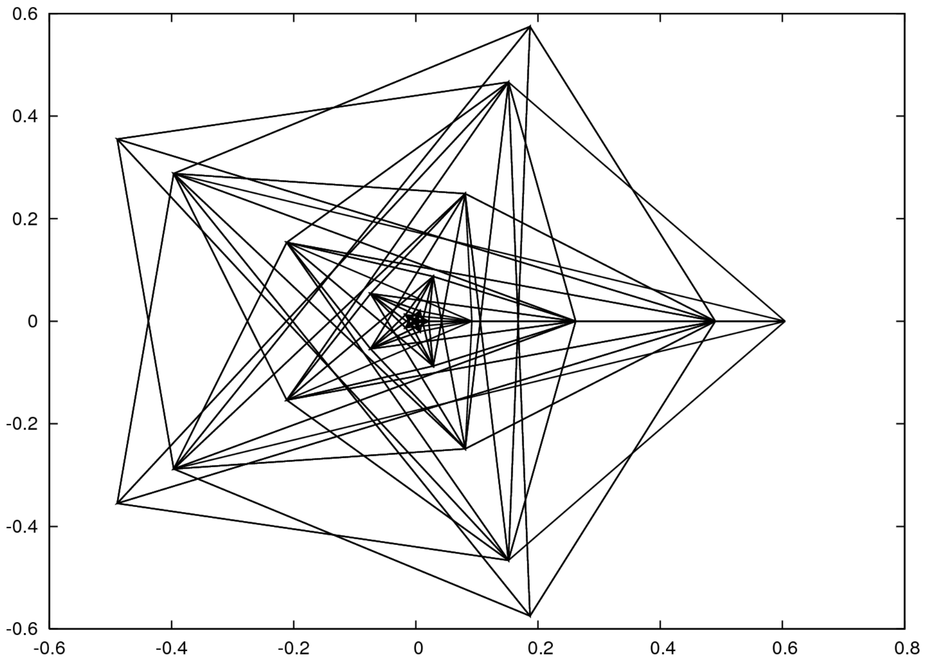
Finite Discrete Gabor Analysis

Søndergaard, Peter Lempel; Hansen, Per Christian; Christensen, Ole

Publication date:
2007



Finite Discrete Gabor Analysis

Peter L. Søndergaard

Institut for Matematik – DTU - 2007

```
g=pgauss(15);
c=reshape(eye(25),5,5,25);
F=idgt(c,g,3);
F=[F;F(L,:)];
plot(F);
```

Abstract

Gabor analysis is a method for analyzing signals through the use of a set of basic building blocks. The building blocks consists of a certain function (the window) that is shifted in time and frequency. The Gabor expansion of a signal contains information on the behavior of the signal in certain frequency bands at certain times.

Gabor theory can be formulated for both functions on the real line and for discrete signals of finite length. The two theories are largely the same because many aspects come from the same underlying theory of locally compact Abelian groups.

The two types of Gabor systems can also be related by sampling and periodization. This thesis extends on this theory by showing new results for window construction. It also provides a discussion of the problems associated to discrete Gabor bases.

The sampling and periodization connection is handy because it allows Gabor systems on the real line to be well approximated by finite and discrete Gabor frames. This method of approximation is especially attractive because efficient numerical methods exists for doing computations with finite, discrete Gabor systems. This thesis presents new algorithms for the efficient computation of finite, discrete Gabor coefficients.

Reconstruction of a signal from its Gabor coefficients is done by the use of a so-called dual window. This thesis presents a number of iterative algorithms to compute dual and self-dual windows.

The Linear Time Frequency Toolbox is a Matlab/Octave/C toolbox for doing basic discrete time/frequency and Gabor analysis. It is intended to be both an educational and a computational tool. The toolbox was developed as part of this Ph.D. project to provide a solid foundation for the field of computational Gabor analysis.

