered data along the principal directions $\boldsymbol{h}_i$ and use the corresponding shifted quintic box spline $\vartheta_i^4(\boldsymbol{x})$ to reconstruct the directional derivative. Analogous to the CC shifted reconstruction schemes, instead of reconstructing the directional derivative at $\boldsymbol{x}$, we reconstruct it at the translated point $\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{h}_i/2$ using the reconstruction code for the centered quintic box-spline.

3.5 Results and Discussion

In order to assess the impact of our filters on volume visualization, we rendered isosurface images of the synthetic ML test function using the same parameters as Marschner and Lobb [ML94](see Figure 2.3 on page 40). We sampled the function on CC and BCC grids of equivalent resolutions. To effectively discern the effect of a gradient estimation scheme, we used the analytic form to reconstruct the isosurface but used the sampled data to reconstruct the gradients according to the different schemes presented in Section 3.4. Figure 3.4 shows the renditions obtained using the various schemes.

CC vs. BCC: The BCC lattice incurs less errors as compared to the CC lattice. It is known to be an optimal sampling lattice and produces better scalar reconstructions [TMG01, EVM08]. It is therefore not surprising that this benefit carries over to gradient reconstruction as well.

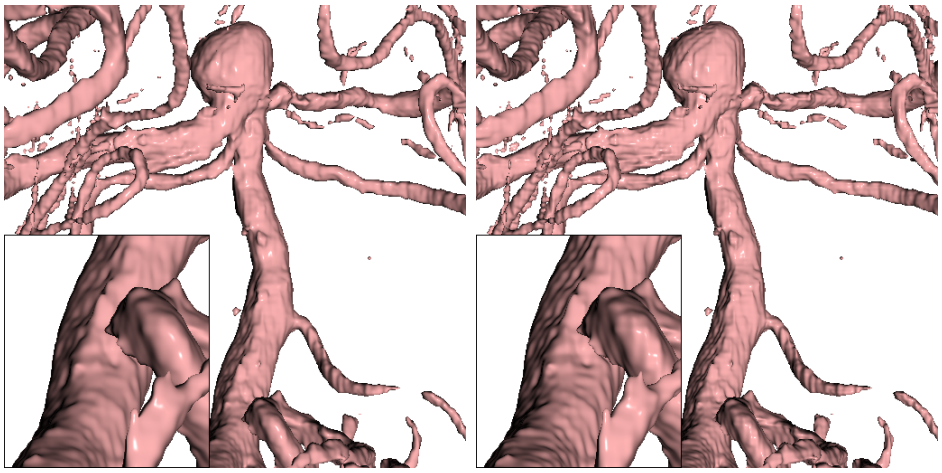
OP vs. Practical: The OP schemes perform better than the FIR schemes both in terms of angles and magnitudes. This further corroborates our analysis in Section 3.3.4 (see also Figure 3.3) where we have shown that the OP filters yield lower error kernels.

Shift vs. Centered: The shifted OP filters incur much less magnitude errors as predicted by the derivative error kernel. However, surprisingly, this trend seems to be reversed when we consider the angular error distributions. On the other hand, for the FIR filters, the shifts have a clear advantage. The filters *pFIR-s* and *P-FIR-s* lead to lower angular and magnitude errors as compared to their centered counterparts *pFIR* and *P-OPT*26.

(a) *pANALYTIC*, 12.46°, 1.05 (b) *CC*, 9.22°, 1.11 (c) *CC-s*, 10.21°, 0.92 (d) *pFIR*, 24.14°, 2.11 (e) *pFIR-s*, 11.53°, 1.15

(f) *P-ANALYTIC*, 10.05°, 0.94 (g) *QQ*, 6.91°, 0.91 (h) *QQ-s*, 7.43°, 0.76 (i) *P-OPT*26, 18.10°, 1.96 (j) *P-FIR-s*, 8.7°, 1.03

Figure 3.4: The 0.5 isosurface of the ML function shaded using different gradient estimation schemes; top row, CC, and bottom row, BCC. The analytic function was used to reconstruct the isosurface while sampled data (CC: $41 \times 41 \times 41$, BCC: $32 \times 32 \times 64$) were used for gradient estimation. For comparison, the left half of each image shows the truth. The mean angular error and the mean length of the error vector are indicated. The terms *pANALYTIC* and *P-ANALYTIC* refer to computing the analytic gradient of the scalar approximation.



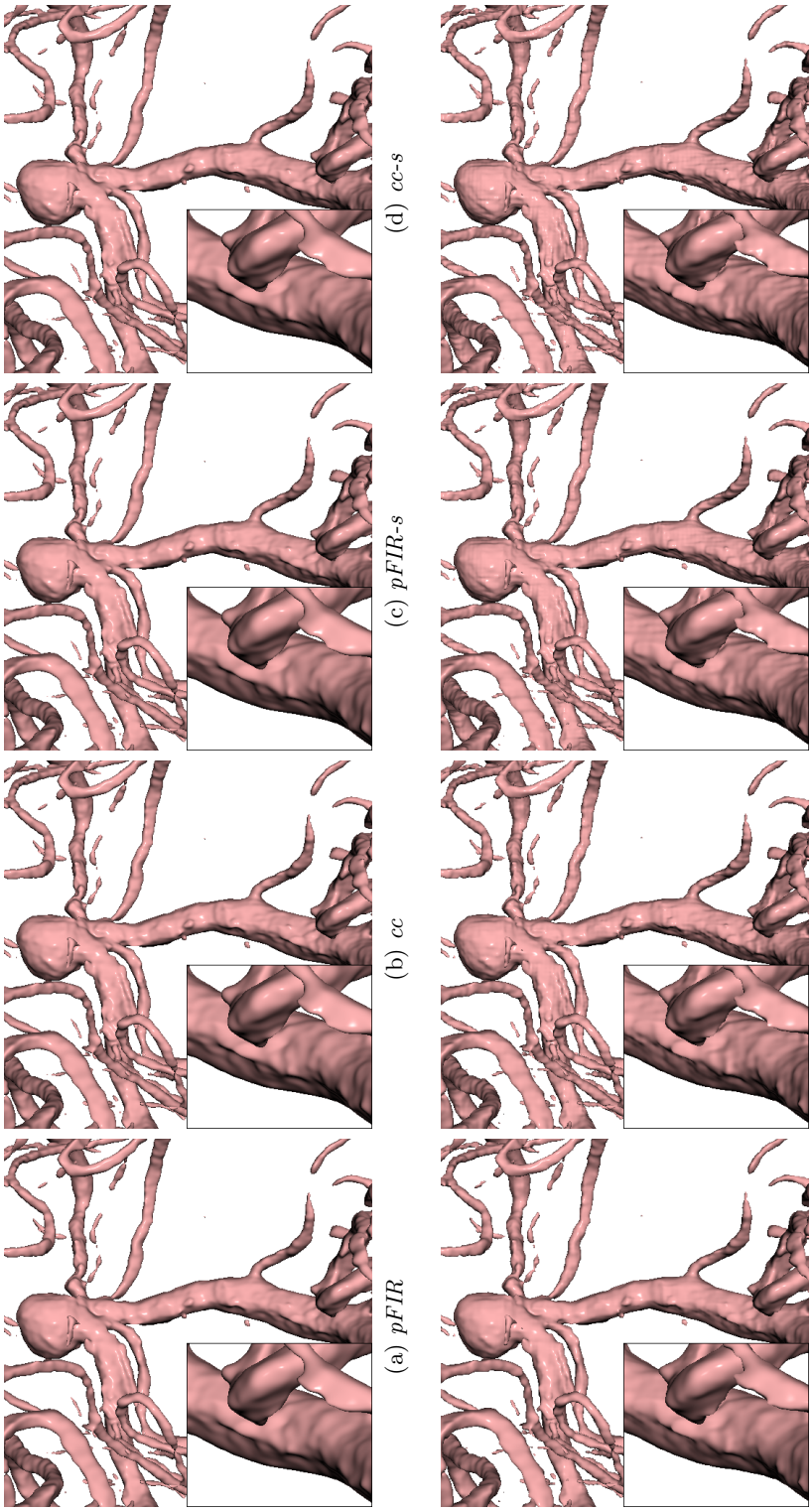
(a) *pFIR* (b) *pFIR-s*

Figure 3.5: An isosurface (isovalue = 1000) of the high resolution (512^3) aneurysm CC dataset reconstructed using prefiltered tricubic B-spline interpolation.

Analytic vs. FIR: For the most part, the analytic derivative performs well, specially in comparison to the centered, orthogonal FIR schemes *pFIR* and *P-OPT*26. Undoubtedly, the IIR OP schemes are better, and even more so when the crucial aspect is the orientation of the gradient. With the introduction of a shift, the FIR schemes become almost as good as the analytical gradient in terms of the gradient magnitude. They seem to have an advantage in terms of the orientation of the gradient. However, this is rather fortuitous since the criterion optimized by the error kernel is the magnitude and not the orientation.

In order to investigate the effect of the shifted schemes, we experimented with an aneurysm dataset obtained through an angiography scan. Isosurface renditions of the original high resolution CC dataset are shown in Figure 3.5. Even at this resolution, the differences between the centered and shifted FIR schemes are remarkable, *pFIR-s* clearly reveals details that are smoothed out by the centered scheme *pFIR*.

We downsampled this dataset on equivalent CC and BCC grids and reconstructed the same isosurface using the gradient estimation schemes outlined in Section 3.4. In order to ensure that we remain in the low-pass regime, appropriate anti-aliasing filters were applied in the Fourier domain (using the FFT) before downsampling. To create a BCC volume downsampled by a factor of 4, we filtered the CC dataset by zeroing out the spectrum outside a rhombic dodecahedron that is the Voronoi cell of the dual FCC lattice. The resulting CC



(a) *pFIR* (b) *cc* (c) *pFIR-s* (d) *cc-s*

(e) *P-OPT26* (f) *QQ* (g) *P-FIR-s* (h) *QQ-s*

Figure 3.6: Downsampled isosurface renditions of the aneurysm dataset, top row CC (323^3) and bottom row, BCC ($256 \times 256 \times 512$). The shading differences are solely due to the different gradient estimation schemes.

volume was then simply subsampled on a BCC lattice. An equivalent CC volume was created by discarding the spectrum outside the rectangular region corresponding to the Voronoi cell of the downsampled CC volume. The resulting images are shown in Figure 3.6. It should be stressed that the underlying isosurface for each lattice type is the same, since the same prefiltered reconstruction scheme is used to find the isosurface.

The visual differences between the various CC renditions are subtle. Nevertheless, one can observe that the OP scheme *cc* does a better job at preserving the high frequency details as compared to the FIR scheme *pFIR* which has the greatest smoothing effect. With an introduction of a shift, the FIR scheme *pFIR-s* recovers the lost details and is visually comparable to the shifted OP scheme *cc-s*. In contrast to the CC lattice, the BCC lattice provides a better scalar reconstruction and is more sensitive to the various derivative filter combinations. As before, the centered FIR scheme *P-OPT*26 has a strong smoothing effect. In comparison, the centered OP scheme *QQ* fares a lot better as shown by the zoomed in regions of the corresponding images. The greatest improvement is shown by the the shifted schemes *P-FIR-s* and *QQ-s*. They dramatically improve visual quality by revealing high frequency details and enhancing contrast. We have also compared these renditions to those obtained by computing the analytic gradient. The visual differences between the shifted schemes and the analytic gradient are hard to discern, although we did notice that the shifted OP schemes reproduce edges better and are more accurate in preserving the gradient magnitude in high frequency regions.

The benefits of the shifted schemes also extend to Direct Volume Rendering (DVR) as shown in Figure 3.7 for the case of the CC carp dataset. The images obtained by rendering the downsampled dataset clearly demonstrate the dramatic impact a mere shift can have on visual quality. Even though several color values are composited to produce a DVR rendering, the effect of a poor normal estimation scheme persists specially in areas of high variability.

In summary, the shifted OP schemes yield the best results. However, they achieve the superior visual quality at the expense of an added storage overhead. On the other hand, the shifted FIR schemes not only yield results that rival those obtained through the shifted OP schemes, they are also cheap to compute and do not require any changes to the underlying interpolation kernel as is the case with the analytic gradient. This makes them ideally suited for practical applications where both efficiency and accuracy are crucial. It should be emphasized that the scenario we have considered is the bare minimum to guarantee fourth-order convergence. Both the frameworks considered in Section 3.3 can be easily extended

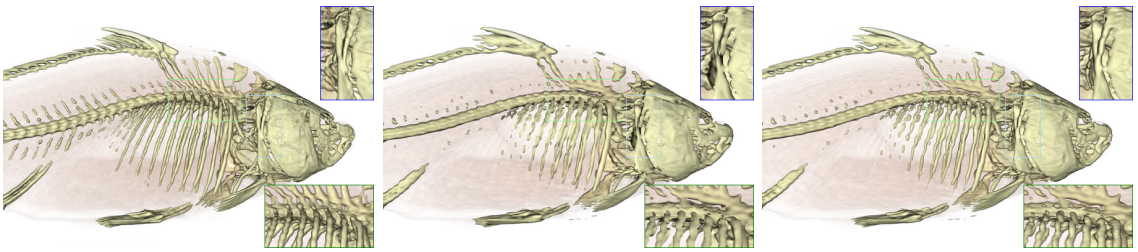


Figure 3.7: DVR images of the carp dataset. Left: Original Cartesian (256^3) dataset reconstructed using prefiltered tricubic B-splines and shaded using the analytic gradient. Middle: Downsampled Cartesian (128^3) dataset reconstructed using prefiltered tricubic B-splines and shaded using the centered scheme *pFIR*. Right: The middle dataset shaded using the shifted scheme *pFIR-s*. Notice how the details in the bones and skull are much better preserved as compared to the centered scheme even though the underlying scalar interpolation is the same. This gain in quality comes at no additional cost.

to obtain higher quality filters which can be combined with the tricubic B-spline (CC) or the quintic box spline (BCC) to further reduce the error.

3.6 Notes

This chapter is largely based on our work [AMC10]. *Mathematica* notebooks for the computation of the OP filter weights as well as test results using the trilinear B-spline on CC and the linear box spline on BCC can be found in the supplementary material of this work. It can be diffult to visually discern the differences between the various rendering in print. We believe that these differences are best observed by overlaying the images on top of each other and flipping between them.

The inspiration for the term '*revitalized*' comes from the work of Blu *et al.* [BTU04], where the authors show that the optimal linear interpolator (one that minimizes the error kernel (1.31) on page 14) in the univariate setting is actually a shifted one. Our motivation for using a shifted kernel is somewhat different. However, one could make use of the derivative error kernel (3.4) to find optimally shifted derivative reconstruction kernels.

The idea of using a shifted generator for approximating derivatives is akin to a staggered grid-based approach to solving the Navier-Stokes equations, where scalar quantities are stored at cell centeres while vector quantities are stored at cell faces [HW65, FSJ01, Bri08]. This leads to reduced numerical dissipation when computing the divergence of the flow at cell centers.

Chapter 4

On the Solution of Poisson's Equation in a Rectangular Domain

4.1 Introduction

In this chapter, we build upon the ideas presented in the previous chapters to propose a method for solving Poisson's equation with homogeneous Dirichlet boundary conditions inside the unit hypercube $\mathcal{C}^s$. We follow an approach that is similar to our treatment of gradient estimation. We first identify the operator that needs to be discretized and then seek a convolution based solution methodology that is consistent with a chosen target space, i.e. it fully harnesses the L_2 approximation order of the space.

Grid based numerical solutions to elliptic partial differential equations such as Poisson's equation are usually studied under the umbrella of *finite difference* methods [Str04]. Finite difference methods are designed to guarantee a certain pointwise rate of convergence of the error. On the other hand, we are interested in seeking an approximate solution V_{app} that lies in a shift-invariant space generated by a kernel φ. The relevant error metric is therefore the L_2-norm $\|\cdot\|_{L_2(\mathcal{C}^s)}$ and we require that the approximation satisfies $\|V - V_{\mathsf{app}}\|_{L_2(\mathcal{C}^s)} = O(h^k)$, where V is the analytic solution, k is the approximation order provided by φ, and h is the isotropic scaling parameter. In this sense, our solution methodology is reminiscent of the *finite element* method [QV08]. The explicit connection is detailed in Section 4.3.3.

4.2 Preliminaries

We shall normalize the domain of interest to the s-dimensional unit cube $\mathcal{C}^s := [0,1]^s$. We shall be dealing with periodic functions that have their fundamental period in the domain $\mathcal{P}^s := [-1,1]^s$. We denote the corresponding open domains as $\mathcal{C}_o^s := (0,1)^s$ and $\mathcal{P}_o^s := (-1,1)^s$ respectively.

In a manner similar to (1.7) on page 4, we denote the inner product between two $L_2(\mathcal{P}^s)$ functions $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ over the domain $\mathcal{P}^s$ as (note the absence of complex conjugation)

$$\left\langle f_1, f_2 \right\rangle_{\mathcal{P}^s} := \frac{1}{2^s} \int_{\mathcal{P}^s} f_1(\boldsymbol{x}) f_2(\boldsymbol{x}) d\boldsymbol{x}. \tag{4.1}$$

The corresponding norm induced by this inner product is indicated as $\|\cdot\|_{L_2(\mathcal{P}^s)}$.

We wish to seek an approximation of the function V that satisfies the Poisson equation with homogeneous Dirichlet boundary conditions, i.e.

$$\begin{aligned} \Delta V &= f, \ \text{ in } \ \mathcal{C}_o^s, \\ V &= 0, \ \text{ on } \ \partial\mathcal{C}^s, \end{aligned} \tag{4.2}$$

where Δ is the s-dimensional Laplace operator and $\partial\mathcal{C}^s$ denotes the boundary of $\mathcal{C}^s$, i.e. $\partial\mathcal{C}^s \cup \mathcal{C}_o^s = \mathcal{C}^s$.

4.2.1 Analytic Solution

Green's Function

The Poisson equation is an example of a well-studied partial differential equation. When the domain of interest is the unit cube $\mathcal{C}^s$, the solution can be analytically expressed in terms of a Green's function $\mathsf{G}(\boldsymbol{x}, \boldsymbol{y})$ that represents the potential due to a point source placed at the location $\boldsymbol{x}$ inside $\mathcal{C}^s$. In other words,

$$\Delta_{\boldsymbol{x}} \mathsf{G}(\boldsymbol{x}, \boldsymbol{y}) = \delta(\boldsymbol{x} - \boldsymbol{y}), \tag{4.3}$$

where δ is the Dirac delta distribution and the Laplacian operates on the $\boldsymbol{x}$ variable. The Green's function G has the Fourier Sine series expansion

$$\mathsf{G}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{m} \in \mathbb{Z}_+^s} \frac{\prod_{i=1}^s \sin(m_i \pi x_i) \sin(m_i \pi y_i)}{-\pi^2 \|\boldsymbol{m}\|^2}, \tag{4.4}$$

and the solution V is given by

$$V(\boldsymbol{x}) = \int_{\mathcal{C}^s} f(\boldsymbol{y}) \mathsf{G}(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y}. \tag{4.5}$$

Fourier Domain Interpretation

Even though there has been much effort on finding alternate rapidly convergent series representations of the Green's function (see e.g. Marshall [Mar99]), it turns out that the form of the Green's function given in (4.4), is ideally suited for our needs as it allows us to easily express the solution in the Fourier domain in terms of the Fourier coefficients of f.