Dansk resumé

Gabor analyse er en metode til at analysere signaler ved brug af nogle basale 'byggesten'. Byggestenene består af en funktion (vinduet), som flyttes i både tid og frekvens. En Gabor ekspansion af et signal indeholder information om signalets opførsel i bestemte frekvensbånd til bestemte tidspunkter.

Gabor teori kan formuleres både for funktioner på den reele akse og for endeligt diskrete signaler. De to teorier minder meget om hinanden fordi de udspringer af den samme grundlæggende teori om lokalkompakte abelske grupper.

De to slags Gabor systemer kan også relateres via sampling og periodisering. Denne afhandling udbygger denne teori og viser nye resultater indenfor konstruktion af vinduer, samt giver en diskussion af problemerne relaterede til endeligt diskrete Gabor baser.

Relationen via sampling og periodisering er nyttig fordi den kan bruges til at approksimere Gabor systemer på den reele akse ved brug af endeligt diskrete Gabor systemer. Denne approksimationsmetode er særligt attraktiv fordi der findes effektive numeriske metoder til beregning med endeligt diskrete Gabor systemer. Denne afhandling præsenterer nye algoritmer til effektiv beregning af Gabor koefficienter for endelige signaler.

Rekonstruktion af et signal udfra dets Gabor koefficienter kan ske ved brug af et såkaldt dualt vindue. I afhandlingen præsenteres en række iterative algoritmer til beregning af duale og selv-duale vinduer.

"The Linear Time Frequency Toolbox" er en programpakke skrevet i Matlab/Octave/C til simpel beregning af diskret tids/frekvens- og Gabor analyse. Den er ment som både et undervisningredskab og som et beregningsmæssigt værktøj. Programpakken er udviklet som en del af dette ph.d. projekt for a give et solidt fundament for numerisk Gabor analyse.

Preface

This thesis is submitted in partial fulfilment of the requirements for obtaining the Ph.D.-degree. The work has been carried out at the Department of Mathematics, Technical University of Denmark from September 2002 to April 2007 under the supervision of Docent Ole Christensen, MAT, DTU and Professor Per Christian Hansen, IMM, DTU. The Ph.D. study was funded by the Danish Technical Research Council through the WAVES programme, `http://www.control.aau.dk/waves/`.

Acknowledgements

I would like to thank my advisors Per Christian Hansen and Ole Christensen; Per Christian for teaching me the importance of good programming, numerics and creative hand-waving and Ole for teaching me the accuracy and rigor of a mathematician.

During my travel abroad I stayed with the Numerical Harmonic Analysis Group at the Department of Mathematics, Vienna. I would like to thank Hans Feichtinger for welcomming me and for all the help and inspiration he has given me, all the people at NuHAG for their hospitality and discussions. In particular I wish to thank Hans Feichtinger and Karlheinz Gröchenig for organizing the "special semester on modern time-frequency analysis" during 2005.

I wish to thank my coauthers A.J.E.M Janssen, Bruno Torrésani and Peter Balazs providing me the gift of coorperation. Without your ideas I would only have gotten half the way.

Finally I wish to thank my wife and children for keeping my mind to better things than computers and for comming with me to Vienna.

Overview

The thesis is build up of three lines of interest in the field of Gabor analysis:

1. The connection between Gabor systems for $L^2(\mathbb{R})$, $l^2(\mathbb{Z})$, $L^2(\mathbb{T})$ and $\mathbb{C}^L$. Research in this direction is presented in Chapter 2 and 3.

2. Efficient algorithms for Gabor systems for $\mathbb{C}^L$. Research in this direction is presented in Chapter 4 and 5.

3. The construction of a Gabor toolbox. An early version of a paper on the toolbox is presented in Chapter 6 and the reference manual for the toolbox is included as Appendix B.

Chapter 1 provides a quick introduction to Gabor analysis necessary for the understanding of the thesis and provides a general discussion of the results presented in the rest of the thesis.

Appendix A presents a small taxonomy of symmetric Gabor and Wilson transforms implemented by the toolbox.