Suppose we are given an $L_2(\mathcal{C}^s)$ function $\rho(x)$ that is defined on the open unit cube $\mathcal{C}^s_o$. In order to obtain a Fourier sine series that converges to ρ almost everywhere in $\mathcal{C}^s_o$, we need to extend ρ periodically such that it is odd with respect to each variable. Particularly, let us extend the domain of ρ so that, within the interval $\mathcal{P}^s$, it satisfies

$$\rho(\boldsymbol{x}) := \begin{cases} \big(\prod_{i=1}^s \operatorname{sgn}(x_i)\big)\rho(\big|x_1\big|,\ldots,\big|x_s\big|) & \text{if } \forall i\ \big|x_i\big| \neq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{4.6}$$

while outside $\mathcal{P}^s$, it is $\mathcal{P}^s$-periodic, i.e. $\rho(\boldsymbol{x}+2\boldsymbol{k}) = \rho(\boldsymbol{x})$ for $\boldsymbol{k} \in \mathbb{Z}^s$. The extended function ρ can be developed into a multidimensional Fourier sine series, the coefficients of which are given by

$$\tilde{\rho}[\boldsymbol{m}] = 2^s \big\langle \rho(\boldsymbol{x}), \textstyle\prod_{i=1}^s \sin(m_i \pi x_i) \big\rangle_{\mathcal{P}^s}, \quad \text{for } \boldsymbol{m} \in \mathbb{Z}^s_+. \tag{4.7}$$

The coefficient sequence $\tilde{\rho}[\boldsymbol{m}]$ can also be seen as a special case of the more general multidimensional Fourier series. In fact, with an odd extension of the sequence $\tilde{\rho}$, the function ρ can be expressed as a Fourier series. In particular, if we define the Fourier series coefficients $\hat{\rho}[\boldsymbol{m}]$ (for $\boldsymbol{m} \in \mathbb{Z}^s$) as

$$\hat{\rho}[\boldsymbol{m}] := \begin{cases} \frac{1}{(2\imath)^s} \prod_i \operatorname{sgn}(m_i) \tilde{\rho}[\big|m_1\big|,\ldots,\big|m_s\big|] & \text{if } \forall i\ \big|m_i\big| \neq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{4.8}$$

then ρ is also given by the Fourier series

$$\rho(\boldsymbol{x}) = \sum_{\boldsymbol{m} \in \mathbb{Z}^s} \hat{\rho}[\boldsymbol{m}] \exp(\imath \pi \boldsymbol{m} \cdot \boldsymbol{x}). \tag{4.9}$$

Thus, we use the notation $\tilde{\rho}[\cdot]$ and $\hat{\rho}[\cdot]$ to distinguish between the Fourier sine series and Fourier series coefficients of the periodic function ρ.

Since the Green's function G is defined in $\mathcal{C}^s_o$ and has a Fourier sine series representation, it is natural to seek a solution that can be developed into a Fourier sine series as well. Consequently, for the remainder of this section, we shall only be dealing with Fourier sine series. Therefore, it suffices to consider only the coefficients in $\mathbb{Z}^s_+$.

Using the analytic solution (4.5) and the sine series representation of the Green's function (4.4), it is easy to show that the solution is given by

$$-\tilde{V}[\boldsymbol{m}] = \frac{\tilde{f}[\boldsymbol{m}]}{\pi^2\|\boldsymbol{m}\|^2}, \tag{4.10}$$

where $\boldsymbol{m} \in \mathbb{Z}_+^s$ and $\tilde{f}[\cdot]$ represents the Fourier sine series coefficients of the odd extension of f. To facilitate subsequent discussions, let us introduce the solution operator $\Delta^{-1} \Leftrightarrow \frac{1}{-\pi^2\|\boldsymbol{m}\|^2}$ ($\boldsymbol{m} \in \mathbb{Z}_+^s$), where the symbol $\Leftrightarrow$ represents how the Fourier sine series coefficients are affected by the operator. The self-adjoint operator Δ^{-1} is the inverse of the Laplace operator. Self-adjointness can be easily verified with the aid of Parseval's relation, which states that

$$\big\langle a, b\big\rangle_{\mathcal{P}^s} = \sum_{\boldsymbol{m}\in\mathbb{Z}_+^s} \tilde{a}[\boldsymbol{m}]\tilde{b}[\boldsymbol{m}] \tag{4.11}$$

for any $\mathcal{P}^s$-periodic functions a and b that are in $L_2(\mathcal{P}^s)$ and odd.

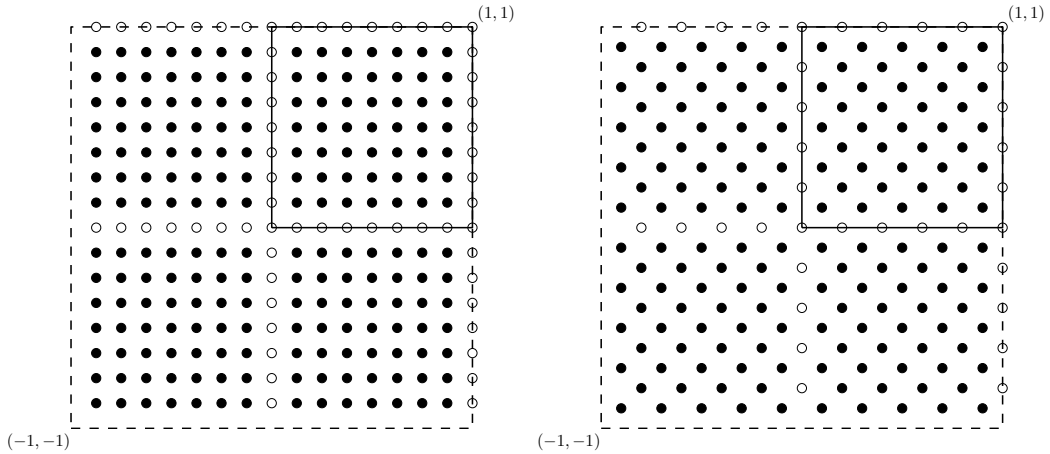
4.2.2 Approximate Solution

We are interested in the scenario where the function f is only known through its point samples. Specifically, we assume that the samples of f reside on a scaled version of an s-dimensional lattice $\mathcal{L}$ that is generated by the matrix $\mathbf{L}$. Recall that the lattice $\mathcal{L}$ is the group (under addition) formed by the set of points $\{\mathbf{L}\boldsymbol{k} : \boldsymbol{k} \in \mathbb{Z}^s\}$. As before, we denote by $\mathcal{L}_h$, the scaled lattice generated by the matrix $h\mathbf{L}$. Unless otherwise stated, we assume that $\mathbf{L}$ is normalized, i.e. $\big|\det(\mathbf{L})\big| = 1$.

For the purpose of enumerating the samples that are contained within $\mathcal{P}^s$, let us define the point set

$$\mathcal{P}_h := \underbrace{\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{2^sN}\}}_{\mathcal{I}_h} \cup \underbrace{\{\boldsymbol{x}_{2^sN+1}, \ldots, \boldsymbol{x}_{2^sN+M}\}}_{\mathcal{B}_h}, \tag{4.12}$$

where $\mathcal{I}_h := \mathcal{P}_o^s \cap \{\boldsymbol{x} + \boldsymbol{m} : \boldsymbol{x} \in \mathcal{L}_h \cap \mathcal{C}_o^s, \boldsymbol{m} \in \mathbb{Z}^s\}$ consists of interior lattice points, while $\mathcal{B}_h$ consists of extended boundary points, i.e. $\mathcal{B}_h := (\mathcal{L}_h \cap \partial\mathcal{P}^s \cap (-1, 1]^s) \cup (\mathcal{L}_h \cap \mathcal{P}_o^s \backslash \mathcal{I}_h)$. Two dimensional illustrations are shown in Figure 4.1. We note that with the above definition of $\mathcal{I}_h$, the points $\boldsymbol{x}_j$ for $j \in \{1, \ldots, N\}$ are contained in the open unit cube $\mathcal{C}_o^s$ whereas the remaining points ($\boldsymbol{x}_j$ for $j \in \{N+1, \ldots, 2^sN\}$) lie outside. Furthermore, we assume that the lattice $\mathcal{L}$ and the sampling rate h are such that the set $\{\boldsymbol{x} + 2\boldsymbol{m} : \boldsymbol{x} \in \mathcal{P}_h, \boldsymbol{m} \in \mathbb{Z}^s\} = \mathcal{L}_h$. We remark that this requirement for $\mathcal{L}_h$ is also satisfied by integration lattices that are commonly used to devise quadrature rules within the unit cube $\mathcal{C}^s$ [Nie92].



(a) Cartesian (b) Quincunx

Figure 4.1: Illustration of the point set $\mathcal{P}_h$ on the two dimensional Cartesian ($\mathbf{L} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, $h = \frac{1}{8}$) and Quincunx ($\mathbf{L} = \left(\begin{smallmatrix} 1 & -1 \\ 1 & 1 \end{smallmatrix}\right)$, $h = \frac{1}{10}$) lattices. The interior points ($\bullet$) belong to the set $\mathcal{I}_h$ while the extended boundary points ($\circ$) belong to $\mathcal{B}_h$.

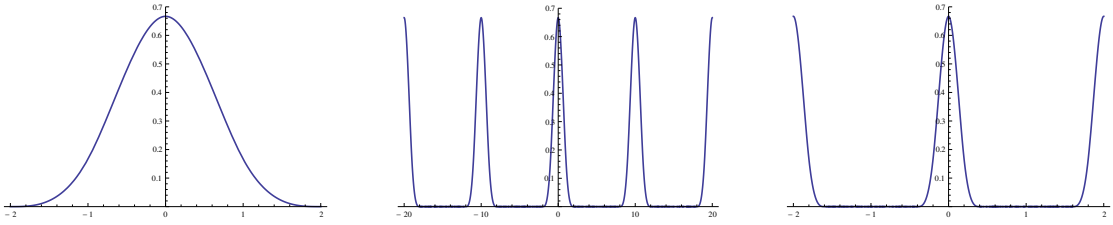
We denote the interior samples of f as $f[\boldsymbol{x}_j] := f(\boldsymbol{x}_j)$ where $\boldsymbol{x}_j \in \mathcal{I}_h$. Note that $f[\boldsymbol{x}_j]$ only needs to be known in $\mathcal{C}_o^s$ ($j \in \{1, \ldots, N\}$), the other samples can be inferred from oddity.

We wish to use the samples of f to seek an approximation V_{app} of the function V that solves the Poisson equation (4.2). Since V is a periodic function, we follow the recipe of Jacob *et al.* [JBU02] and seek an approximation that lies in a space generated by a periodic reconstruction function. Specifically, we are interested in the case where V_{app} lies in the space $\mathbb{V}(\mathcal{L}_h, \varphi_p) := \text{span}_{\boldsymbol{x}_j \in \mathcal{P}_h}\{\varphi_p(\frac{\boldsymbol{x}-\boldsymbol{x}_j}{h})\}$ that is spanned by the scaled and translated versions of a periodic function φ_p. Our sought-after approximation is given by

$$V_{\mathsf{app}}(\boldsymbol{x}) = \sum_{\boldsymbol{x}_j \in \mathcal{P}_h} c[\boldsymbol{x}_j] \varphi_p\big(\frac{\boldsymbol{x} - \boldsymbol{x}_j}{h}\big), \tag{4.13}$$

where $c[\boldsymbol{x}_j]$ is an unknown coefficient sequence defined on the point set $\mathcal{P}_h$ that is to be determined from the samples of f. The function φ_p is a periodized version of a generating function φ (Figure 4.2) and is defined as

$$\varphi_p(\boldsymbol{x}) := \sum_{\boldsymbol{m} \in \mathbb{Z}^s} \varphi\big(\boldsymbol{x} - \frac{2}{h}\boldsymbol{m}\big). \tag{4.14}$$



(a) $\beta^3(x)$ (b) $\beta^3_p(x)$ (c) $\beta^3_p(\frac{x}{h})$

Figure 4.2: 1D illustration of the periodization operation defined by (4.14): $\varphi(x) = \beta^3(x)$ and $h = 0.2$.

It is easy to verify that, with this definition of φ_p, V_{app} is also $\mathcal{P}^s$-periodic. Even though we are only interested in the behavior of V_{app} inside $\mathcal{C}^s$, extending V_{app} to the entire domain $\mathcal{P}^s$ using (4.13) allows us to use the Fourier domain error kernel proposed by Jacob *et. al* [JBU02] to quantify the approximation error $\|V - V_{\mathsf{app}}\|_{L_2(\mathcal{P}^s)}$.

Another simplification is in order here. Since the solution V to the Poisson problem (4.2) is odd with respect to each variable, we look for an approximate solution $V_{\mathsf{app}} \in \mathbb{V}(\mathcal{L}_h, \varphi_p)$ that is also odd in each variable. This can be achieved by setting the boundary coefficients to zero and by requiring that the resulting sequence be odd. With $c[\boldsymbol{x}_j] = 0$ for $\boldsymbol{x}_j \in \mathcal{B}_h$, the approximation (4.13) simplifies to

$$V_{\mathsf{app}}(\boldsymbol{x}) = \sum_{j=1}^{2^s N} c[\boldsymbol{x}_j]\varphi_p(\frac{\boldsymbol{x} - \boldsymbol{x}_j}{h}). \tag{4.15}$$

If the generator φ is even and the sequence $c[\cdot]$ is odd, the resulting approximation V_{app} will also be odd. This can be easily verified by inspecting the Fourier series coefficients of V_{app}. Using the Poisson summation formula (see (1.4) on page 3), the Fourier series coefficients $\widehat{V_{\mathsf{app}}}[\cdot]$ are given by

$$\widehat{V_{\mathsf{app}}}[\boldsymbol{m}] = \hat{\varphi}(\frac{h}{2}\boldsymbol{m}) \sum_{j=1}^{2^s N} c[\boldsymbol{x}_j] \exp(-\imath\pi \boldsymbol{m} \cdot \boldsymbol{x}_j), \quad \text{where } \boldsymbol{m} \in \mathbb{Z}^s. \tag{4.16}$$

Since the coefficient sequence $c[\cdot]$ is odd, the summation in (4.16) can be simplified to yield

$$\widehat{V_{\mathsf{app}}}[\boldsymbol{m}] = \hat{\varphi}(\frac{h}{2}\boldsymbol{m})(2\imath)^s \tilde{C}[\boldsymbol{m}], \tag{4.17}$$

where $\tilde{C}[\boldsymbol{m}]$ denotes the multidimensional discrete sine transform (MDST) of the sequence

$c[\cdot]$ (not to be confused with the Fourier sine series), and is given by

$$\tilde{C}[\boldsymbol{m}] := \sum_{j=1}^{N} c[\boldsymbol{x}_j]\big(\textstyle\prod_{i=1}^{s} \sin(m_i \pi x_{j,i})\big), \quad \text{where } \boldsymbol{m} \in \mathbb{Z}^s. \tag{4.18}$$

Now, since $\hat{\varphi}(\boldsymbol{\omega})$ is a real even function and $\tilde{C}[\boldsymbol{m}]$ is odd, the Fourier series coefficients $\widehat{V_{\mathsf{app}}}[\boldsymbol{m}]$ satisfy the odd extension (4.8) with the sine series coefficients being $\widetilde{V_{\mathsf{app}}}[\boldsymbol{m}] = (-4)^s \hat{\varphi}(\frac{h}{2}\boldsymbol{m})\tilde{C}[\boldsymbol{m}]$, for $\boldsymbol{m} \in \mathbb{Z}_+^s$. This establishes the fact that the approximate solution V_{app} is also odd.

With this simplification, the coefficient sequence $c[\boldsymbol{x}_j]$ only needs to be determined for $j \in \{1, \ldots, N\}$. The rest can be inferred from oddity. In order to obtain $c[\cdot]$ from the samples $f[\cdot]$, we look for a digital filtering solution that can be efficiently implemented in the Fourier domain via the MDST. In Section 4.3, we extend the error kernel formulation of Jacob *et al.* [JBU02] to analyze the error behaviour of a digital filtering based approximation methodology. In Section 4.4, we present two digital filtering schemes that can be tuned to achieve a desired order of accuracy. Some numerical tests in two and three dimensions are presented in Section 4.5.

4.3 Error Analysis

4.3.1 Error Kernel for Periodic Functions

In many approximation problems involving periodic functions, one is interested in seeking an approximation z_{app} of a periodic function z from its measurements that lie on some sampling lattice. In light of the notations introduced earlier, suppose that $z \in L_2(\mathcal{P}^s)$ and is also $\mathcal{P}^s$-periodic. Additionally, suppose that the measurements are made at the locations $\boldsymbol{x}_j \in \mathcal{P}_h$ so that one can obtain an approximation z_{app} that belongs to $\mathbb{V}(\mathcal{L}_h, \varphi_p)$ and — in a manner similar to the expansion (4.13) — is given by

$$z_{\mathsf{app}}(\boldsymbol{x}) = \sum_{\boldsymbol{x}_j \in \mathcal{P}_h} \zeta[\boldsymbol{x}_j] \varphi_p\Big(\frac{\boldsymbol{x} - \boldsymbol{x}_j}{h}\Big), \tag{4.19}$$

where φ_p is a periodized generating function, and the coefficient sequence $\zeta[\cdot]$ is obtained through the discrete measurements

$$\zeta[\boldsymbol{x}_j] = \big\langle z(\boldsymbol{x}), \tilde{\varphi}_p\big(\frac{\boldsymbol{x} - \boldsymbol{x}_j}{h}\big)\big\rangle_{\mathcal{P}^s} \tag{4.20}$$

made with the scaled and shifted versions of a periodic analysis function $\tilde{\varphi}_p(\boldsymbol{x})$, where the periodization operation is similar to (4.14). The main result of Jacob *et al.* [JBU02] states that the mean square approximation error $\|z - z_{\mathsf{app}}\|_{L_2(\mathcal{P}^s)}$ at scale h can be predicted according to

$$\sqrt{\sum_{\boldsymbol{m}\in\mathbb{Z}^s} \|\hat{z}[\boldsymbol{m}]\|^2 E(\frac{h}{2}\boldsymbol{m})} \tag{4.21}$$

where $\hat{z}[\boldsymbol{m}]$ are the Fourier series coefficients of z and $E(\boldsymbol{\omega})$ is the error kernel of Blu *et al.* [BU99a] and is given by

$$E(\boldsymbol{\omega}) := E_{\mathsf{min}}(\boldsymbol{\omega}) + \underbrace{\hat{A}_{\varphi}(\boldsymbol{\omega})\left|\hat{\tilde{\varphi}}(\boldsymbol{\omega}) - \hat{\mathring{\varphi}}(\boldsymbol{\omega})\right|^2}_{E_{\mathsf{res}}(\boldsymbol{\omega})}, \tag{4.22}$$

where the minimum error kernel $E_{\mathsf{min}}(\boldsymbol{\omega})$ is given by (1.31) on page 14. Remarkably, the same error kernel can be used to predict the error for periodic and non-periodic functions alike. The only change that needs to be made is in the prediction equation. For functions in $L_2(\mathbb{R}^s)$, the error is predicted according to the integral in (1.32). On the other hand, for periodic functions that belong to $L_2(\mathcal{P}^s)$, the prediction equation turns into the summation given in (4.21). We refer the reader to Jacob *et al.* [JBU02] for details. Recall that the error kernel (4.22) achieves its minimum value of $E_{\mathsf{min}}(\boldsymbol{\omega})$ when the residue error $E_{\mathsf{res}}(\boldsymbol{\omega}) = 0$, or equivalently, when the analysis function $\tilde{\varphi}$ is the dual of φ. Alternatively, in situations where the minimum approximation scenario cannot be realized, the residue error kernel provides a convenient way to get close to the minimum by designing analysis functions $\tilde{\varphi}$ that match the dual $\mathring{\varphi}$ up to a suitable level that meets the demands of the application.

4.3.2 Extension to Linear Operators

Many a time, we are interested in approximating from the discrete measurements of z, not the function z itself but a function Γz that is obtained by applying the linear operator Γ to z. This is exactly the problem we are faced with in the case of approximating the analytic solution (4.10) to Poisson's equation. We would like to extend the error kernel formulation (4.22) so that it will allow us to quantify the error incurred in approximating Γz from the measurements of z. Towards this end, we assume that the approximation $(\Gamma z)_{\mathsf{app}}$ lies in a shift-invariant space spanned by the generator φ_p, where the coefficients are obtained from the measurements through a digital filtering operation. Particularly, we

seek an approximation that is given by

$$(\Gamma z)_{\mathsf{app}}(\boldsymbol{x}) = C_h \sum_{\boldsymbol{x}_j \in \mathcal{P}_h} (\zeta \circledast \gamma_h)[\boldsymbol{x}_j] \varphi_p(\frac{\boldsymbol{x} - \boldsymbol{x}_j}{h}), \tag{4.23}$$

where ζ denotes the discrete measurements of z obtained through (4.20), γ is a digital filter defined on the lattice $\mathcal{L}$ and γ_h is its corresponding scaled version, i.e. $\gamma_h[\boldsymbol{x}_j] := \gamma[\frac{\boldsymbol{x}_j}{h}]$ where $\frac{\boldsymbol{x}_j}{h} \in \mathcal{L}$, and C_h is an associated scaling constant. The digital filter γ represents a suitable discretization of the operator Γ on the lattice $\mathcal{L}$. It is to be applied to the measurements through a cyclic convolution operation (denoted by $\circledast$)on the lattice $\mathcal{L}_h$ using a periodized version of γ_h. In particular, the convolution operation in (4.23) is defined as

$$(\zeta \circledast \gamma_h)[\boldsymbol{x}_j] := \sum_{\boldsymbol{x}_k \in \mathcal{P}_h} \big(\sum_{\boldsymbol{m} \in \mathbb{Z}^s} \gamma[\frac{\boldsymbol{x}_k + 2\boldsymbol{m}}{h}]\big) \zeta[\boldsymbol{x}_j - \boldsymbol{x}_k], \text{ where } \boldsymbol{x}_j \in \mathcal{L}_h. \tag{4.24}$$

In the above definition, $\zeta[\cdot]$ is assumed to be $\mathcal{P}^s$-periodic. Furthermore, the resulting sequence $(\zeta \circledast \gamma_h)[\cdot]$ is also $\mathcal{P}^s$-periodic.

Let $\hat{\Gamma}(\boldsymbol{\omega})$ be the Fourier transform of the operator Γ, and $\hat{G}(\boldsymbol{\omega})$ be the DTFT of the filter γ. Furthermore, let Γ be bounded so that $\Gamma z \in L_2(\mathcal{P}^s)$. A simple modification of the measurement process (4.20) yields the desired extension of the error kernel formulation (4.22). Our main result can be summarised as follows.

Theorem 1. *Suppose that Γ is self-adjoint and shift-invariant. The Fourier error kernel that predicts the approximation error $\|(\Gamma z)_{\mathsf{app}} - \Gamma z\|_{L_2(\mathcal{P}^s)}$ is given by $E(\boldsymbol{\omega}) := E_{\mathsf{min}}(\boldsymbol{\omega}) + E_{\mathsf{mod}}(\boldsymbol{\omega})$, where the minimum error kernel $E_{\mathsf{min}}(\boldsymbol{\omega})$ is given in* (1.31)*, while the modified residue error kernel is given by*

$$E_{\mathsf{mod}}(\boldsymbol{\omega}) = \hat{A}_{\varphi}(\boldsymbol{\omega}) \big| \frac{\hat{\bar{\varphi}}(\boldsymbol{\omega}) \hat{G}(\boldsymbol{\omega})}{\hat{\Gamma}(\boldsymbol{\omega})} - \hat{\bar{\varphi}}(\boldsymbol{\omega}) \big|^2. \tag{4.25}$$

Proof. For the purpose of approximating Γz in the space spanned by the periodic generator φ_p, we need to analyze Γz in a manner similar to (4.20). Let the analysis function used to measure Γz be $\tilde{\psi}_p$, which is obtained by periodizing the function $\tilde{\psi}$. Let the resulting coefficient sequence be $\tau[\cdot]$, i.e.

$$\tau[\boldsymbol{x}_j] = \big\langle (\Gamma z)(\boldsymbol{x}), \tilde{\psi}_p(\frac{\boldsymbol{x} - \boldsymbol{x}_j}{h}) \big\rangle_{\mathcal{P}^s},$$

so that the approximation of Γz is $(\Gamma z)_{\mathsf{app}}(\boldsymbol{x}) = \sum_{\boldsymbol{x}_j \in \mathcal{P}_h} \tau[\boldsymbol{x}_j] \varphi_p(\frac{\boldsymbol{x} - \boldsymbol{x}_j}{h})$.

The error $\|(\Gamma z)_{\mathsf{app}} - \Gamma z\|_{L_2(\mathcal{P}^s)}$ can thus be predicted according to (4.21) and (4.22) with the substitutions $\hat{z} \to \widehat{(\Gamma z)}$ and $\hat{\tilde{\varphi}} \to \hat{\tilde{\psi}}$ respectively.

Now, since Γ is self-adjoint and shift-invariant, we have

$$\big\langle (\Gamma z)(\boldsymbol{x}), \tilde{\psi}_p(\frac{\boldsymbol{x}-\boldsymbol{x}_j}{h})\big\rangle_{\mathcal{P}^s} = C_h \big\langle z(\boldsymbol{x}), (\Gamma\tilde{\psi})_p(\frac{\boldsymbol{x}-\boldsymbol{x}_j}{h})\big\rangle_{\mathcal{P}^s},$$

where C_h is some constant depending on the scale h. Thus, measuring Γz with $\tilde{\psi}_p$ is equivalent to measuring z with $(\Gamma\tilde{\psi})_p$ in a distributional sense. We are interested in a digital filtering solution where $\tau[\boldsymbol{x}_j] = C_h(\zeta \circledast \gamma_h)[\boldsymbol{x}_j]$. This can be realized by requiring that $\tilde{\psi}$ satisfies $\hat{\Gamma}(\boldsymbol{\omega})\hat{\tilde{\psi}}(\boldsymbol{\omega}) = \hat{\tilde{\varphi}}(\boldsymbol{\omega})\hat{G}(\boldsymbol{\omega})$. From here, we infer that $\hat{\tilde{\psi}}(\boldsymbol{\omega}) = \frac{\hat{\tilde{\varphi}}(\boldsymbol{\omega})\hat{G}(\boldsymbol{\omega})}{\hat{\Gamma}(\boldsymbol{\omega})}$ and the result follows. □

Note that, even though the operator Γ acts in the space $L_2(\mathcal{P}^s)$, it is extended to the more general space $L_2(\mathbb{R}^s)$ in this error kernel formulation. Therefore, (4.25) can be used to quantify the L_2 error in both $L_2(\mathbb{R}^s)$ and $L_2(\mathcal{P}^s)$. Going back to the original problem (4.2), Theorem 1 gives us a way to design and analyze digital filtering solutions that approximate the analytic solution (4.10). In particular, the linear operator that we wish to discretize is Δ^{-1}. In the sequel, we show how to use the modified residue error kernel (4.25) to design filtering schemes that discretize this operator and can be efficiently implemented in the Fourier domain.

In order to characterize the order of accuracy provided by a periodic generating function φ_p, we shall switch to the more general space $L_2(\mathbb{R}^s)$. The link between the approximation properties of φ and those of its periodized version φ_p is established by the error kernel formulation introduced in the previous section. Since the same kernel can be used in both cases, henceforth, we shall reason about the approximation properties of the generator φ as the same properties are also applicable to its periodized counterpart φ_p. We use the notation $\Omega(\varphi)$ to denote the approximation order provided by the generator φ.

In many approximation scenarios, one typically assumes a Dirac point sampling model, i.e. $\tilde{\varphi}(\boldsymbol{x}) = \delta(\boldsymbol{x})$. This prevents the direct realization of the minimum approximation scenario. Furthermore, for many functions encountered in practice, the power spectrum is concentrated around $\boldsymbol{\omega} = 0$. For such scenarios, one speaks of an *asymptotically optimal* k-th order approximation scheme if the L_2 error behaves as $O(h^k)$ where $k = \Omega(\varphi)$. Our operator discretization approach is also based on these assumptions. Provided that $V \in W_2^k(\mathcal{P}^s)$, our goal is to design suitable digital filters that yield V_{app} such that $\|V - V_{\mathsf{app}}\|_{L_2(\mathcal{P}^s)} = O(h^k)$.

As discussed earlier in Section 1.2.4, the criterion that needs to be satisfied is $E_{\mathsf{mod}}(\boldsymbol{\omega}) = O(\|\boldsymbol{\omega}\|^{2k})$.

4.3.3 Relationship with the Galerkin Method

The modified residue error kernel (4.25) can also be analyzed in light of the Galerkin method [QV08] using a weak formulation of the Poisson equation (4.2), where the trial and test spaces are the same with the notable difference that the spaces are not required to explicitly satisfy any particular boundary conditions. Rather, a zero boundary condition is implicitly obtained by requiring that the solution coefficients $c[\cdot]$ be odd as explained in Section 4.2.2. Here, we establish the connection for the homogeneous case but the analysis also easily extends to the non-homogeneous case, as well as to other types of differential operators.

In its weak form, the solution $V \in L_2(\mathcal{P}^s)$ to the homogeneous Poisson equation (4.2) satisfies the weak formulation

$$\big\langle V, u \big\rangle_{\mathcal{P}^s} = \mathcal{F}(u), \;\; \forall u \in L_2(\mathcal{P}^s), \tag{4.26}$$

where the functions u are suitable test functions. The functional $\mathcal{F}(\cdot)$ is defined as $\mathcal{F}(u) := \big\langle \Delta^{-1} f, u \big\rangle_{\mathcal{P}^s}$, where f is the odd extension of the function that appears on the right hand side of (4.2), and the operator $\Delta^{-1} \Leftrightarrow \frac{1}{-\pi^2 \|\boldsymbol{m}\|^2}$ has the interpretation of the inverse Laplacian as introduced in (4.10). We remark that this formulation is slightly stronger than the usual weak formulations based on the Laplacian Δ where the trial and test spaces coincide with the Sobolev space $W_2^1(\mathcal{P}^s)$. In comparison, here the trial and test spaces coincide with the more general space $L_2(\mathcal{P}^s)$. This formulation is also in direct correspondence with our earlier treatment of the problem.

Let us now restrict attention to the finite dimensional space $\mathbb{V}(\mathcal{L}_h, \varphi_p) \subset L_2(\mathcal{P}^s)$. We seek a weak solution $V_{\mathsf{app}} \in \mathbb{V}(\mathcal{L}_h, \varphi_p)$ so that

$$\big\langle V_{\mathsf{app}}, u_h \big\rangle_{\mathcal{P}^s} = \mathcal{F}(u_h), \;\; \forall u_h \in \mathbb{V}(\mathcal{L}_h, \varphi_p), \tag{4.27}$$

and the Galerkin residual satifies the orthogonality condition

$$\big\langle V - V_{\mathsf{app}}, u_h \big\rangle_{\mathcal{P}^s} = 0, \;\; \forall u_h \in \mathbb{V}(\mathcal{L}_h, \varphi_p). \tag{4.28}$$

From this, it can be deduced that the minimum error approximation V_{app} is the orthogonal

projection of V upon $\mathbb{V}(\mathcal{L}_h, \varphi_p)$. The coefficients of the approximation are thus given by

$$c[\boldsymbol{x}_j] = \mathcal{F}(h^{-s}\mathring{\varphi}(\frac{\boldsymbol{x}-\boldsymbol{x}_j}{h})) = \big\langle f(\boldsymbol{x}), \Delta^{-1}(h^{-s}\mathring{\varphi}(\frac{\boldsymbol{x}-\boldsymbol{x}_j}{h}))\big\rangle_{\mathcal{P}^s}. \tag{4.29}$$

Observe that if we use the analysis function $\tilde{\varphi}_p = \Delta^{-1}\,\mathring{\varphi}_p$ (in a distributional sense) in the measurement equation (4.20), and subsequently employ no filtering (i.e. $\gamma[\boldsymbol{x}_j] = \delta_{\boldsymbol{x}_j}$ in (4.23)), then the measurements obtained will realize the orthogonal projection of V upon $\mathbb{V}(\mathcal{L}_h, \varphi_p)$. In this case, the Fourier residue error kernel (4.25) vanishes.

On the other hand, if the point samples of f are already available (i.e. $\tilde{\varphi} = \delta$) and the orthogonal projection cannot be realized, then the goal of the asymptotically optimal procedure is to find a suitable correction filter γ such that $E_{\mathsf{mod}}(\boldsymbol{\omega}) = O(\|\boldsymbol{\omega}\|^{2k})$ where $k = \Omega(\varphi)$. In other words, our approximation procedure attempts to reproduce the Galerkin orthogonality condition given by (4.28).

4.4 Operator Discretization

The asymptotic optimality criterion amounts to requiring that the residue error kernel asymptotically behaves like the minimum error kernel, i.e. $E_{\mathsf{mod}}(\boldsymbol{\omega}) \approx E_{\mathsf{min}}(\boldsymbol{\omega}) = O(\|\boldsymbol{\omega}\|^{2k})$ around $\boldsymbol{\omega} = 0$. In this section, we consider two discretization models: an *interpolative* model that is based on an interpolation prefilter obtained from the samples of φ, and a more accurate two-stage *quasi-interpolative* model that — like our treatment of gradient estimation in Chapter 2 — makes use of a higher order auxiliary generator ψ. Both scenarios yield digital filters that can be readily applied to the samples in the Fourier domain via the MDST.

4.4.1 Interpolative Model

Suppose that our desired approximation space is spanned by generator φ where $\Omega(\varphi) = k$. From (4.10), the operator that we need to discretize is Δ^{-1}. In order to obtain a discretization, we make use of the filter $\hat{P}_\varphi(\boldsymbol{\omega})$ that is obtained from the samples of φ that lie on the nodes of $\mathcal{L}$. Recall that the inverse filter $(\hat{P}_\varphi)^{-1}$ is needed to make the generator φ interpolating (see (1.27) in Section 1.2.3).

Let us denote the discretization of Δ^{-1} on the normalized lattice $\mathcal{L}$ as l^{-1} so that the

approximate solution on the lattice $\mathcal{L}_h$ can be written as

$$V_{\mathsf{app}}(\boldsymbol{x}) = h^2 \sum_{j=1}^{2^s N} \underbrace{(f \circledast l_h^{-1})}_{c}[\boldsymbol{x}_j]\varphi_p(\frac{\boldsymbol{x}-\boldsymbol{x}_j}{h}), \tag{4.30}$$

where the implied constant $C_h = h^2$. Using the residue error kernel (4.25), the optimality criterion boils down to requiring that

$$\hat{A}_\varphi(\boldsymbol{\omega})\big|4\pi^2\|\boldsymbol{\omega}\|^2\widehat{L^{-1}}(\boldsymbol{\omega}) + \hat{\tilde{\varphi}}(\boldsymbol{\omega})\big|^2 = O(\|\boldsymbol{\omega}\|^{2k}). \tag{4.31}$$

In order to arrive at this result, we have extended the operator Δ^{-1} to $L_2(\mathbb{R}^s)$ where $\Delta^{-1} \leftrightarrow (-4\pi^2\|\boldsymbol{\omega}\|^2)^{-1}$. The reason for choosing l^{-1} as a discretization of Δ^{-1} will become apparent in light of the following proposition which gives us a way of reformulating the problem of designing l^{-1}, a filter that discretizes Δ^{-1}, into a simpler problem of designing the filter $l := (l^{-1})^{-1}$ that discretizes the Laplacian Δ.

Proposition 2. *Let the digital filter $\hat{L}(\boldsymbol{\omega})$ be given by the combined filter $\hat{L}(\boldsymbol{\omega}) = \hat{P}_\varphi(\boldsymbol{\omega})\hat{\Lambda}(\boldsymbol{\omega})$, where $\hat{\Lambda}(\boldsymbol{\omega})$ is a digital filter that satisfies*

$$\hat{\Lambda}(\boldsymbol{\omega}) = -4\pi^2\|\boldsymbol{\omega}\|^2 + O(\|\boldsymbol{\omega}\|^{k+2}). \tag{4.32}$$

Then the combined inverse filter $\widehat{L^{-1}}(\boldsymbol{\omega}) = (\hat{P}_\varphi(\boldsymbol{\omega})\hat{\Lambda}(\boldsymbol{\omega}))^{-1}$ provides a k-th order asymptotically optimal discretization of the inverse Laplacian Δ^{-1}.

Proof. From (4.32),

$$\hat{L}(\boldsymbol{\omega}) = \hat{P}_\varphi(\boldsymbol{\omega})\hat{\Lambda}(\boldsymbol{\omega}) = -4\pi^2\hat{P}_\varphi(\boldsymbol{\omega})\|\boldsymbol{\omega}\|^2 + O(\|\boldsymbol{\omega}\|^{k+2}).$$

After some rearrangement and using the fact that $\hat{L}(\boldsymbol{\omega}) = O(\|\boldsymbol{\omega}\|^2)$, we obtain

$$-\frac{4\pi^2\|\boldsymbol{\omega}\|^2}{\hat{L}(\boldsymbol{\omega})} = \frac{1}{\hat{P}_\varphi(\boldsymbol{\omega})} + O(\|\boldsymbol{\omega}\|^k),$$

which satisifies the asymptotic optimality criterion (4.31) since we know from Section 1.2.3 that $(\hat{P}_\varphi(\boldsymbol{\omega}))^{-1} = \hat{\tilde{\varphi}}(\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^k)$. □

It is not hard to see that this method is essentially a collocation method with respect to the trial space spanned by the generator φ_p. If we apply the discrete Laplacian filter $\lambda \leftrightarrow \hat{\Lambda}$ to the coefficients in approximation (4.30), we obtain

$$(\Delta\, V_{\mathsf{app}})(\boldsymbol{x}) \approx \sum_{j=1}^{2^s N} (f \circledast p_\varphi^{-1})[\boldsymbol{x}_j]\varphi_p(\frac{\boldsymbol{x}-\boldsymbol{x}_j}{h}), \tag{4.33}$$

where $p_\varphi^{-1}[\cdot] \leftrightarrow (\hat{P}_\varphi(\cdot))^{-1}$. Owing to the fact that p_φ^{-1} is an interpolation filter, this approximation, when evaluated at the lattice sites contained in $\mathcal{C}_o^s$, will yield the samples of f.

4.4.2 Quasi-interpolative Model

We now describe a two-stage model that relies on two spaces: a desired target space $\mathbb{V}(\mathcal{L}_h, \varphi_p)$ spanned by the generator φ_p where $\Omega(\varphi) = k$, and an auxiliary space $\mathbb{V}(\mathcal{L}_h, \psi_p)$ spanned by the generator ψ_p where $\Omega(\psi) \geq k$. In the first stage, we use the interpolative model from the previous section to seek an auxiliary solution that lies in $\mathbb{V}(\mathcal{L}_h, \psi_p)$. Then, the auxiliary solution is orthogonally projected to the target space $\mathbb{V}(\mathcal{L}_h, \varphi_p)$. This method is similar to our two-stage gradient estimation framework presented in Chapter 2 (see also Figure 2.1 on page 31). Owing to the shift-invariant nature of the spaces, the orthogonal projection is tantamount to applying a digital filter to the first-stage approximation coefficients as described below.