Most of the research presented in this thesis have previously been published / submitted for publication. The papers are in order of submission:

- Søndergaard, P.L. *Gabor Frames by Sampling and Periodization.* Published online by Advances in Computational Mathematics, 2007. Presented in this thesis as Chapter 2.

- Janssen, A.J.E.M and Søndergaard, P.L. *Iterative algorithms to approximate canonical Gabor windows: Computational aspects.* Published online by Journal of Fourier Analysis and Applications, 2007. Presented in this thesis as Chapter 5.

- Søndergaard, P.L. *Symmetric, discrete fractional splines and Gabor systems.* Submitted to International Journal of Wavelets, Multiresolution and Information Processing. Presented in this thesis as Chapter 3.

- Søndergaard, P.L. *An Efficient Algorithm for the Discrete Gabor Transform using full length Windows.* Submitted to IEEE Signal Processing Letters. Presented in this thesis as Chapter 4.

- Søndergaard, P.L, Torresani, B. and Balazs, P. *Algorithm XXX: The Linear Time Frequency Analysis Toolbox.* Will be submitted to Transactions on Mathematical Software. Presented in the thesis as Chapter 6.

Peter L. Søndergaard
March, 2007

Contents

1 Gabor analysis **11**

1.1 Mathematical background . . . 11

1.1.1 Fourier transformation . . . 11

1.1.2 Frames and Gabor systems . . . 11

1.1.3 Window classes . . . 13

1.1.4 Zak transforms and Gabor analysis . . . 15

1.2 From the continuous to the discrete . . . 15

1.2.1 Window functions by Poisson summation . . . 17

1.2.2 Discrete Balian Low . . . 20

1.3 Efficient algorithms . . . 23

1.3.1 Factorization methods . . . 23

1.3.2 Iterative algorithms . . . 24

2 Gabor frames by sampling and periodization **27**

2.1 Abstract . . . 27

2.2 Introduction . . . 27

2.3 Basic theory . . . 29

2.3.1 The Fourier transforms . . . 29

2.3.2 The translation, modulation and dilation operators . . . 29

2.3.3 Window classes . . . 30

2.3.4 The sampling and periodization operators . . . 31

2.3.5 Frames for Hilbert spaces . . . 32

2.3.6 Gabor systems . . . 33

2.4 Between the spaces . . . 33

2.4.1 From $L^2(\mathbb{R})$ to $l^2(\mathbb{Z})$. . . . 34

2.4.2 From $L^2(\mathbb{R})$ to $L^2([0,L])$ . . . 35

2.4.3 From $L^2([0,L])$ to $\mathbb{C}^L$ . . . 36

2.4.4 From $l^2(\mathbb{Z})$ to $\mathbb{C}^L$ . . . 37

2.4.5 From $L^2(\mathbb{R})$ to $\mathbb{C}^L$ . . . 37

2.5 Going back . . . 38

2.6 Proofs . . . 40

2.6.1 Some additional theory. . . . 40

2.6.2 Proofs of section 2.4.2 . . . 42

3 Symmetric, discrete fractional splines and Gabor systems 43
3.1 Abstract . . . 43
3.2 Introduction . . . 43
3.3 Definitions . . . 44
3.4 Symmetric splines for the real line . . . 46
3.5 Discrete splines by sampling and periodization . . . 49
3.6 Discrete splines by the DFT . . . 51
3.7 Numerical results . . . 52
3.8 Conclusion . . . 56
3.9 Derivation of the periodic, discrete fractional splines by sampling and periodization . . . 56

4 An Efficient Algorithm for the Discrete Gabor Transform using full length Windows 59
4.1 Abstract . . . 59
4.2 Introduction . . . 59
4.3 Definitions . . . 60
4.4 The method . . . 61
4.5 Extensions . . . 65
4.6 Special cases . . . 65
4.7 Efficient implementation . . . 65