Let us denote the first-stage approximation coefficients as $c_\psi[\boldsymbol{x}_j]$. These coefficients are obtained by applying the discrete operator l_ψ^{-1} to the samples of f as given in (4.30). The discretization l_ψ^{-1} is obtained from the interpolative model where the interpolation prefilter p_ψ is derived from the samples of the first-stage generator ψ. Now, if ψ is chosen to have a sufficiently high order, then we know that this auxiliary approximation will be very close to the true solution. Therefore, we can use it to find an approximation in our desired target space $\mathbb{V}(\mathcal{L}_h, \varphi)$. In particular, we orthogonally project the auxiliary approximation onto the target space so as to minimize the L_2 error. The orthogonal projection is obtained by taking inner products of the auxiliary approximation with the lattice translates of $\mathring{\varphi}_p$: the dual of the target space generator φ_p. In particular, the second-stage approximation can be written as

$$V_{\mathsf{app}}(\boldsymbol{x}) = \sum_{j=1}^{2^s N} \Big(\sum_{k=1}^{2^s N} c_\psi \big\langle \psi_p(\frac{\cdot - \boldsymbol{x}_k}{h}), h^{-s} \mathring{\varphi}_p(\frac{\cdot - \boldsymbol{x}_j}{h}) \big\rangle_{\mathcal{P}^s} \Big) \varphi_p(\frac{\boldsymbol{x} - \boldsymbol{x}_j}{h}). \tag{4.34}$$

Assuming that ψ and φ are symmetric admissible generators, this is equivalent to the digital filtering scheme

$$V_{\mathsf{app}}(\boldsymbol{x}) = \sum_{j=1}^{2^s N} (c_\psi \circledast q_h)[\boldsymbol{x}_j] \varphi_p(\frac{\boldsymbol{x} - \boldsymbol{x}_j}{h}), \tag{4.35}$$

where the weights of the filter q_h are obtained according to

$$q_h[h\mathbf{L}\boldsymbol{k}] = q[\mathbf{L}\boldsymbol{k}] := (\psi * \mathring{\varphi})(\mathbf{L}\boldsymbol{k}). \tag{4.36}$$

Here, $(\psi * \mathring{\varphi})$ denotes the continuous convolution of the two generators ψ and $\mathring{\varphi}$ in $L_2(\mathbb{R}^s)$.

As we saw in Section 1.2.3, an approximation model is said to be quasi-interpolating of order k if it is capable of reproducing all multivariate polynomials of degree $(k-1)$, or equivalently, $E_{\mathsf{res}}(\boldsymbol{\omega}) = O(\|\boldsymbol{\omega}\|^{2m})$ $(m \geq k)$ where E_{res} is as defined in (1.31). We retain the same terminology and say that an approximate solution to the Poisson equation (4.2) is quasi-interpolating of order k if it satisfies $E_{\mathsf{mod}}(\boldsymbol{\omega}) = O(\|\boldsymbol{\omega}\|^{2m})$, where $m \geq k$. The following proposition gives us a way of choosing the auxiliary space $\mathbb{V}(\mathcal{L}_h, \psi_p)$ so as to ensure that the resulting approximation (4.35) is quasi-interpolating of order k.

Proposition 3. *Let $\mathbb{V}(\mathcal{L}_h, \psi_p)$ and $\mathbb{V}(\mathcal{L}_h, \varphi_p)$ be the auxiliary and target approximation spaces respectively, with $\Omega(\psi) = m$ and $\Omega(\varphi) = k$. The approximation V_{app} obtained through (4.35) is quasi-interpolating of order k if $m \geq k$.*

Proof. The modified residue error kernel for this approximation scheme is given by

$$E_{\mathsf{mod}}(\boldsymbol{\omega}) = \hat{A}_{\varphi}(\boldsymbol{\omega})\Big|\frac{4\pi^2\|\boldsymbol{\omega}\|^2}{\hat{L}_{\psi}(\boldsymbol{\omega})}\hat{Q}(\boldsymbol{\omega}) + \hat{\mathring{\varphi}}(\boldsymbol{\omega})\Big|^2,$$

where $l_{\psi}^{-1} \leftrightarrow 1/\hat{L}_{\psi}$ denotes the filter obtained from Section 4.4.1 with ψ as the generator. Given that $m \geq k$, we need to show that $-\frac{4\pi^2\|\boldsymbol{\omega}\|^2}{\hat{L}_{\psi}(\boldsymbol{\omega})}\hat{Q}(\boldsymbol{\omega}) = \hat{\mathring{\varphi}}(\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^n)$ for some $n \geq k$.

Now, from the proof of Proposition 2 we know that $-\frac{4\pi^2\|\boldsymbol{\omega}\|^2}{\hat{L}_{\psi}(\boldsymbol{\omega})} = \hat{\mathring{\psi}}(\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^m)$. Furthermore, using (4.36) and the fact that $\Omega(\psi) = m$ and $\Omega(\varphi) = k$, we have $\hat{Q}(\boldsymbol{\omega}) = \hat{\psi}(\boldsymbol{\omega})\hat{\mathring{\varphi}}(\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^{m+k})$. Therefore,

$$\begin{aligned}-\frac{4\pi^2\|\boldsymbol{\omega}\|^2}{\hat{L}_{\psi}(\boldsymbol{\omega})}\hat{Q}(\boldsymbol{\omega}) &= \big(\hat{\mathring{\psi}}(\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^m)\big)\big(\hat{\psi}(\boldsymbol{\omega})\hat{\mathring{\varphi}}(\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^{m+k})\big) \\ &= \hat{\psi}(\boldsymbol{\omega})\hat{\mathring{\psi}}(\boldsymbol{\omega})\hat{\mathring{\varphi}}(\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^m) = \hat{\mathring{\varphi}}(\boldsymbol{\omega}) + O(\|\boldsymbol{\omega}\|^m),\end{aligned}$$

since $\hat{\psi}$, $\hat{\mathring{\psi}}$ and $\hat{\mathring{\varphi}}$ are all $O(1)$ around $\boldsymbol{\omega} = \mathbf{0}$. □

We note that when the auxiliary space is chosen to be the same as the target space, i.e. when $\psi = \varphi$, then $q[\mathbf{L}\boldsymbol{k}] = \delta_{\boldsymbol{k}}$ and this scheme reduces to the interpolative model described in the previous section.

4.5 Numerical Experiments

We now present some numerical experiments on the 2D Cartesian lattice (Fig. 4.1a), and the 3D CC and BCC lattices. Since we are interested in a digital filtering solution, we first describe a circular convolution scheme for the Cartesian lattice that can be efficiently implemented in the Fourier domain via a separable MDST. The topic of efficiently implementing the MDST on the BCC lattice is discussed in detail in Chapter 5 (see Section 5.3 on page 95). We then turn to the problem of designing asymptotically optimal filters that are suitable for use with shift-invariant spaces generated by the tensor product B-splines (2D and 3D Cartesian) and the rhombic dodecahedral box splines (BCC).

4.5.1 Circular Convolution on the 2D Cartesian Lattice

Since our solution methodology is convolution based, we look for an efficient way to implement the cyclic convolution operation (4.24) on the 2D Cartesian lattice. Recall that the cyclic convolution of two periodic sequences in the spatial domain amounts to a pointwise multiplication of their MDFTs. In our case, the sampled sequence $f[\boldsymbol{x}_j]$ is known only for the interior points that are contained within the open unit square $\mathcal{C}_o^2$ (i.e. $j \in \{1, \ldots, N\}$). The remaining sequence is inferred from oddity. If t is a symmetric filter, then the result of the convolution will also be odd. It can therefore be computed efficiently by using the MDST (4.18).

For the 2D Cartesian lattice, it is convenient to enumerate the N points contained within $\mathcal{C}_o^2$ as $\{(\frac{j_1}{M+1}, \frac{j_2}{M+1}) : 1 \le j_1, j_2 \le M\}$ so that $N = M^2$ and $h = 1/(M+1)$. We also drop the dependence on M when denoting the sequence f and simply index it as $f[j_1, j_2]$.

The MDST (4.18) of f can now be written in a separable manner as

$$\tilde{F}[m_1, m_2] = \sum_{j_1, j_2 = 1}^{M} f[j_1, j_2] \sin(\pi \frac{j_1 m_1}{M+1}) \sin(\pi \frac{j_2 m_2}{M+1}), \quad \text{for } 1 \le m_1, m_2 \le M. \tag{4.37}$$

The MDST coefficients of the resulting sequence $c = (f \circledast t_h)$ can be computed from the MDST of the sequence f and the DTFT of the filter t_h according to

$$\tilde{C}[m_1, m_2] = \tilde{F}[m_1, m_2] \hat{T}(\frac{m_1}{2(M+1)}, \frac{m_2}{2(M+1)}). \tag{4.38}$$

The resulting sequence c is now given by the inverse MDST of $\tilde{C}$, i.e.

$$c[j_1, j_2] = \frac{4}{(M+1)^2} \sum_{m_1, m_2 = 1}^{M} \tilde{C}[m_1, m_2] \sin(\pi \frac{j_1 m_1}{M+1}) \sin(\pi \frac{j_2 m_2}{M+1}), \tag{4.39}$$

for $1 \leq j_1, j_2 \leq M$. The remaining sequence is inferred from oddity. Since the MDST (4.37) is a tensor product of type-I DSTs [Mar94], it can be implemented efficiently via the FFT [FJ05]. This recipe also easily extends to the 3D CC lattice.

4.5.2 Filter Design

2D Cartesian Lattice

We now turn to the problem of designing filters that discretize the operator $\Delta^{-1} \leftrightarrow (-4\pi^2(\omega_1^2 + \omega_2^2))^{-1}$, and are to be used in conjunction with the tensor-product B-splines. Recall that the space $\mathbb{V}(\mathbb{Z}_h^2, B^k)$ generated by $B^k(x_1, x_2) = \beta^k(x_1)\beta^k(x_2)$ (where β^k is as defined in (1.39) on page 19), satisfies the approximation property: $\Omega(B^k) = k + 1$. This result also extends to the Cartesian lattice in higher dimensions. Consequently, we cover the 2D case in detail here. The extension to 3D is left to the reader.

For the generator B^k, we are interested in a $(k + 1)$-th order asymptotically optimal discretization of the inverse Laplacian Δ^{-1}. According to Proposition 2, we need the two filters p_{B^k} and λ.

Since B^k is a compact generator, p_{B^k} will also be a compact filter. Furthermore, since B^k is separable, p_{B^k} can be determined by sampling $\beta^k(x)$ to yield the filter $p_{\beta^k}[n] = \beta^k(n)$ where $n \in \mathbb{Z}$. The weights of p_{B^k} are then obtained by a simple tensor product, i.e. $p_{B^k}[n_1, n_2] = p_{\beta^k}[n_1]p_{\beta^k}[n_2] \leftrightarrow \hat{P}_{B^k}(\omega_1, \omega_2) = \hat{P}_{\beta^k}(\omega_1)\hat{P}_{\beta^k}(\omega_2)$.

As for the filter λ that discretizes Δ, we follow a similar approach and first look for a symmetric and compact 1D filter λ_1 that satisfies the 1D analog of (4.31), i.e. $\hat{\Lambda}_1(\omega) = -4\pi^2\omega^2 + O(\omega^{k+3})$. The unknown filter weights are determined by requiring that the Taylor developments of $(\hat{\Lambda}_1(\omega) + 4\pi^2\omega^2)$ be zero up to ω^{k+3}. The filter $\lambda \leftrightarrow \hat{\Lambda}$ is then given by a simple addition, i.e. $\hat{\Lambda}(\omega_1, \omega_2) = \hat{\Lambda}_1(\omega_1) + \hat{\Lambda}_1(\omega_2)$.

The sizes of p_{β^k} and λ_1 obviously depend on the degree k. The filters used in our experiments are tabulated in Table 4.1.

Quasi-Interpolation. For the quasi-interpolative model, we need the additional filter q (4.36). Choosing ψ to be the tensor product B-spline B^l where $l > k$, the weights of q are given by the tensor product of the 1D filter $q_1[n] := (\beta^l * \mathring{\beta}^k)(n)$ with itself. Using the recursive definition of the 1D B-splines, q_1 can be further split as $q_1[n] = (w_1 * a_{\beta^k}^{-1})[n]$ where $w_1[n] := \beta^{k+l+1}(n)$ and $a_{\beta^k}^{-1}[\cdot] \leftrightarrow \frac{1}{\hat{A}_{\beta^k}(\cdot)}$. Example filter weights for the case $l = 5$ are provided in Table 4.1.

Table 4.1: The various 1D filters to be used in conjunction with bilinear and bicubic approximation.

	$\mathbb{V}(\mathbb{Z}_h^2, B^1)$	$\mathbb{V}(\mathbb{Z}_h^2, B^3)$	
	Interp.	Interp.	Quasi. ($l=5$)
$p_{\beta^k}[n]$	δ_n	$[\frac{1}{6}\ \frac{2}{3}\ \frac{1}{6}]$	$[\frac{1}{120}\ \frac{13}{60}\ \frac{11}{20}\ \frac{13}{60}\ \frac{1}{120}]$
$\hat{P}_{\beta^k}(\frac{\nu}{2\pi})$	1	$\frac{1}{3}(2+\cos(\nu))$	$\frac{1}{60}(33+26\cos(\nu)+\cos(2\nu))$
$\lambda_1[n]$	$[1\ -2\ 1]$	$[\frac{-1}{12}\ \frac{4}{3}\ \frac{-5}{2}\ \frac{4}{3}\ \frac{-1}{12}]$	$[\frac{1}{90}\ \frac{-3}{20}\ \frac{3}{2}\ \frac{-49}{18}\ \frac{3}{2}\ \frac{-3}{20}\ \frac{1}{90}]$
$\hat{\Lambda}_1(\frac{\nu}{2\pi})$	$-2+2\cos(\nu)$	$\frac{2}{3}(-7+\cos(\nu))\sin^2(\frac{\nu}{2})$	$\frac{2}{45}(-111+23\cos(\nu)-2\cos(2\nu))\sin^2(\frac{\nu}{2})$
$w_1[n]$			$[\frac{1}{362880}\ \frac{251}{181440}\ \frac{913}{22680}\ \frac{44117}{181440}\ \frac{15619}{36288}\ \cdots]$
$\hat{W}_1(\frac{\nu}{2\pi})$			$\frac{1}{181440}(78095+88234\cos(\nu)+14608\cos(2\nu)+502\cos(3\nu)+\cos(4\nu))$
$a_{\beta^k}[n]$			$[\frac{1}{5040}\ \frac{1}{42}\ \frac{397}{1680}\ \frac{151}{315}\ \frac{397}{1680}\ \frac{1}{42}\ \frac{1}{5040}]$
$\hat{A}_{\beta^k}(\frac{\nu}{2\pi})$			$\frac{1}{2520}(1208+1191\cos(\nu)+120\cos(2\nu)+\cos(3\nu))$

BCC Lattice

For convenience, we work with the (scaled) BCC lattice $\mathcal{H}$ that is generated by the matrix $\mathbf{H}$ given by (2.10) on page 34, and employ the rhombic dodecahedral box splines as generators. Recall that box spline ϑ^k, as defined by (2.11) on page 35, satisfies $\Omega(\vartheta^k) = k$ (for even k) with respect to the space $\mathbb{V}(\mathcal{H}_h, \vartheta^k)$.

The operator we wish to discretize is $\Delta^{-1} \leftrightarrow -4\pi^2(\omega_1^2+\omega_2^2+\omega_3^2)^{-1}$. According to Proposition 2, we need the combined filter $(p_{\vartheta^k} * \lambda)$ where $\lambda \leftrightarrow \hat{\Lambda}$ satisfies (4.32), i.e. $\hat{\Lambda}(\boldsymbol{\omega}) = -4\pi^2\|\boldsymbol{\omega}\|^2 + O(\|\boldsymbol{\omega}\|^{k+2})$. The compact filter p_{ϑ^k} is non-separable and is determined by the samples of ϑ^k, i.e. $p_{\vartheta^k}[\boldsymbol{x}_j] = \vartheta^k(\boldsymbol{x}_j)$ where $\boldsymbol{x}_j \in \mathcal{H}$.

As for the filter λ, observe that

$$\|\boldsymbol{\omega}\|^2 = \frac{1}{4}\sum_{i=1}^{4}(\boldsymbol{\theta}_i{}^{\mathsf{T}}\boldsymbol{\omega})^2, \tag{4.40}$$

where the (scaled) box spline direction vectors $\boldsymbol{\theta}_i$ ($i \in \{1,2,3,4\}$) correspond to the column vectors $(1,1,-1)^{\mathsf{T}}$, $(1,-1,1)^{\mathsf{T}}$, $(-1,1,1)^{\mathsf{T}}$ and $(-1,-1,-1)^{\mathsf{T}}$ respectively (also see (1.40) on page 20). Since $\boldsymbol{\theta}_i \in \mathcal{H}$, we can re-use the 1D filter $\lambda_1[n]$ from the previous section and apply it across the directions $\boldsymbol{\theta}_i$ to yield the filter λ. Therefore, $\lambda \leftrightarrow \hat{\Lambda}$ is given by

$$\hat{\Lambda}(\boldsymbol{\omega}) = \frac{1}{4}\sum_{i=1}^{4}\hat{\Lambda}_1(\boldsymbol{\theta}_i{}^{\mathsf{T}}\boldsymbol{\omega}), \tag{4.41}$$

where the 1D filter $\lambda_1 \leftrightarrow \hat{\Lambda}_1$ satisfies $\hat{\Lambda}_1(\omega) = -4\pi^2\omega^2 + O(\omega^{k+2})$.

Table 4.2 summarizes the various filters that are the BCC analogs of Cartesian filters listed in Table 4.1.

Quasi-Interpolation. We need the additional filter q as given by (4.36). Choosing ψ to be the rhombic dodecahedral box spline ϑ^l where $l > k$, the weights of q are given by the BCC samples of $(\vartheta^l * \mathring{\vartheta}^k)$. Like the Cartesian case, q can be split up as $q[\boldsymbol{x}_j] = (w_{\mathsf{b}} * a^{-1}_{\vartheta^k})[\boldsymbol{x}_j]$ where $w_{\mathsf{b}}[\boldsymbol{x}_j] = \vartheta^{k+l}(\boldsymbol{x}_j)$ and $a^{-1}_{\vartheta^k}[\cdot] \leftrightarrow \frac{1}{\hat{A}_{\vartheta^k}(\cdot)}$. The case $l = 6$ is shown in Table 4.2.

4.5.3 Results

We conducted numerical tests to verify our solution methodology for the various cases presented in Tables 4.1 and 4.2. We started with a known synthetic function V and used the samples of its Laplacian $f = \Delta V$ inside the unit cube $\mathcal{C}^s$ $(s \in \{2, 3\})$ to obtain the approximations V_{app} for various grid sizes. On the 2D and 3D Cartesian lattices, the grid size was controlled by an integer parameter M which determines the number of samples along one dimension; therefore, the total number of interior samples is $N = M^s$, and the scaling parameter $h = \frac{1}{M+1}$. On the (scaled) BCC lattice $\mathcal{H}_h$, we used the paramter M to determine the scaling paramter h so that the BCC lattice has roughly M^3 interior samples. The paramter h is given by

$$h = \frac{1}{2M_{\mathsf{b}}}, \quad \text{where } M_{\mathsf{b}} = \lfloor \frac{M+1}{\sqrt[3]{2}} \rfloor, \tag{4.42}$$

and the total number of interior samples is $N = (M_{\mathsf{b}} - 1)^3 + M_{\mathsf{b}}^3$.

We used a Monte-Carlo procedure to determine the approximation error. In particular, the error measure used is given by

$$\epsilon_2 := \left(\frac{1}{\mathsf{T}} \sum_{i=1}^{\mathsf{T}} (V(\bar{\boldsymbol{x}}_i) - V_{\mathsf{app}}(\bar{\boldsymbol{x}}_i))^2 \right)^{\frac{1}{2}}, \tag{4.43}$$

where the random variable $\bar{\boldsymbol{x}}_i$ is uniformly distributed inside the unit cube. This RMS error measure approximates $\|V - V_{\mathsf{app}}\|_{L_2(\mathcal{C}^s)}$. We also computed the maximum error according to

$$\epsilon_\infty := \max_{i=1}^{\mathsf{T}} \big| V(\bar{\boldsymbol{x}}_i) - V_{\mathsf{app}}(\bar{\boldsymbol{x}}_i) \big|, \tag{4.44}$$

which approximates $\|V - V_{\mathsf{app}}\|_{L_\infty(\mathcal{C}^s)}$. For all the 2D experiments, $\mathsf{T} = 10^6$, while for the 3D experiments $\mathsf{T} = 10^7$.

Table 4.2: Filters to be used in conjunction with the linear and quintic box splines on the BCC lattice. $\mathrm{c}(\theta)$ is short for $\cos(\theta)$.

	$\mathbb{V}(\mathcal{H}_h, \vartheta^2)$	$\mathbb{V}(\mathcal{H}_h, \vartheta^4)$	
	Interp.	Interp.	Quasi. ($l = 6$)
$p_{\vartheta^k}[\boldsymbol{y}]$ $(\boldsymbol{y} \in \mathcal{H})$	$\delta_{\boldsymbol{y}}$	$\begin{cases} \frac{2}{5} & \text{if } \boldsymbol{y} = \boldsymbol{0}, \\ \frac{1}{20} & \text{if } \|\boldsymbol{y}\| = \sqrt{3}, \\ \frac{1}{30} & \text{if } \|\boldsymbol{y}\| = 2, \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} \frac{379}{1680} & \text{if } \boldsymbol{y} = \boldsymbol{0}, \\ \frac{1177}{20160} & \text{if } \|\boldsymbol{y}\| = \sqrt{3}, \\ \frac{7}{180} & \text{if } \|\boldsymbol{y}\| = 2, \\ \frac{43}{10080} & \text{if } \|\boldsymbol{y}\| = \sqrt{8}, \\ \frac{17}{20160} & \text{if } \|\boldsymbol{y}\| = \sqrt{11}, \\ \frac{1}{4032} & \text{if } \|\boldsymbol{y}\| = \sqrt{12}, \\ \frac{1}{10080} & \text{if } \|\boldsymbol{y}\| = 4, \\ 0 & \text{otherwise} \end{cases}$
$\hat{P}_{\vartheta^k}(\frac{u}{2\pi}, \frac{v}{2\pi}, \frac{w}{2\pi})$	1	$\frac{1}{15}\big(6 + \mathrm{c}(2u) + \mathrm{c}(2v) + \mathrm{c}(2w) + 6\,\mathrm{c}(u)\,\mathrm{c}(v)\,\mathrm{c}(w)\big)$	$\frac{1}{5040}\Big(2\,\mathrm{c}(u)\,\mathrm{c}(v)\,\mathrm{c}(w)\big[34\,\mathrm{c}(2u) + 34\,\mathrm{c}(2v) + 1143\big] + 2\,\mathrm{c}(2w)\big[\mathrm{c}(2u)\big(5\,\mathrm{c}(2v) + 43\big) + 43\,\mathrm{c}(2v) + 196\big] + 34\,\mathrm{c}(u)\,\mathrm{c}(v)\,\mathrm{c}(3w) + 43\,\mathrm{c}\big(2(u - v)\big) + 43\,\mathrm{c}\big(2(u + v)\big) + 392\,\mathrm{c}(2u) + \mathrm{c}(4u) + 392\,\mathrm{c}(2v) + \mathrm{c}(4v) + \mathrm{c}(4w) + 1137\Big)$
$\lambda_1[n]$ $\hat{\Lambda}_1(\frac{\nu}{2\pi})$			see Table 4.1
$w_{\mathsf{b}}[\boldsymbol{y}]$ $\hat{W}_{\mathsf{b}}(\boldsymbol{\omega})$	N/A	N/A	see Appendix A.1
$a_{\vartheta^k}[\boldsymbol{y}]$ $\hat{A}_{\vartheta^k}(\boldsymbol{\omega})$	N/A	N/A	see Appendix A.2

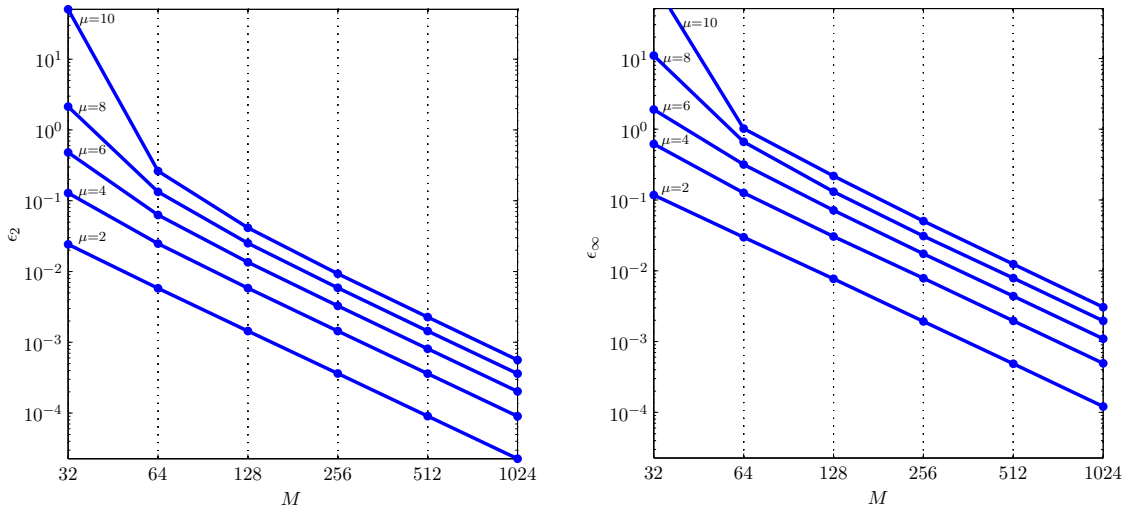


Figure 4.3: Effect of the frequency paramter μ on bilinear approximation

For all our tests, we used the synthetic function

$$V(\boldsymbol{x}) := \sin\big(2\pi\mu \textstyle\prod_{i=1}^{s} \sin(\pi x_i)\big), \quad \text{where } s \in \{2,3\}. \tag{4.45}$$

This function satisfies zero Dirichlet boundary conditions. The parameter μ can be varied to control the oscillation frequency and hence the rate of decay of the Fourier spectrum.

2D

Figure 4.3 shows the effect of the frequency parameter μ on the rate of error decay for the space $\mathbb{V}(\mathbb{Z}_h^2, B_p^1)$ that is generated by the bilinear B-spline B^1. All the curves exhibit a second-order trend, which is the approximation order provided by the bilinear B-spline. The parameter μ only affects the overall error; it does not affect the asymptotic decay rate. The L_∞-error also exhibits a second-order decay.

Figure 4.4 compares the decay rate associated with nearest-neighbor ($\mathbb{V}(\mathbb{Z}_h^2, B_p^0)$), bilinear ($\mathbb{V}(\mathbb{Z}_h^2, B_p^1)$), and bicubic ($\mathbb{V}(\mathbb{Z}_h^2, B_p^3)$) approximation schemes. The nearest-neighbor approximation scheme is obtained by combining the second-order filters associated with the bilinear B-spline (Table 4.1) with the Voronoi spline B^0. Since $\Omega(B^0) = 1$, the nearest-neighbor scheme exhibits a first-order decay. In contrast, the interpolative bilinear (resp. bicubic) scheme shows a first (resp. fourth) order decay, and fully exploits the approximation capabilities provided by B^1 (resp. B^3). The quasi-interpolative bicubic approximation

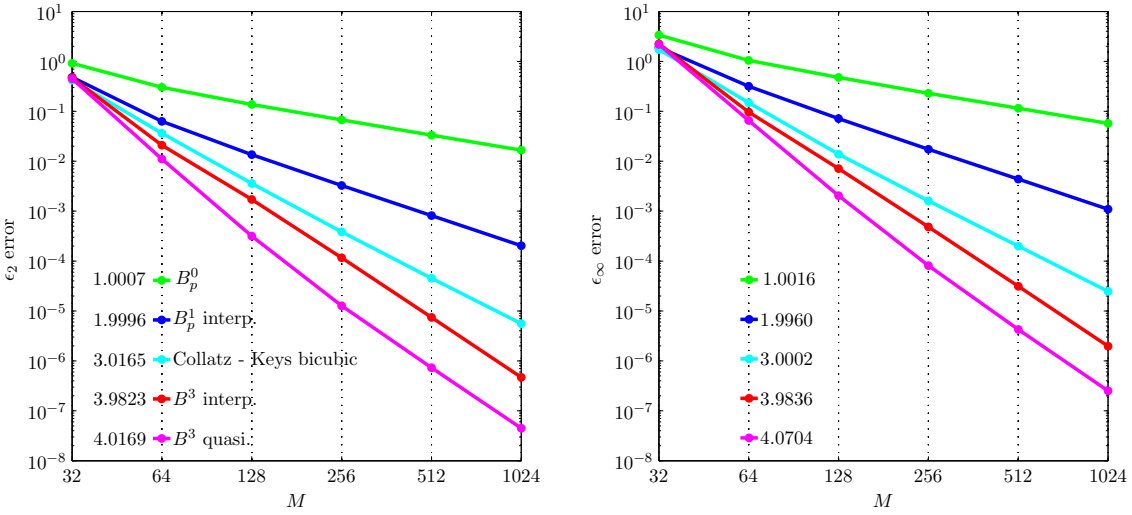


Figure 4.4: Comparison of 2D approximation schemes. The asymptotic approximation order (as determined from the last two data points) is indicated.

scheme has a decay rate that is comparable to the interpolative bicubic approximation scheme. However, the overall error is much lower due to the higher order of the first stage approximation space. The L_∞-error curves exhibit a similar trend as their L_2 counterparts which suggests that our approximation methodology can also be extended to other L_p norms.

Figure 4.4 also shows the results obtained by combining the well-known Collatz stencil [Col60] with Keys' bicubic interpolation [Key81]. This scheme can be easily cast into our framework for implementation and analysis purposes. The Collatz stencil Λ provides an implicit finite difference discretization of Δ. Its DTFT is given by

$$\hat{\Lambda}(\frac{\nu_1}{2\pi}, \frac{\nu_2}{2\pi}) = \frac{4(-5 + 2\cos(\nu_1) + 2\cos(\nu_2) + \cos(\nu_1)\cos(\nu_2))}{4 + \cos(\nu_1) + \cos(\nu_2)}. \tag{4.46}$$

It is easy to check that this filter satisfies (4.32) and provides a fourth-order discretization of the Laplacian Δ. An interpolation prefilter is not necessary since the Keys' bicubic generator is already interpolating. Note that the cost of Keys' bicubic approximation is the same as bicubic B-spline approximation since both generators have the same support size. However, unlike the fourth-order cubic B-spline β^3, the Keys' bicubic generator is only third-order [BTU01, MU03]. This limits the overall approximation order of the *Collatz - Keys* scheme to 3.

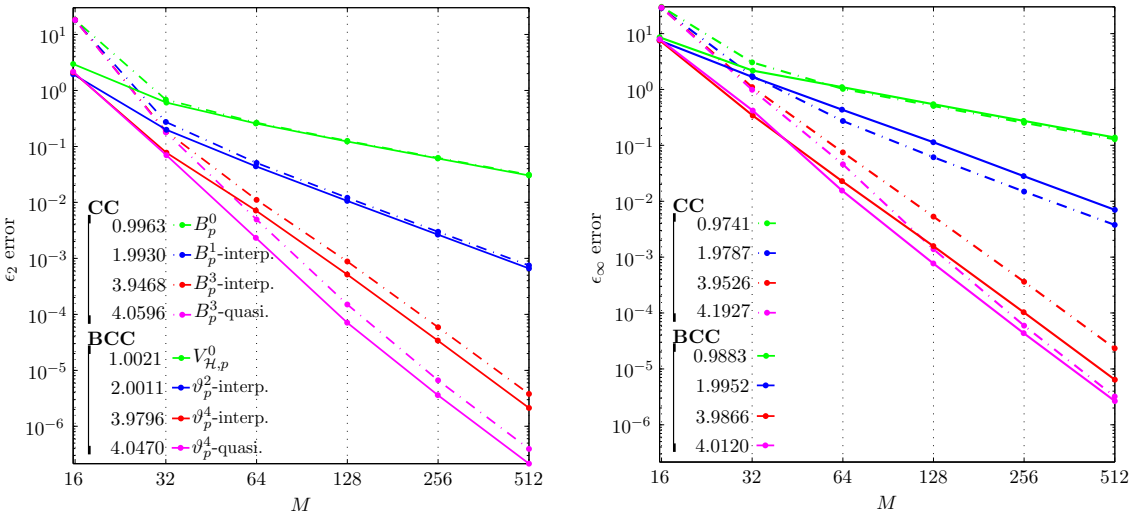


Figure 4.5: Comparison of 3D approximation schemes. The asymptotic approximation order (as determined from the last two data points) is indicated.

3D

We also compared the approximation capabilities of the CC and BCC lattices using the tensor-product extensions of the 1D filters listed in Table 4.1 (CC), and the non-separable 3D filters listed in Table 4.2 (BCC). We used a 3D version of the synthetic test function V as defined in (4.45). Our results are shown in Figure 4.5. Like the 2D experiments, the nearest-neighbor scheme is obtained by the combining the second-order filter obtained from an interpolative model, with the corresponding first-order Voronoi spline.

As expected, the Voronoi splines (B_p^0 on CC and $V_{\mathcal{H},p}^0$ on BCC) exhibit a first-order convergence, the trilinear B-spline B_p^1 on CC and the linear box spline ϑ_p^2 on BCC have a second-order convergence, while the tricubic B-spline B_p^3 on CC and the quinitic box spline ϑ_p^4 on BCC reveal a fourth-order convergence rate. In terms of the L_2-error, the BCC lattice outperforms across the board. The approximation gain also becomes more pronounced as the order increases. As we saw in 2D, owing to the higher order of the first-stage space, the quasi-interpolative model reduces the error considerably. These results are also in agreement with our earlier comparison of these spaces (see Figure 1.3 on page 23). The only exception is the second-order case where the interpolative BCC model has a slight edge of its CC counterpart.

The L_∞ error decay also follows a similar trend. However, for the second-order spaces,

there is a remarkable difference between the trilinear B-spline on CC and the linear box spline on BCC. Linear box spline interpolation on the BCC lattice is equivalent to a barycentric interpolation over a tetrahedralized BCC grid. As pointed out by Carr *et al.* [CMS06], such an interpolation scheme leads to local girdering artifacts which deteriorate the performance with respect to the L_∞ error metric.

4.6 Notes

4.6.1 Relationship between the Modified Residue Kernel and the Derivative Error Kernel

The modified residue error kernel (see (4.25) on page 73) can also be used to quantify the error incurred when approximation directional derivatives. Indeed, under a point-sampling model ($\hat{\tilde{\varphi}} = 1$), if Γ is the directional derivative operator ($\hat{\Gamma} = 2\pi\imath \boldsymbol{l}^\mathsf{T}\boldsymbol{\omega}$) and the filter $d \leftrightarrow \hat{D}$ is applied to the point samples, (4.25) is equivalent to the residue term $E^{\boldsymbol{l}}_{\mathsf{res}}$ in the derivative error kernel (see (3.4) on page 50) of Condat and Möller [CM11] that was derived from different principles. Therefore, Theorem 1 also provides an alternate derivation of the derivative error kernel.

4.6.2 Extensibility

Our solution methodology can also be extended to the non-homogeneous case as well as to other types of PDEs. We briefly highlight two straightforward extensions.

Non-homogeneous Dirichlet Boundary Conditions

When the boundary conditions are non-homogeneous, the Poisson equation (4.2) becomes:

$$\begin{aligned} \Delta V &= f, \ \text{ in } \ \mathcal{C}^s_o, \\ V &= g, \ \text{ on } \ \partial\mathcal{C}^s. \end{aligned} \tag{4.47}$$

The analytic solution is given by (see Marshall [Mar99])

$$V(\boldsymbol{x}) = \int_{\mathcal{C}^s} f(\boldsymbol{y})\mathsf{G}(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y} + \int_{\partial\mathcal{C}^s} g(\boldsymbol{y})\frac{\partial \mathsf{G}}{\partial n}(\boldsymbol{x},\boldsymbol{y})dS(\boldsymbol{y}), \tag{4.48}$$

where G is the Green's function given in (4.4) and $\frac{\partial \mathsf{G}}{\partial n}$ denotes the derivative of G in the direction of the outward unit normal. Since $\frac{\partial \mathsf{G}}{\partial n}$ is always axis-aligned, one can easily obtain

a Fourier domain expression for the analytic solution and identify the additional operators that need to be discretized in order to account for the contribution due to the boundary. However, unlike the homogeneous case, the odd extension of the non-homogeneous case is discontinuous at the boundary $\partial\mathcal{C}^s$, and some care is needed to avoid Gibb's oscillations due to the discontinuity. This is a subject of future research.

Homogeneous Neumann boundary conditions can be handled by changing the sine terms in the Green's function (4.4) to cosine terms [Mar99]. Since our solution methodology is convolution based, we can account for this modification by using the multidimensional discrete cosine transform (MDCT) instead of the MDST.

Helmholtz Equation

The Helmholtz equation, under periodic boundary conditions, is given by

$$\Delta V + \alpha^2 V = f, \quad \text{in } \mathcal{C}^s. \tag{4.49}$$

Expanding both sides in a Fourier series, the solution in the Fourier domain is given by

$$\hat{V}[\boldsymbol{m}] = \frac{\hat{f}[\boldsymbol{m}]}{\alpha^2 - 4\pi^2\imath\|\boldsymbol{m}\|^2}, \quad \text{where } \boldsymbol{m} \in \mathbb{Z}^s.$$

From here, we easily infer that the operator to be discretized is $\left(\alpha^2 - 4\pi^2\imath\|\boldsymbol{m}\|^2\right)^{-1}$. Applying our solution methodology to the discretization of this operator is straightforward. Dirichlet (resp. Neumann) boundary conditions can be handled by using the MDST (resp. MDCT) instead of the MDFT.

Chapter 5

Discrete Fourier and Sine Transforms with Rectangular Output on the BCC Lattice

5.1 Introduction

This chapter discusses the efficient, non-redundant evaluation of the MDFT on the BCC lattice. Most applications of the DFT in higher dimensions rely on a tensor-product extension of a one-dimensional DFT, with the assumption that the underlying data is sampled on a Cartesian lattice. This extension has the advantage that it allows for a straightforward application of the FFT [FJ05]. Here, we show that the FFT can also be used to efficiently compute the MDFT on the BCC lattice. The key idea is to use an axis-aligned window to truncate and periodize the sampled function which leads to separable transforms. We exploit the geometry of the BCC lattice and identify a suitable non-redundant rectangular region in the frequency domain that contains the entire spectrum.

The MDFT has been extended to non-Cartesian lattices. Mersereau provided a derivation of a DFT for a hexagonally periodic sequence and designed other digital filters suitable for a 2D hexagonal lattice [Mer79]. Later, the idea was extended to higher dimensions and a MDFT for arbitrary sampling lattices was proposed [MS83]. Guessoum and Mersereau proposed an algorithm for evaluating the MDFT that has the same computational complexity as the Cartesian DFT [GM86]. Our approach also yields an evaluation algorithm that has

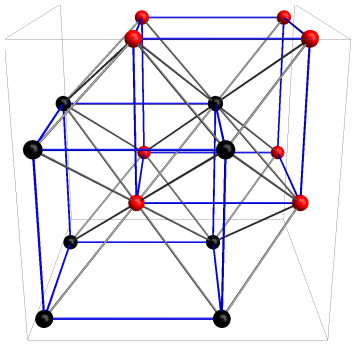


Figure 5.1: The BCC lattice, a 16 point view. Cosets are indicated by different colors.

the same complexity as the Cartesian DFT. We arrive at this complexity by taking advantage of the Cartesian coset structure that exists within the BCC lattice. This approach can also be extended to other lattices that have Cartesian cosets — a notable example being the FCC lattice.