5 Iterative algorithms to approximate canonical Gabor windows: Computational aspects 67
5.1 Abstract . . . 67
5.2 Introduction . . . 67
5.3 Paper outline and results . . . 71
5.4 Frame operator calculus and basic inequalities . . . 72
5.5 Norm scaling . . . 73
5.5.1 Iterations for approximating g^t . . . 73
5.5.2 Iterations for approximating g^d . . . 74
5.6 Initial scaling . . . 75
5.7 Considerations in the adjoint orbit space . . . 77
5.7.1 Estimate for upper frame bound . . . 79
5.7.2 Error measure . . . 79
5.7.3 Influence of out-of-space components . . . 80
5.8 Zak domain considerations . . . 81
5.9 Sampling and periodization of Gabor frames . . . 83
5.10 Implementational aspects . . . 84
5.10.1 Matrix representation and the SVD . . . 85
5.10.2 Factorization of finite, discrete Gabor systems . . . 86
5.10.3 Other methods . . . 87
5.10.4 Implementational costs . . . 88
5.10.5 Stopping criterion . . . 89
5.10.6 Window functions . . . 89
5.11 Experiments . . . 89
5.11.1 Convergence and divergence of norm scaling . . . 90

5.11.2 Comparison with other methods 92
5.11.3 Number of iterations 94
5.11.4 Choosing an initial scaling 95
5.11.5 Summary from the numerical experiments 95
5.12 A result on Condition A' 96

6 Algorithm XXX: The Linear Time Frequency Toolbox 98
6.1 Abstract 98
6.2 Introduction 98
6.3 Basic Fourier and time-frequency analysis 101
6.4 Time/Frequency transforms 102
6.4.1 The discrete Gabor transform 102
6.4.2 The discrete Wilson transform 104
6.4.3 The modified discrete cosine transform (MDCT) 105
6.5 Window design 105
6.5.1 FIR windows 106
6.5.2 IIR windows 107
6.5.3 Symmetries 107
6.5.4 Dual / tight windows 107
6.5.5 Non-canonical dual window 108
6.6 Other tools 108
6.6.1 Time variant filtering 108
6.6.2 Plotting routines 108
6.7 Signal processing tasks 109
6.7.1 Compression 109
6.7.2 Denoising 109
6.7.3 Feature extraction using formants 109
6.7.4 OFDM transmission 110
6.8 Implementation 110

7 Symmetric Gabor and Wilson systems 111
7.1 Symmetries of discrete Fourier analysis 111
7.2 The symmetric DGT transforms 112
7.3 The symmetric Wilson transforms 112
7.3.1 DWILT 113
7.3.2 DWILTII 114
7.3.3 DWILTIII 116
7.3.4 DWILTIV 116

A Symmetric Gabor and Wilson systems 117
A.1 Symmetries of discrete Fourier analysis 117
A.2 The symmetric DGT transforms 118
A.3 The symmetric Wilson transforms 118
A.3.1 DWILT 119
A.3.2 DWILTII 120
A.3.3 DWILTIII 122
A.3.4 DWILTIV 122

B Reference manual **123**
B.1 The Linear Time-Frequency Analysis Toolbox. 124
B.1.1 Basic Fourier/TF analysis 124
B.1.2 Gabor systems. 128
B.1.3 Window construction for Gabor frames. 132
B.1.4 Wilson bases and MDCT. 135
B.1.5 Window functions. 144
B.1.6 Time Varying Filtering. 149
B.1.7 Conditions numbers. 150
B.1.8 Plots. 152
B.1.9 Cosine and Sine transforms. 155
B.2 LTFAT - Object oriented interface 160
B.2.1 Setup and deletion of object 160
B.2.2 Analysis and synthesis 161
B.2.3 Information 164
B.2.4 Condition numbers etc. 165
B.2.5 Multipliers 165
B.3 LTFAT - Signal processing tools 166
B.3.1 Working with coefficients. 166
B.3.2 Extra signal processing tools. 168
B.4 LTFAT - Examples 169
B.4.1 Simple examples 169
B.4.2 Applications 169
B.4.3 Aspect of particular functions 175
B.4.4 Misc: 179
B.5 LTFAT - Signals 181
B.5.1 Sound signals. 181
B.5.2 Images. 182

Chapter 1

Gabor analysis

1.1 Mathematical background

In this section we will make a brief introduction to Gabor analysis necessary for the understanding of the thesis. The material is mostly taken from [41]. The book [41] provides an excellent exposition of Gabor analysis on $L^2\left(\mathbb{R}^d\right)$.