5.2 Discrete Fourier Transform on BCC

Let $f \in \mathbb{R}^3$ be a periodic function that has a rectangularly shaped fundamental region and let $\hat{F}(\boldsymbol{\omega})$ be the DTFT (see (1.14) on page 8) of the finite sequence obtained by restricting the samples $f[\boldsymbol{n}] = f(\mathbf{L}\boldsymbol{n})$ to one complete period. The periodic sequence $f[\boldsymbol{n}]$ can then be represented in a Fourier basis, the coefficients of which are obtained by sampling $\hat{F}(\boldsymbol{\omega})$ in a rectangular fashion.

Recall that the (scaled) BCC lattice $\mathcal{H}$ is generated by the matrix $\mathbf{H}$ as given by (2.10) on page 34. It can also be built from shifts of a Cartesian sublattice as shown in Figure 5.1. In particular, samples that lie on the corners of cubes form the sublattice $2\mathbb{Z}^3$. The quotient group $\mathcal{H}/2\mathbb{Z}^3$ is isomorphic to $\mathbb{Z}_2$, the group of equivalence classes of the integers modulo 2. Therefore, $\mathcal{H}$ can be partitioned into two Cartesian cosets as shown in Figure 5.1.

The BCC lattice with arbitrary scaling is obtained via the sampling matrix $h\mathbf{H}$ where h is a positive scaling parameter. The Voronoi cell is a truncated octahedron having a volume of $\left|\det(h\mathbf{H})\right| = 4h^3$. The Voronoi cell of the dual FCC lattice is a rhombic dodecahedron having a volume of $\frac{1}{4h^3}$. Since $\mathcal{H}$ has two Cartesian cosets, a sampled sequence can be split up into two subsequences given by

$$f_0[\boldsymbol{n}] := f(2h\mathbf{I}\boldsymbol{n}) \quad \text{and} \quad f_1[\boldsymbol{n}] := f(2h\mathbf{I}\boldsymbol{n} + h\boldsymbol{\tau}), \tag{5.1}$$

where $\mathbf{I}$ is the 3×3 identity matrix, $\boldsymbol{\tau}$ is the translation vector $(1,1,1)^{\mathsf{T}}$ and $\boldsymbol{n}\in\mathbb{Z}^3$. $f_0[\boldsymbol{n}]$ is the sequence associated with the primary coset while $f_1[\boldsymbol{n}]$ is associated with the secondary coset.

5.2.1 Forward Transform

Let us assume that the samples $f_0[\boldsymbol{n}]$ and $f_1[\boldsymbol{n}]$ within one complete period of f can be indexed via the set

$$\mathcal{N} := \{\boldsymbol{n}\in\mathbb{Z}^3 : 0\le n_1<N_1, 0\le n_2<N_2, 0\le n_3<N_3\}, \tag{5.2}$$

for some positive integers N_1, N_2 and N_3. This region consists of $2N_1N_2N_3$ data points (i.e. Voronoi cells) and has a total volume of $8N_1N_2N_3h^3$. Let $\mathbf{N}$ be the diagonal matrix $\mathrm{diag}(N_1,N_2,N_3)$. Since f is periodic, the two subsequences $f_0[\boldsymbol{n}]$ and $f_1[\boldsymbol{n}]$ also exhibit Cartesian periodicity, i.e.

$$f_0[\boldsymbol{n}+\mathbf{N}\boldsymbol{r}] = f_0[\boldsymbol{n}] \quad\text{and}\quad f_1[\boldsymbol{n}+\mathbf{N}\boldsymbol{r}] = f_1[\boldsymbol{n}], \tag{5.3}$$

for all $\boldsymbol{n}$ and $\boldsymbol{r}$ in $\mathbb{Z}^3$.

This Cartesian periodicity in the spatial domain amounts to a Cartesian sampling in the Fourier domain. In particular, the DTFT $\hat{F}(\boldsymbol{\omega})$ is sampled at the frequencies $\boldsymbol{\omega}=\frac{1}{2h}\mathbf{N}^{-1}\boldsymbol{k}$ yielding the sequence

$$\begin{aligned}\hat{F}[\boldsymbol{k}] =& \hat{F}(\boldsymbol{\omega})\big|_{\boldsymbol{\omega}=\frac{1}{2h}\mathbf{N}^{-1}\boldsymbol{k}}\\ =& \sum_{\boldsymbol{n}\in\mathcal{N}} f_0[\boldsymbol{n}]\exp\big(\frac{-2\pi\imath}{2h}\boldsymbol{k}^{\mathsf{T}}\mathbf{N}^{-1}2h\mathbf{I}\boldsymbol{n}\big) + f_1[\boldsymbol{n}]\exp\big(\frac{-2\pi\imath}{2h}\boldsymbol{k}^{\mathsf{T}}\mathbf{N}^{-1}(2h\mathbf{I}\boldsymbol{n}+h\boldsymbol{\tau})\big)\\ =& \sum_{\boldsymbol{n}\in\mathcal{N}} \big(f_0[\boldsymbol{n}] + f_1[\boldsymbol{n}]\exp\big(-\pi\imath\boldsymbol{k}^{\mathsf{T}}\mathbf{N}^{-1}\boldsymbol{\tau}\big)\big)\cdot\exp\big(-2\pi\imath\boldsymbol{k}^{\mathsf{T}}\mathbf{N}^{-1}\boldsymbol{n}\big),\end{aligned} \tag{5.4}$$

where $\boldsymbol{k}\in\mathbb{Z}^3$ is the frequency index vector. The above equation defines a forward BCC DFT. Since it is a sampled version of the DTFT (1.14) that is periodic on the dual FCC lattice, it should be invariant under translations that lie on the dual lattice generated by the matrix $(h\mathbf{H})^{-\mathsf{T}}=\frac{1}{2h}\mathbf{G}$, where

$$\mathbf{G} := \begin{bmatrix}0&1&1\\1&0&1\\1&1&0\end{bmatrix}. \tag{5.5}$$

This property is easily demonstrated as follows. If $\boldsymbol{r} \in \mathbb{Z}^3$, then after substituting $\boldsymbol{\omega} = \frac{1}{2h}(\mathbf{N}^{-1}\boldsymbol{k} + \mathbf{G}\boldsymbol{r})$ in (5.4) and simplifying, we get

$$\begin{aligned}
&\hat{F}\big(\frac{1}{2h}(\mathbf{N}^{-1}\boldsymbol{k} + \mathbf{G}\boldsymbol{r})\big) \\
&= \sum_{\boldsymbol{n}\in\mathcal{N}} \big(f_0[\boldsymbol{n}] + f_1[\boldsymbol{n}] \exp\big(-\pi\imath(\boldsymbol{k}^\mathsf{T}\mathbf{N}^{-1} + \boldsymbol{r}^\mathsf{T}\mathbf{G})\boldsymbol{\tau}\big)\big) \cdot \exp\big(-2\pi\imath(\boldsymbol{k}^\mathsf{T}\mathbf{N}^{-1} + \boldsymbol{r}^\mathsf{T}\mathbf{G})\boldsymbol{n}\big) \\
&= \hat{F}[\boldsymbol{k}],
\end{aligned}$$

since $\boldsymbol{r}^\mathsf{T}\mathbf{G}\boldsymbol{\tau}$ is always even and $\boldsymbol{r}^\mathsf{T}\mathbf{G}\boldsymbol{n}$ is always an integer.

One fundamental period of the BCC DFT is contained within a rhombic dodecahedron of volume $\frac{1}{4h^3}$. The sampling density in the frequency domain is given by $\big|\det(\frac{1}{2h}\mathbf{N}^{-1})\big| = (8N_1N_2N_3h^3)^{-1}$. Thus, the fundamental period consists of a total of $2N_1N_2N_3$ distinct frequency samples which is the same as the number of distinct spatial samples.

5.2.2 Inverse Transform

The inverse BCC DFT is obtained by summing over all the distinct sinusoids and evaluating them at the spatial sample locations. This gives

$$f_0[\boldsymbol{n}] = \frac{1}{N} \sum_{\boldsymbol{k}\in\mathcal{K}} \hat{F}[\boldsymbol{k}] \exp\big(2\pi\imath\boldsymbol{k}^\mathsf{T}\mathbf{N}^{-1}\boldsymbol{n}\big) \tag{5.6a}$$

$$f_1[\boldsymbol{n}] = \frac{1}{N} \sum_{\boldsymbol{k}\in\mathcal{K}} \hat{F}[\boldsymbol{k}] \exp\big(2\pi\imath\boldsymbol{k}^\mathsf{T}\mathbf{N}^{-1}(\boldsymbol{n} + \tfrac{1}{2}\boldsymbol{\tau})\big) \tag{5.6b}$$

where $N = 2N_1N_2N_3$ is the number of samples and $\mathcal{K} \subset \mathbb{Z}^3$ is any set that indexes all the distinct frequency samples. It can be easily verified that both the sequences (5.6a) and (5.6b) are periodic with periodicity matrix $\mathbf{N}$.

5.2.3 Efficient Evaluation

Since $\mathbf{N}$ is diagonal, the summations in (5.4) and (5.6) are separable. This suggests that the transform can be efficiently computed via the rectangular multidimensional FFT, provided that a suitable rectangular index set $\mathcal{K}$ can be found. Observe that the Cartesian sequence $\hat{F}[\boldsymbol{k}]$ is also periodic with periodicity matrix $2\mathbf{N}$, i.e. $\hat{F}[\boldsymbol{k} + 2\mathbf{N}\boldsymbol{r}] = \hat{F}[\boldsymbol{k}]$ for all $\boldsymbol{r} \in \mathbb{Z}^3$. Therefore, one way to obtain a rectangular index set is to choose $\mathcal{K}$ such that it contains all the frequency indices within one period generated by the matrix $2\mathbf{N}$. This consists of a

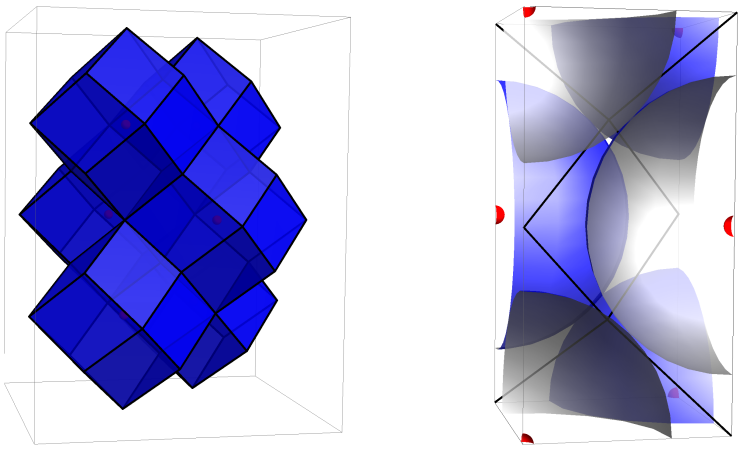


Figure 5.2: (Left) Six rhombic dodecahedra contribute to a non-redundant rectangular region. (Right) Zoomed in view of the non-redundant rectangular region that contains the full spectrum split into six pieces.

total of $\left|\det(2\mathbf{N})\right| = 4N$ indices and hence contains four replicas of the fundamental rhombic dodecahedron.

A non-redundant rectangular index set can be found by exploiting the geometric properties of the FCC lattice. If we consider the first octant only, $4N$ samples are contained within a cube formed by the FCC lattice sites that have even parity. This cube also contains six face-centered sites. By joining any two axially opposite face-centered sites, we can split the cube into four rectangular regions such that each region consists of non-redundant samples only. Six rhombic dodecahedra contribute to such a region as illustrated in Figure 5.2. The non-redundant region shown is obtained by limiting $\boldsymbol{k}$ to the index set given by

$$\mathcal{K} := \{\boldsymbol{k} \in \mathbb{Z}^3 : 0 \leq k_1 < N_1, 0 \leq k_2 < N_2, 0 \leq k_3 < 2N_3\}. \tag{5.7}$$

This region can further be subdivided into two cubes stacked on top of each other, each containing $N_1 \times N_2 \times N_3$ samples. The forward transform (5.4) can then be evaluated in the two cubes separately by appropriately applying the Cartesian FFT to the two sequences $f_0[\boldsymbol{n}]$ and $f_1[\boldsymbol{n}]$ and combining the results together. After rearranging terms in (5.4), the forward transform in the bottom cube becomes

$$\begin{aligned}\hat{F}_0[\boldsymbol{k}] := \hat{F}[\boldsymbol{k}] = &\sum_{\boldsymbol{n}\in\mathcal{N}} f_0[\boldsymbol{n}] \exp\bigl(-2\pi\imath \boldsymbol{k}^{\mathsf{T}}\mathbf{N}^{-1}\boldsymbol{n}\bigr) \\ &+ \exp(-\pi\imath \boldsymbol{k}^{\mathsf{T}}\mathbf{N}^{-1}\boldsymbol{\tau}) \sum_{\boldsymbol{n}\in\mathcal{N}} f_1[\boldsymbol{n}] \exp\bigl(-2\pi\imath \boldsymbol{k}^{\mathsf{T}}\mathbf{N}^{-1}\boldsymbol{n}\bigr),\end{aligned} \tag{5.8}$$

where $\boldsymbol{k}$ is now restricted to the set $\mathcal{N}$. Since this equation is valid for all $\boldsymbol{k} \in \mathbb{Z}^3$, the forward transform in the top cube can be computed from (5.8) by $\hat{F}_1[\boldsymbol{k}] := \hat{F}_0[\boldsymbol{k} + (0, 0, N_3)^{\mathsf{T}}]$ which simplifies to

$$\begin{aligned}\hat{F}_1[\boldsymbol{k}] = &\sum_{\boldsymbol{n}\in\mathcal{N}} f_0[\boldsymbol{n}] \exp\bigl(-2\pi\imath \boldsymbol{k}^{\mathsf{T}}\mathbf{N}^{-1}\boldsymbol{n}\bigr) \\ &- \exp(-\pi\imath \boldsymbol{k}^{\mathsf{T}}\mathbf{N}^{-1}\boldsymbol{\tau}) \sum_{\boldsymbol{n}\in\mathcal{N}} f_1[\boldsymbol{n}] \exp\bigl(-2\pi\imath \boldsymbol{k}^{\mathsf{T}}\mathbf{N}^{-1}\boldsymbol{n}\bigr),\end{aligned} \tag{5.9}$$

for $\boldsymbol{k} \in \mathcal{N}$. Both (5.8) and (5.9) are now in a form that permits a straightforward application of the Cartesian FFT. Since the two equations are structurally similar, only two $N_1 \times N_2 \times N_3$ FFT computations are needed, one for the sequence $f_1[\boldsymbol{n}]$ and one for $f_2[\boldsymbol{n}]$.

In a similar fashion, the inverse transform (5.6) can be computed using two inverse FFT computations. Splitting the summations in (5.6) into the two constituent cubes gives

$$\begin{aligned}f_0[\boldsymbol{n}] &= \frac{1}{N} \sum_{\boldsymbol{k}\in\mathcal{N}} \bigl(\hat{F}_0[\boldsymbol{k}] + \hat{F}_1[\boldsymbol{k}]\bigr) \exp\bigl(2\pi\imath \boldsymbol{k}^{\mathsf{T}}\mathbf{N}^{-1}\boldsymbol{n}\bigr), \quad \text{and} \\ f_1[\boldsymbol{n}] &= \frac{1}{N} \sum_{\boldsymbol{k}\in\mathcal{N}} \bigl((\hat{F}_0[\boldsymbol{k}] - \hat{F}_1[\boldsymbol{k}]) \exp(\pi\imath \boldsymbol{k}^{\mathsf{T}}\mathbf{N}^{-1}\boldsymbol{\tau})\bigr) \cdot \exp\bigl(2\pi\imath \boldsymbol{k}^{\mathsf{T}}\mathbf{N}^{-1}\boldsymbol{n}\bigr).\end{aligned} \tag{5.10}$$

5.3 Discrete Sine Transform on BCC

5.3.1 Forward Transform

If the Cartesian periodic function f is odd with respect to each variable, the MDFT turns into a MDST and it suffices to consider only the samples $f_0[\boldsymbol{m}]$ and $f_1[\boldsymbol{m}]$ that are within the positive octant of the fundamental period (see (4.18) on page 71). The forward transform becomes

$$\tilde{F}[\boldsymbol{k}] = \sum_{\boldsymbol{m}\in\mathcal{M}} f_0[\boldsymbol{m}]\bigl(\textstyle\prod_{i=1}^{3} \sin(\frac{\pi}{M_i} k_i m_i)\bigr) + \sum_{\boldsymbol{m}\in\mathcal{M}} f_1[\boldsymbol{m}]\bigl(\textstyle\prod_{i=1}^{3} \sin(\frac{\pi}{M_i} k_i (m_i + \frac{1}{2}))\bigr), \tag{5.11}$$

where $\boldsymbol{k} \in \mathbb{Z}^3_+$ and the index set $\mathcal{M}$ is defined as follows:

$$\mathcal{M} := \{\boldsymbol{m} \in \mathbb{Z}^3 : 0 \le m_1 < M_1, 0 \le m_2 < M_2, 0 \le m_3 < M_3\}. \tag{5.12}$$

Like the BCC DFT, the key to efficiently evaluating this transform is to find a suitable rectangular region in the Fourier domain that contains all the positive frequencies. Recall that the spectrum is replicated on the FCC lattice which means that the positive frequencies

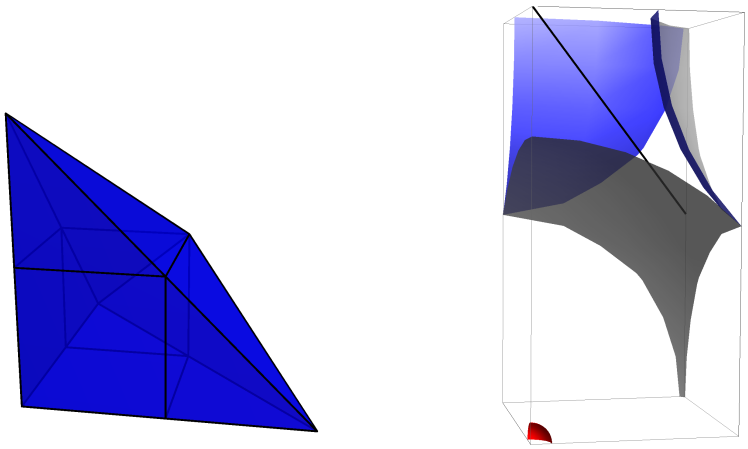


Figure 5.3: (Left) Part of the rhombic dodecahedron that lies in the first octant and contains positive frequencies. It consists of a cube with three pyramids attached to it. (Right) Rectangular region obtained from the rhombic dodecahedral tessellation that contains the positive part of the spectrum split into three pieces.

are contained within that portion of the rhombic dodecahedron that lies in the positive octant (Figure 5.3). Exploiting the geometry of the rhombic dodecahedral tessellation, all the positive frequencies are also contained within a rectangular region obtained by limiting $\boldsymbol{k}$ to the index set

$$\mathcal{K}_s := \{\boldsymbol{k} \in \mathbb{Z}_+^3 : k_1 \leq M_1, k_2 \leq M_2, k_3 \leq 2M_3\}, \tag{5.13}$$

as shown in Figure 5.3. As before, we split this region into two cubes each containing $M_1 \times M_2 \times M_3$ samples. The transform in the bottom cube is given by

$$\tilde{F}_0[\boldsymbol{k}] := \tilde{F}_a[\boldsymbol{k}] + \tilde{F}_b[\boldsymbol{k}], \tag{5.14}$$

where

$$\tilde{F}_a[\boldsymbol{k}] := \sum_{\boldsymbol{m} \in \mathcal{M}} f_0[\boldsymbol{m}] \big(\textstyle\prod_{i=1}^3 \sin(\frac{\pi}{M_i} k_i m_i) \big), \qquad \text{(type-I)} \tag{5.15}$$

$$\tilde{F}_b[\boldsymbol{k}] := \sum_{\boldsymbol{m} \in \mathcal{M}} f_1[\boldsymbol{m}] \big(\textstyle\prod_{i=1}^3 \sin(\frac{\pi}{M_i} k_i (m_i + \frac{1}{2})) \big), \qquad \text{(type-II)} \tag{5.16}$$

and $\boldsymbol{k}$ is restricted to

$$\mathcal{M}_+ := \{\boldsymbol{m} \in \mathbb{Z}_+^3 : m_1 \leq M_1, m_2 \leq M_2, m_3 \leq M_3\}. \tag{5.17}$$

As indicated in (5.15) and (5.16), $\tilde{F}_0[\boldsymbol{k}]$ can be efficiently computed by adding the type-I DST of the primary sequence $f_0[\boldsymbol{m}]$ and the type-II DST of the secondary sequence $f_1[\boldsymbol{m}]$[1]. Analogous to the BCC DFT, the transform in the top cube is given by $\tilde{F}_1[\boldsymbol{k}] := \tilde{F}_0[\boldsymbol{k} + (0, 0, M_3)^{\mathsf{T}}]$ for $\boldsymbol{k} \in \mathcal{M}_+$. After simplifying, we observe that $\tilde{F}_1[\boldsymbol{k}]$ can be obtained by simply reversing the sequences $\tilde{F}_a$ and $\tilde{F}_b$. In particular, we have

$$\tilde{F}_1[k_1, k_2, k_3] = -\tilde{F}_a[k_1, k_2, M_3 - k_3] + \tilde{F}_b[k_1, k_2, M_3 - k_3], \qquad (\boldsymbol{k} \in \mathcal{M}_+) \tag{5.18}$$

with the boundary conditions $\tilde{F}_a[k_1, k_2, 0] = 0$ and $\tilde{F}_b[k_1, k_2, 0] = 0$. Thus, only two DST computations are needed: a type-I DST of $f_0[\cdot]$ and a type-II DST of $f_1[\cdot]$.

At this point, we would like to point out that the forward transform thus obtained has some redundancies which we exploit for the purpose of efficiently computing the inverse transform. Assuming that $\boldsymbol{k} \in \mathcal{M}_+$, the following can be easily verified by inspecting (5.11):

$$\tilde{F}[k_1, k_2, 2M_3] = 0 \quad \text{and} \quad \tilde{F}[M_1, M_2, k_3] = \tilde{F}[M_1, M_2, 2M_3 - k_3]. \tag{5.19}$$

5.3.2 Inverse Transform

The inverse transform is obtained by summing up the sinusoids and evaluating them at the spatial locations of the BCC lattice. Thus, we have

$$\begin{aligned} f_0[\boldsymbol{m}] &= \frac{4}{M} \sum_{\boldsymbol{k} \in \mathcal{K}_s} \tilde{F}[\boldsymbol{k}] \big(\textstyle\prod_{i=1}^{3} \sin(\frac{\pi}{M_i} k_i m_i) \big) \quad \text{and} \\ f_1[\boldsymbol{m}] &= \frac{4}{M} \sum_{\boldsymbol{k} \in \mathcal{K}_s} w[\boldsymbol{k}] \tilde{F}[\boldsymbol{k}] \big(\textstyle\prod_{i=1}^{3} \sin(\frac{\pi}{M_i} k_i (m_i + \frac{1}{2})) \big), \end{aligned} \tag{5.20}$$

where $M = M_1 M_2 M_3$, $\boldsymbol{m} \in \mathcal{M}$, and the weighting sequence $w[\boldsymbol{k}]$ ($\boldsymbol{k} \in \mathcal{K}_s$) handles the redundancy (5.19) and is defined as follows:

$$w[k_1, k_2, k_3] := \begin{cases} \frac{1}{2} & \text{if } k_1 = M_1, k_2 = M_2 \text{ and } k_3 \neq M_3, \\ 1 & \text{otherwise.} \end{cases} \tag{5.21}$$

As before, in order to efficiently evaluate the inverse transform, we distribute the evaluation over the constituent sequences $\tilde{F}_0[\boldsymbol{k}]$ and $\tilde{F}_1[\boldsymbol{k}]$ ($\boldsymbol{k} \in \mathcal{M}_+$). The primary sequence $f_0[\cdot]$ is given by

$$f_0[\boldsymbol{m}] = \frac{4}{M} \sum_{\boldsymbol{k} \in \mathcal{M}_+} (\tilde{F}_0[\boldsymbol{k}] - \tilde{F}_1[k_1, k_2, M_3 - k_3]) \big(\textstyle\prod_{i=1}^{3} \sin(\frac{\pi}{M_i} k_i m_i) \big), \quad \text{(type-I)} \tag{5.22}$$

[1]For the definitions of the various types of the DST, please see the work of Martucci [Mar94].

where $\boldsymbol{m} \in \mathcal{M}$ and we use the boundary condition $\tilde{F}_1[k_1, k_2, 0] = 0$. Likewise, the secondary sequence $f_1[\cdot]$ is given by

$$f_1[\boldsymbol{m}] = \frac{4}{M} \sum_{\boldsymbol{k} \in \mathcal{M}_+} w[\boldsymbol{k}](\tilde{F}_0[\boldsymbol{k}] + \tilde{F}_1[k_1, k_2, M_3 - k_3])\big(\textstyle\prod_{i=1}^{3} \sin(\frac{\pi}{M_i} k_i(m_i + \frac{1}{2}))\big), \quad \text{(type-III)} \tag{5.23}$$

where $\boldsymbol{m} \in \mathcal{M}$ and we use the boundary condition $\tilde{F}_1[k_1, k_2, 0] = \tilde{F}_0[k_1, k_2, M_3]$. As indicated, (5.22) and (5.23) are now in a form that permits the straightforward use of a type-I and type-III DST respectively.

5.4 Notes

The material covered in Section 5.2 also appears in our work [AM09], where we also investigate the efficient non-redundant evaluation of the MDFT on the FCC lattice. A redundant version of the BCC DFT has also been proposed by Csébfalvi and Domonkos [CD08].

Chapter 6

Conclusions

We have seen that the high fidelity promised by the BCC lattice extends beyond scalar representations. By casting the problem of approximating derived quantities into the framework of shift-invariant spaces, we have proposed techniques that inherit the approximation properties of a generating function, thus yielding high quality shift-invariant representations for the derived quantities. Our processing techniques are entirely convolution based and therefore, can be readily implemented in a discrete computational setting. Even though we have focused on the three-dimensional CC and BCC lattices in our experiments, our formulation, for the most part, has remained general. Thus, our processing techniques can be easily extended to other lattices in two and higher dimensions.

6.1 Application Areas

At the outset, we mentioned our brief stint in data acquisition on the BCC lattice [AEM09, FAVM09]. Based on the results we obtained there as well as the results reported in this thesis, we are confident that there are many application areas that can be improved simply by changing the sampling pattern from Cartesian to body-centric. As more and more data is acquired on the BCC lattice, the processing tools outlined here, as well as their extensions described below, will become more and more relevant.

In the context of gradient estimation, we have ignored the fact that the gradient of a scalar function is curl-free. Incorporating this requirement into the shift-invariant formulation poses additional challenges that are worthy of further investigation. On that note, a

related problem is the accurate reconstruction of divergence-free velocity fields for the purpose of flow visualization [HJ05, Part IV]. Particle tracing techniques such as line-integral convolution [CL93] and stream surfaces [vW93] usually interpolate the vector field in a component-wise fashion and therefore, do not respect the divergence-free criterion. Furthermore, these techniques usually assume a Cartesian sampling of the velocity field. A careful treatment that identifies grid-based artifacts and proposes divergence preserving measures is therefore warranted. This problem is also interesting from the perspective of creating realistic fluid effects in computer graphics where, inviscid flows are usually simulated using a semi-Lagrangian advection scheme [Sta99, FSJ01]. Divergence preserving heuristics lead to reduced numerical viscosity and mass loss [LAF11].

The Poisson solver described in Chapter 4 also has some immediate applications in computer graphics. The method of projection for solving the incompressible Navier-Stokes equations yields a Poisson equation [Sta99]. If free-flow boundary conditions are imposed, it can be accurately solved using our shift-invariant formulation. The Poisson equation also shows up in gradient domain image-processing [PGB03], as well as surface reconstruction from range-scan data [KBH06].

The field of diffusion tensor imaging (DTI) is gradually shifting towards the high-angular-resolution diffusion imaging (HARDI) model [TRW$^+$02] which is capable of discerning crossing neuronal fibres. HARDI tensors are usually reconstructed on a CC lattice but the proper interpolation of these tensors at non-grid locations is a problem that has received little attention. Furthermore, the use of non-Cartesian lattices in DTI is completely uncharted territory. The encouraging scalar and gradient reconstruction results we have obtained on the BCC lattice lead us to believe that the field of DTI can also significantly benefit by adopting a non-Cartesian model.

6.2 Future Directions

Our focus has largely been on the development of discrete filters that attempt to fully harness the approximation capabilities of a given generating function. However, the error kernel formulations (see (3.4) and (4.25)) are more flexible and allow for the possibility of optimizing the generating function to better approximate the action of a shift-invariant operator. For instance, we saw in Chapter 3 that a symmetric generator, when shifted in the direction of the derivative, yields better derivative approximations. However, we shifted

the generator by the same amount for all the directions. The possibility of finding optimal shifts for a given symmetric generator is open for future scrutiny.

Despite the data reduction promised by the BCC lattice, the CC lattice remains a popular choice. One reason for this divide is the fact that most practical visualization applications follow Shneiderman's visual information-seeking mantra: *"Overview first, zoom and filter, details on demand"* [Shn96]. In volume visualization, the CC lattice easily lends itself to this mantra since efficient multiscale wavelet representations are readily available. On the other hand, wavelets on non-Cartesian lattices such as the BCC lattice are relatively less explored. Such representations are desirable owing to the reduced support size of the wavelet bases [HJ02]. This can potentially lead to more efficient, yet accurate, multiscale representations suitable for interactive visualization applications.

Compressed sensing — the notion of taking advantage of the sparsity of a signal in a transformed domain [CW08] — is gaining quite a bit of attention in many imaging areas. The optimal sampling paradigm that we have used in this thesis is based on a worst-case scenario in which the entire isotropic bandwidth is occupied. An interesting idea, albeit in the univariate setting, is to take advantage of a sparse shift-invariant representation of the signal [Eld09]. The extension of this idea to higher dimensions and to non-Cartesian lattices are interesting research directions.

Towards the end of Chapter 1, we saw that the fourth-order quintic box spline on BCC, despite having a support that is half the size of the tricubic B-spline on CC, achieves a lower asymptotic constant (see Figure 1.3 on page 23). A natural question therefore arises: What is the minimum support size for a given order k? On a related note – for the same support size, which lattice yields the maximum approximation order? In the univariate setting, the answer to both of these questions is provided by the MOMS framework of Blu *et al.* [BTU01]. In higher dimensions, we are unaware of any work that attempts to address these questions. It is conjectured that the minimum support size is the same as the dimensionality of the polynomial space Π_{k-1}. However, proving this conjecture and finding generators that realize the minimum support are interesting and challenging open issues.

Bibliography

[AEM09] U. R. Alim, A. Entezari, and T. Möller. The Lattice-Boltzmann method on optimal sampling lattices. *IEEE Transactions on Visualization and Computer Graphics* 15 (4):630–641, 2009. ↑viii, 99

[AM09] U. R. Alim and T. Möller. A fast Fourier transform with rectangular output on the BCC and FCC lattices. In Proceedings of the Eighth International Conference on Sampling Theory and Applications (SampTA'09). May 2009. ↑98

[AMC10] U. R. Alim, T. Möller, and L. Condat. Gradient estimation revitalized. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2010)* 16 (6):1494–1503, November 2010. ↑64

[AU94] A. Aldroubi and M. Unser. Sampling procedures in function spaces and asymptotic equivalence with Shannon's sampling theory. *Numerical Functional Analysis and Optimization* 15 (1-2):1–21, 1994. ↑8, 13

[BDHDH05] R. C. M. Brekelmans, L. T. Driessen, H. J. M. Hamers, and D. Den Hertog. Gradient estimation schemes for noisy functions. *Journal of Optimization Theory and Applications* 126 (3):529–551, 2005. ↑29

[BDHDH08] ———. Gradient estimation using Lagrange interpolation polynomials. *Journal of Optimization Theory and Applications* 136 (3):341–357, 2008. ↑29, 43

[BML96] M. J. Bentum, T. Malzbender, and B. B. Lichtenbelt. Frequency analysis of gradient estimators in volume rendering. *IEEE Transactions on Visualization and Computer Graphics* 2 (3):242–254, September 1996. ↑30

[Bri08] R. Bridson. *Fluid Simulation For Computer Graphics*. Ak Peters Series. A. K. Peters, 2008. ↑64

[BTU01] T. Blu, P. Thévenaz, and M. Unser. MOMS: Maximal-order interpolation of minimal support. *IEEE Transactions on Image Processing* 10 (7):1069–1080, 2001. ↑15, 19, 86, 101

[BTU04] T. Blu, P. Thévenaz, and M. Unser. Linear interpolation revitalized. *IEEE Transactions on Image Processing* 13 (5):710–719, May 2004. ↑15, 64

[BU99a] T. Blu and M. Unser. Quantitative Fourier analysis of approximation techniques: Part I—Interpolators and projectors. *IEEE Transactions on Signal Processing* 47 (10):2783–2795, October 1999. ↑14, 15, 25, 32, 47, 50, 51, 72

[BU99b] T. Blu and M. Unser. Approximation error for quasi-interpolators and (multi-)wavelet expansions. *Applied and Computational Harmonic Analysis* 6 (2):219–251, 1999. ↑13, 14

[CBU05] L. Condat, T. Blu, and M. Unser. Beyond interpolation: Optimal reconstruction by quasi-interpolation. In Proceedings of the 2005 IEEE International Conference on Image Processing (ICIP'05), pages 33–36. September 2005. ↑15

[CD08] B. Csébfalvi and B. Domonkos. Pass-band optimal reconstruction on the body-centered cubic lattice. In Proceedings of Vison, Modeling, and Visualization 2008, pages 71–80. October 2008. ↑98

[CD09] B. Csébfalvi and B. Domonkos. Prefiltered gradient reconstruction for volume rendering. *Journal of Winter School of Computer Graphics* 17 (1-3):49–56, 2009. ↑44, 47

[CD90] C. K. Chui and H. Diamond. A characterization of multivariate quasi-interpolation formulas and its applications. *Numerische Mathematik* 121:105–121, 1990. ↑13

[Chu88] C. K. Chui. *Multivariate Splines*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1988. ↑18

[CL93] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '93), pages 263–270. August 1993. ↑100

[CM11] L. Condat and T. Möller. Quantitative error analysis for the reconstruction of derivatives. *IEEE Transactions on Signal Processing* 59 (6):2965 –2969, June 2011. ↑15, 47, 50, 51, 88

[CMS06] H. Carr, T. Möller, and J. Snoeyink. Artifacts caused by simplicial subdivision. *IEEE Transactions on Visualization and Computer Graphics* 12 (2):231–242, 2006. ↑88

[Col60] L. Collatz. *The Numerical Treatment of Differential Equations*. Springer-Verlag, 1960. ↑86

[Csé10] B. Csébfalvi. An evaluation of prefiltered B-spline reconstruction for quasi-interpolation on the body-centered cubic lattice. *IEEE Transactions on Visualization and Computer Graphics* 16 (3):499–512, May 2010. ↑18, 24

[CS99] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups*. Springer, 3rd ed., 1999. ↑3, 16

[Csé08a] B. Csébfalvi. BCC-splines: Generalization of B-splines for the body-centered cubic lattice. *Journal of Winter School of Computer Graphics* 16 (1-3):81–88, 2008. ↑18

[Csé08b] ———. An evaluation of prefiltered reconstruction schemes for volume rendering. *IEEE Transactions on Visualization and Computer Graphics* 14 (2):289–301, 2008. ↑29, 47

[CVDV07] L. Condat and D. Van De Ville. Quasi-interpolating spline models for hexagonally-sampled data. *IEEE Transactions on Image Processing* 16 (5):1195 –1206, May 2007. ↑18, 54

[CW08] E. J. Candes and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine* 25 (2):21–30, March 2008. ↑101

[DC10] B. Domonkos and B. Csébfalvi. DC-splines: Revisiting the trilinear interpolation on the body-centered cubic lattice. In Proceedings of Vision, Modeling, and Visualization 2010, pages 275–282. November 2010. ↑18

[DC11] ———. 3D frequency domain analysis of reconstruction schemes on the body-centered cubic lattice. *Computer, Graphics and Geometry* 13 (1):31–50, 2011. ↑18

[dDR94] C. de Boor, R. A. Devore, and A. Ron. Approximation from shift-invariant subspaces of $L_2(\mathbb{R}^d)$. *Transactions of the American Mathematical Society* 341 (2):787–806, 1994. ↑11

[deB01] C. de Boor. *A Practical Guide to Splines*, Vol. 27. Springer Verlag, 2001. ↑7

[deB87] ———. The polynomials in the linear span of integer translates of a compactly supported function. *Constructive Approximation* 3 (1):199–208, December 1987. ↑26

[deB93] ———. On the evaluation of box splines. *Numerical Algorithms* 5 (1–4):5–23, 1993. ↑23

[dHR93] C. de Boor, K. Höllig, and S. Riemenschneider. *Box Splines*. Springer Verlag, 1993. ↑7, 18, 19, 35, 49

[DK93] S. C. Dutta Roy and B. Kumar. *Handbook of Statistics*, Vol. 10. Elsevier Science Publishers B. V., 1993. ↑30, 38

[DM84] D. E. Dudgeon and R. M. Mersereau. *Multidimensional Digital Signal Processing*. Prentice-Hall, Inc., Englewood-Cliffs, NJ, 1st ed., 1984. ↑3, 8

[EDM04] A. Entezari, R. Dyer, and T. Möller. Linear and cubic box splines for the body centered cubic lattice. In Proceedings of the IEEE Conference on Visualization, pages 11–18. October 2004. ↑18, 20, 24, 37, 44, 47

[Eld09] Y. C. Eldar. Compressed sensing of analog signals in shift-invariant spaces. *IEEE Transactions on Signal Processing* 57 (8):2986 –2997, August 2009. ↑101

[EMK09] A. Entezari, M. Mirzargar, and L. Kalantari. Quasi-interpolation on the body centered cubic lattice. *Computer Graphics Forum* 28 (3):1015 –1022, 2009. ↑18

[Ent07] A. Entezari, *Optimal Sampling Lattices and Trivariate Box Splines*. Ph.D. Thesis, Burnaby BC, Canada, 2007. ↑17

[EVM08] A. Entezari, D. Van De Ville, and T. Möller. Practical box splines for reconstruction on the body centered cubic lattice. *IEEE Transactions on Visualization and Computer Graphics* 14 (2):313 –328, 2008. ↑18, 20, 24, 33, 35, 59

[FAVM09] B. Finkbeiner, U. R. Alim, D. Van De Ville, and T. Möller. High-quality volumetric reconstruction on optimal lattices for computed tomography. *Computer Graphics Forum (Proceedings of the Eurographics/IEEE-VGTC Symposium on Visualization 2009 (EuroVis 2009))* 28 (3):1023–1030, 2009. ↑viii, 29, 47, 99