Fourier and Gabor analysis can be formulated in terms of *locally compact Abelian* (LCA) groups. An LCA group can be seen as an abstraction of the spaces $\mathbb{R}$, $\mathbb{Z}$, $\mathbb{T}$ or $\mathbb{C}^L$ and many other spaces. The benefit of using LCA groups is that theory developed for LCA groups will work for any of the four spaces just mentioned. In the introduction to [41], it is noted that almost all material in the book (except for certain sections) hold for function spaces on LCA groups and not just for $L^2(\mathbb{R})$.

In this thesis we will not pursue the LCA connection, but just mention that it is responsible for the similarities of many of the formulas shown.

1.1.1 Fourier transformation

The basis of all time-frequency analysis is the Fourier transform.

The Fourier transform $\mathcal{F}_{\mathbb{R}} : \left(L^1 \cap L^2\right)(\mathbb{R}) \mapsto L^2(\mathbb{R})$ is defined by

$$\left(\mathcal{F}_{\mathbb{R}} f\right)(\omega) = \int_{\mathbb{R}} f(x) e^{-2\pi i \omega x} dx \tag{1.1}$$

The Fourier transform $\mathcal{F}_{\mathbb{R}}$ can be extended to a bounded, linear, unitary operator, $\mathcal{F}_{\mathbb{R}} : L^2(\mathbb{R}) \to L^2(\mathbb{R})$.

The Discrete Fourier Transform (DFT) of a signal $f \in \mathbb{C}^L$ is defined by

$$\left(\mathcal{F}_L f\right)(k) = \frac{1}{\sqrt{L}} \sum_{l=0}^{L-1} f(l) e^{-2\pi i k l / L}. \tag{1.2}$$

1.1.2 Frames and Gabor systems

A *frame* is an extension of the mathematical concept of a basis. All bases are frames, but a frame might not be a basis. One of the defining characteristics of as basis is that it is linearly independent. A frame might not be linearly independent, but in return it can be *overcomplete*, i.e. consist of more vectors than strictly neccesary to span a space. Frames are thoroughly treated in the book [23].

The precise definition of a frame is as follows:

Definition 1.1.1. A family of elements $\{e_j\}_{j\in J}$ in a separable Hilbert space $\mathcal{H}$ is called a frame if constants $0 < A \leq B < \infty$ exist such that

$$A\,\|f\|_{\mathcal{H}}^2 \leq \sum_{j\in J} \left|\langle f, e_j\rangle_{\mathcal{H}}\right|^2 \leq B\,\|f\|_{\mathcal{H}}^2, \quad \forall f \in \mathcal{H}. \tag{1.3}$$

The constants A and B are called lower and upper frame bounds, respectively.

Two frames for a Hilbert space $\mathcal{H}$, $\{e_j\}$ and $\{f_j\}$, are called *dual frames* if and only if $f = \sum_{j\in J} \langle f, e_j\rangle f_j, \quad \forall f \in \mathcal{H}$.

The *frame operator* of a frame $\{e_j\}_{j\in J}$ for a Hilbert space $\mathcal{H}$ is defined by

$$S : \mathcal{H} \to \mathcal{H} \;\; : \;\; Sf = \sum_j \langle f, e_j\rangle_{\mathcal{H}}\, e_j, \tag{1.4}$$

where the series defining Sf converges unconditionally for all $f \in \mathcal{H}$.

The condition (1.3) ensures that the frame operator is both bounded and invertible on $\mathcal{H}$.