[FEVM10] B. Finkbeiner, A. Entezari, D. Van De Ville, and T. Möller. Efficient volume rendering on the body centered cubic lattice using box splines. *Computers & Graphics* 34 (4):409–423, 2010. ↑18, 24

[FJ05] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE* 93 (2):216–231, 2005. ↑38, 81, 90

[FS04] H. Farid and E. P. Simoncelli. Differentiation of discrete multi-dimensional signals. *IEEE Transactions on Image Processing* 13 (4):496–508, 2004. ↑30

[FSJ01] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH '01), pages 15–22. August 2001. ↑64, 100

[GM86] A. Guessoum and R. Mersereau. Fast algorithms for the multidimensional discrete Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing* ASSP-34 (4):937–943, 1986. ↑90

[Gos94] M. E. Goss. An adjustable gradient filter for volume visualization image enhancement. In Proceedings of Graphics Interface '94, pages 67–74. May 1994. ↑30

[HAM11] Z. Hossain, U. R. Alim, and T. Möller. Toward high quality gradient estimation on regular lattices. *IEEE Transactions on Visualization and Computer Graphics* 17 (4):426–439, April 2011. ↑38, 40, 42, 43, 44, 57, 59

[HJ02] B. Han and R. Q. Jia. Quincunx fundamental refinable functions and quincunx biorthogonal wavelets. *Mathematics of Computation* 71 (237):165–196, 2002. ↑101

[HJ05] C. D. Hansen and C. R. Johnson. *The Visualization Handbook*. Referex Engineering. Butterworth-Heinemann, 2005. ↑38, 100

[HW65] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids* 8 (12):2182, 1965. ↑64

[JBU02] M. Jacob, T. Blu, and M. Unser. Sampling of periodic signals: A quantitative error analysis. *IEEE Transactions on Signal Processing* 50 (5):1153–1159, 2002. ↑15, 69, 70, 71, 72

[Jia95] R. Q. Jia. Refinable shift-invariant spaces: from splines to wavelets. In Approximation Theory VIII, Vol. 2, pages 403–427. 1995. ↑6

[Jia98] ———. Approximation properties of multivariate wavelets. *Mathematics of Computation* 67 (222):647–666, 1998. ↑7, 26

[JP01] K. Jetter and G. Plonka. A survey on L_2-approximation order from shift-invariant spaces. In Multivariate Approximation and Applications, pages 73–111. 2001. ↑6, 9, 10, 11

[KBH06] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In Proceedings of the fourth Eurographics symposium on Geometry processing, pages 61–70. June 2006. ↑100

[KD98] G. Kindlmann and J. W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In IEEE Symposium on Volume Visualization, pages 79–86. October 1998. ↑28

[KEP08] M. Kim, A. Entezari, and J. Peters. Box spline reconstruction on the face-centered cubic lattice. *IEEE Transactions on Visualization and Computer Graphics* 14 (6):1523–1530, 2008. ↑18

[Key81] R. G. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transaction on Acoustics, Speech, and Signal Processing* 29 (6):1153–1160, December 1981. ↑10, 86

[KP10] M. Kim and J. Peters. Symmetric box-splines on the $\mathcal{A}_n^*$ lattice. *Journal of Approximation Theory* 162 (9):1607–1630, 2010. ↑18

[KP11] ———. Symmetric box-splines on root lattices. *Journal of Computational and Applied Mathematics* 235 (14):3972–3989, 2011. ↑18

[Kre89] E. Kreyszig. *Introductory Functional Analysis with Applications*. Wiley Classics Library. Wiley, 1989. ↑4, 9

[KTW06] G. Kindlmann, X. Tricoche, and C.-F. Westin. Anisotropy creases delineate white matter structure in diffusion tensor MRI. In Ninth International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI '06), pages 126–133. October 2006. ↑28

[KWTM03] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Möller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In Proceedings of the IEEE Conference on Visualization, pages 513–520. October 2003. ↑28

[LAF11] M. Lentine, M. Aanjaneya, and R. Fedkiw. Mass and momentum conservation for fluid simulation. In Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pages 91–100. August 2011. ↑100

[LDL09] Y. M. Lu, M. N. Do, and R. S. Laugesen. A computable Fourier condition generating alias-free sampling lattices. *IEEE Transactions on Signal Processing* 57 (5):1768–1782, 2009. ↑3

[Mar94] S. A. Martucci. Symmetric convolution and the discrete sine and cosine transforms. *IEEE Transactions on Signal Processing* 42 (5):1038–1051, 1994. ↑81, 97

[Mar99] S. L. Marshall. A rapidly convergent modified Green's function for Laplace's equation in a rectangular region. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 455 (1985):1739–1766, 1999. ↑67, 88, 89

[ME10] M. Mirzargar and A. Entezari. Voronoi splines. *IEEE Transactions on Signal Processing* 58 (9):4572–4582, 2010. ↑18, 19, 21

[ME11] M. Mirzargar and A. Entezari. Quasi interpolation with Voronoi splines. *IEEE Transactions on Visualization and Computer Graphics* 17 (12):1832–1841, 2011. ↑18

[Mer79] R. M. Mersereau. The processing of hexagonally sampled two-dimensional signals. *Proceedings of the IEEE* 67 (6):930–949, June 1979. ↑90

[MES+11] T. Meng, A. Entezari, B. Smith, T. Möller, D. Weiskopf, and A. E. Kirkpatrick. Visual comparability of 3D regular sampling and reconstruction. *IEEE Transactions on Visualization and Computer Graphics* 17 (10):1420–1432, 2011. ↑18

[Miy59] H. Miyakawa. Sampling theorem of stationary stochastic variables in multidimensional space. *Journal of the Institute of Electronic and Communication Engineers of Japan* 42:421–427, 1959. ↑2

[Miz73] S. Mizohata. *The Theory of Partial Differential Equations*. University Press, 1973. ↑22

[ML94] S. R. Marschner and R. J. Lobb. An evaluation of reconstruction filters for volume rendering. In Proceedings of the IEEE Conference on Visualization, pages 100–107. October 1994. ↑10, 26, 39, 40, 59

[MMK+98] T. Möller, K. Mueller, Y. Kurzion, R. Machiraju, and R. Yagel. Design of accurate and smooth filters for function and derivative reconstruction. In Proceedings of the Symposium on Volume Visualization, pages 143–151. October 1998. ↑28, 29

[MMMY97a] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. A comparison of normal estimation schemes. In Proceedings of the IEEE Conference on Visualization, pages 19–26. October 1997. ↑46

[MMMY97b] T. Möller, R. Machiraju, K. Müller, and R. Yagel. Evaluation and design of filters using a Taylor series expansion. *IEEE Transactions on Visualization and Computer Graphics* 3 (2):184–199, April 1997. ↑6, 29

[MN88] D. P. Mitchell and A. N. Netravali. Reconstruction filters in computer graphics. In Computer Graphics (Proceedings of SIGGRAPH '88), pages 221–228. August 1988. ↑10, 26

[MS83] R. Mersereau and T. Speake. The processing of periodically sampled multidimensional signals. *IEEE Transactions on Acoustics, Speech, and Signal Processing* ASSP-31 (1):188–194, 1983. ↑90

[MSE+07] T. Meng, B. Smith, A. Entezari, A. E. Kirkpatrick, D. Weiskopf, L. Kalantari, and T. Möller. On visual quality of optimal 3D sampling and reconstruction. In Graphics Interface 2007, pages 265 –272. May 2007. ↑37, 47

[MU03] E. Meijering and M. Unser. A note on cubic convolution interpolation. *IEEE Transactions on Image Processing* 12 (4):477–479, 2003. ↑86

[Nie92] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial Mathematics, 1992. ↑68

[NM02] N. Neophytou and K. Mueller. Space-time points: 4D splatting on efficient grids. In VVS '02: Proceedings of the 2002 IEEE symposium on Volume visualization, pages 97–106. October 2002. ↑29, 47

[PB11] J. D. Patterson and B. Bailey. *Solid-State Physics: Introduction to the Theory*. Springer, 2011. ↑16, 17

[PGB03] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on Graphics* 22 (3):313–318, July 2003. ↑100

[Pin02] M. A. Pinsky. *Introduction to Fourier Analysis and Wavelets*, Vol. 102. American Mathematical Society, 2002. ↑2, 10

[PM62] D. P. Petersen and D. Middleton. Sampling and reconstruction of wave-number-limited functions in N-dimensional Euclidean spaces. *Information and Control* 5 (4):279–323, December 1962. ↑2, 3

[QV08] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*. Springer Series in Computational Mathematics. Springer, 2008. ↑65, 75

[Sch46] I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. *Quarterly of Applied Mathematics* 4 (2):45–99, 1946. ↑5

[SF71] G. Strang and G. J. Fix. A Fourier analysis of the finite element variational method. *Constructive Aspects of Functional Analysis*:796–830, 1971. ↑6

[Sha49] C. E. Shannon. Communication in the presence of noise. *Proceedings of the Institute of Radio Engineers* 37 (1):10–21, 1949. ↑2

[Shn96] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In Proceedings of the 1996 IEEE Symposium on Visual Languages, pages 336–343. 1996. ↑101

[SKZC03] H. Sun, N. Kang, J. Zhang, and E. S. Carlson. A fourth-order compact difference scheme on face centered cubic grids with multigrid method for solving 2D convection diffusion equation. *Mathematics and Computers in Simulation* 63 (6):651–661, 2003. ↑29

[Sta99] J. Stam. Stable fluids. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99), pages 121–128. August 1999. ↑100

[Str04] J. C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Society for Industrial Mathematics, 2004. ↑65

[TMG01] T. Theußl, T. Möller, and E. Gröller. Optimal regular volume sampling. In Proceedings of the IEEE Conference on Visualization 2001, pages 91–98. October 2001. ↑7, 29, 37, 47, 59

[TRW+02] D. S. Tuch, T. G. Reese, M. R. Wiegell, N. Makris, J. W. Belliveau, and V. J. Wedeen. High angular resolution diffusion imaging reveals intravoxel white matter fiber heterogeneity. *Magnetic Resonance in Medicine* 48 (4):577–582, 2002. ↑100

[UAE93] M. Unser, A. Aldroubi, and M. Eden. B-Spline signal processing: Part I—Theory. *IEEE Transactions on Signal Processing* 41 (2):821–833, February 1993. ↑17, 19, 33, 34

[UD97] M. Unser and I. Daubechies. On the approximation power of convolution-based least squares versus interpolation. *IEEE Transaction on Signal Processing* 45 (7):1697–1711, 1997. ↑13

[Uns00] M. Unser. Sampling-50 years after Shannon. *Proceedings of the IEEE* 88 (4):569–587, 2000. ↑6, 9, 12

[Uns95] ———. A general Hilbert space framework for the discretization of continuous signal processing operators. In Proceedings of the SPIE Conference on Mathematical Imaging: Wavelet Applications in Signal and Image Processing III, pages 51–61. July 1995. ↑31

[Uns96] ———. Approximation power of biorthogonal wavelet expansions. *IEEE Transactions on Signal Processing* 44 (3):519–527, 1996. ↑9

[VBU+04] D. Van De Ville, T. Blu, M. Unser, W. Philips, I. Lemahieu, and R. Van de Walle. Hex-splines: A novel spline family for hexagonal lattices. *IEEE Transactions on Image Processing* 13 (6):758–772, June 2004. ↑18

[vW93] J. J. van Wijk. Flow visualization with surface particles. *IEEE Computer Graphics and Applications* 13 (4):18–24, 1993. ↑100

[Wan01] R. Wang. *Multivariate Spline Functions and their Applications*. Mathematics and its Applications. Kluwer Academic Publishers, 2001. ↑18

[Wik12a] Wikipedia. *Rhombic dodecahedron — Wikipedia, the free encyclopedia*, 2012. [Online; accessed 06-April-2012]. ↑17

[Wik12b] ———. *Truncated octahedron — Wikipedia, the free encyclopedia*, 2012. [Online; accessed 06-April-2012]. ↑17

[Wol10] Wolfram Research Inc. *Mathematica, Version 8.0*, Champaign, IL, 2010. ↑23

[You92] E. C. Young. *Vector and Tensor Analysis*. Marcel Dekker Inc., 1992. ↑56

Appendix A

BCC Quasi-interpolation Poisson Filters

The following filters correspond to the quasi-interpolative case where $\psi = \vartheta^6$ and $\varphi = \vartheta^4$ (see Table 4.2 on page 84).

A.1 w_{b}

$(0,0,0)$	1	1865002207/16937496576
$(1,1,1)$	8	16154080177/338749931520
$(2,0,0)$	6	1157637193/31757806080
$(2,2,0)$	12	1409618989/127031224320
$(3,1,1)$	24	33424913/7258927104
$(2,2,2)$	8	346906663/112916643840
$(4,0,0)$	6	41079899/36294635520
$(3,3,1)$	24	18196601/63515612160
$(4,2,0)$	24	20701277/84687482880
$(4,2,2)$	24	373289/8065474560
$(5,1,1)$	24	992161/36294635520
$(3,3,3)$	8	211763/19926466560
$(6,0,0)$	6	225221/127031224320
$(4,4,0)$	12	479737/508124897280
$(5,3,1)$	48	83233/254062448640
$(4,4,2)$	24	1651/13028843520
$(6,2,0)$	24	3967/39086530560
$(6,2,2)$	24	1319/203249958912
$(5,3,3)$	24	3089/1016249794560
$(7,1,1)$	24	1627/1016249794560
$(4,4,4)$	8	1/10265149440
$(8,0,0)$	6	1/72589271040

This filter is obtained by sampling the rhombic dodecahedral box spline $\vartheta^{10}(\boldsymbol{x})$ at the lattice sites of $\mathcal{H}$. The weights are given above. Only the weights for the first octant are reported. Due to the symmetry of the rhombic dodecahedral box splines, all points obtained

by the sign flips and permutations of the coordinates, have equivalent weights. The number of equivalent points is given in the second column.

The closed-form expression for the DTFT $\hat{W}_{\mathtt{b}}(\frac{u}{2\pi}, \frac{v}{2\pi}, \frac{w}{2\pi})$ (in Matlab® syntax) is:

```
(27975033105 + 18522195088.*cos(2.*u) + 575118586.*cos(4.*u) + ...
900884.*cos(6.*u) + 7.*cos(8.*u) + 51571.*cos(2.*(u - 3.*v)) + ...
124207662.*cos(2.*(u - 2.*v)) + 124207662.*cos(4.*u - 2.*v) + ...
51571.*cos(6.*u - 2.*v) + 5638475956.*cos(2.*(u - v)) + ...
479737.*cos(4.*(u - v)) + 18522195088.*cos(2.*v) + 575118586.*cos(4.*v) +...
900884.*cos(6.*v) + 7.*cos(8.*v) + 5638475956.*cos(2.*(u + v)) + ...
479737.*cos(4.*(u + v)) + 124207662.*cos(2.*(2.*u + v)) + ...
51571.*cos(2.*(3.*u + v)) + 124207662.*cos(2.*(u + 2.*v)) + ...
51571.*cos(2.*(u + 3.*v)) + 2.*cos(u).*cos(v).*(39449302405 + ...
3254.*cos(6.*u) + 8722458374.*cos(2.*v) + 2.*cos(4.*u).*(27445949 + ...
665864.*cos(2.*v)) + 54891898.*cos(4.*v) + 2.*cos(2.*u).*(4361229187 + ...
580959504.*cos(2.*v) + 665864.*cos(4.*v)) + 3254.*cos(6.*v)).*cos(w) + ...
2.*(9261097544 + 51571.*cos(6.*u) + (5638475956 + 6595.*cos(6.*u)).*...
cos(2.*v) + 124207662.*cos(4.*v) + 6.*cos(4.*u).*(20701277 + ...
7839069.*cos(2.*v) + 21463.*cos(4.*v)) + 51571.*cos(6.*v) + ...
cos(2.*u).*(5638475956 + 3122159967.*cos(2.*v) + 47034414.*cos(4.*v) + ...
6595.*cos(6.*v))).*cos(2.*w) + 2.*cos(u).*cos(v).*(4108656187 + ...
560037898.*cos(2.*v) + 2.*cos(4.*u).*(329843 + 6178.*cos(2.*v)) + ...
659686.*cos(4.*v) + 2.*cos(2.*u).*(280018949 + 21587470.*cos(2.*v) + ...
6178.*cos(4.*v))).*cos(3.*w) + 2.*(287559293 + 124207662.*cos(2.*v) + ...
479737.*cos(4.*v) + cos(4.*u).*(479737 + 128778.*cos(2.*v) + ...
99.*cos(4.*v)) + 6.*cos(2.*u).*(20701277 + 7839069.*cos(2.*v) + ...
21463.*cos(4.*v))).*cos(4.*w) + 2.*cos(u).*cos(v).*(27117733 + ...
659686.*cos(2.*v) + 2.*cos(2.*u).*(329843 + 6178.*cos(2.*v))).*cos(5.*w)...
+ 2.*(450442 + 51571.*cos(2.*v) + cos(2.*u).*(51571 + ...
6595.*cos(2.*v))).*cos(6.*w) + 3254.*cos(u).*cos(v).*cos(7.*w) + ...
7.*cos(8.*w))/2.5406244864e11;
```

A.2 a_{ϑ^4}

This is the autocorrelation sequence of the quintic box spline ϑ^4 and is obtained by sampling $\vartheta^8(\boldsymbol{x})$ at the lattice sites of $\mathcal{H}$. The non-redundant filter weights along with their multiplicities are as follows:

$(0,0,0)$	1	40853/270270
$(1,1,1)$	8	1396301/25945920
$(2,0,0)$	6	3029911/77837760
$(2,2,0)$	12	54889/6486480
$(3,1,1)$	24	9743/3538080
$(2,2,2)$	8	1613/1081080
$(4,0,0)$	6	2383/4864860
$(3,3,1)$	24	719/15567552
$(4,2,0)$	24	659/15567552
$(4,2,2)$	24	59/19459440
$(5,1,1)$	24	167/77837760
$(3,3,3)$	8	1/3706560
$(6,0,0)$	6	1/15567552

The closed-form expression for the DTFT $\hat{A}_{\vartheta^4}(\frac{u}{2\pi}, \frac{v}{2\pi}, \frac{w}{2\pi})$ (in Matlab® syntax) is:

```
(1/38918880).*...
(5882832 + 3029911.*cos(2.*u) + 38128.*cos(4.*u) + 5.*cos(6.*u) +...
3295.*cos(2.*(u - 2.*v)) + 3295.*cos(4.*u - 2.*v) + ...
658668.*cos(2.*(u - v)) + 3029911.*cos(2.*v) + 38128.*cos(4.*v) +...
5.*cos(6.*v) + 658668.*cos(2.*(u + v)) + 3295.*cos(2.*(2.*u + v)) +...
3295.*cos(2.*(u + 2.*v)) + 8.*cos(u).*cos(v).*(1882070 + ...
167.*cos(4.*u) + 210584.*cos(2.*v) + cos(2.*u).*(210584 + ...
7190.*cos(2.*v)) + 167.*cos(4.*v)).*cos(w) + (3029911 + ...
1317336.*cos(2.*v) + cos(4.*u).*(6590 + 944.*cos(2.*v)) + ...
6590.*cos(4.*v) + 8.*cos(2.*u).*(164667 + 58068.*cos(2.*v) + ...
118.*cos(4.*v))).*cos(2.*w) + 4.*cos(u).*cos(v).*(207177 + ...
7148.*cos(2.*v) + cos(2.*u).*(7148 + 84.*cos(2.*v))).*cos(3.*w) + ...
2.*(19064 + 3295.*cos(2.*v) + cos(2.*u).*(3295 + 472.*cos(2.*v))).*...
cos(4.*w) + 668.*cos(u).*cos(v).*cos(5.*w) + 5.*cos(6.*w));
```