The inverse frame operator can be used to give a decomposition of any function $f \in \mathcal{H}$:

$$f = \sum_j \left\langle f, S^{-1}e_j\right\rangle e_j, \quad \forall f \in \mathcal{H}. \tag{1.5}$$

The frame $\{S^{-1}e_j\}$ is known as the canonical dual frame.

A *tight* frame is a frame where the frame bounds are equal, $A = B$. If $A = B = 1$ the frame is known as a normalized tight frame, or as a Parseval frame, because the frame condition (1.3) resembles the Parseval equality for orthogonal bases.

The advantage of a tight frame is that reconstruction can be done by the frame itself:

$$f = \frac{1}{A}\sum_j \langle f, e_j\rangle e_j, \quad \forall f \in \mathcal{H}. \tag{1.6}$$

A special tight frame related to a frame $\{e_j\}$ is the *canonical tight frame* given by $\left\{S^{-1/2}e_j\right\}$.

A Gabor system (g, α, β) is a family of functions $g_{m,n} \in L^2(\mathbb{R})$ of the following form

$$g_{m,n}(x) = e^{2\pi i m\beta x} g(x - n\alpha), \quad m, n \in \mathbb{Z},$$

where $g \in L^2(\mathbb{R})$ and $\alpha, \beta > 0$.

A Gabor system that is also a frame for $L^2(\mathbb{R})$ is called a *Gabor frame.* The density of a Gabor frame is given by $\alpha\beta$. Three important cases for the value of $\alpha\beta$ are given by

$\alpha\beta > 1$: The *undersampled* case. In this case a Gabor system cannot be a basis or a frame.

$\alpha\beta = 1$: The *critially sampled* case. In this case, if a Gabor system is frame then it is also a basis.

$\alpha\beta < 1$: The *oversampled* case. In this case, a Gabor system cannot be a basis, but it may very well be a frame.

The Gabor frame operator S commutes with time and frequency shifts and this property makes the canonical dual and canonical tight frames of a Gabor frame again be Gabor frames. The big advantage of this is that in order to find the canonical dual or tight frame, we do not need to apply S^{-1} or $S^{-1/2}$ to all frame elements, but only to the window in order to compute the window g^d of the canonical dual frame or the window g^t of the canonical tight frame:

$$g^d = S^{-1}g, \tag{1.7}$$

$$g^t = S^{-1/2}g. \tag{1.8}$$

These two windows are known as the canonical dual and canonical tight windows.

Gabor analysis for $\mathbb{C}^L$ can be defined entirely similar to that of $L^2(\mathbb{R})$. We denote for $g \in \mathbb{C}^L$ and $a, b \in \mathbb{N}$ by (g, a, b) the collection of time-frequency shifted windows

$$g_{na,mb}, \quad n \in \mathbb{Z}, m \in \mathbb{Z}, \tag{1.9}$$

where for $j, k \in \mathbb{Z}$ we denote

$$g_{j,k} = e^{2\pi ikl/L}g(l-j), \quad l = 0, \ldots L-1. \tag{1.10}$$

Note that it must hold that $L = Na = Mb$ for some $M, N \in \mathbb{N}$. Gabor system can be written as

$$\frac{ab}{L} = \frac{a}{M} = \frac{b}{N}. \tag{1.11}$$

The density divides Gabor systems into undersampled, critically sampled and oversampled systems just as in the $L^2(\mathbb{R})$ case.

The *Discrete Gabor transform* of a signal $f \in \mathbb{C}^L$ is the coefficients $c \in \mathbb{C}^{M\times N}$ computed by

$$c(m,n) = \langle f, g_{na,mb} \rangle \tag{1.12}$$

$$= \sum_{l=0}^{l=L-1} e^{-2\pi ilmb/L} f(l)\overline{g(l-na)}. \tag{1.13}$$

The Gabor frames presented in these section uses a rectangular grid in time and frequency. It is possible to generalize Gabor system to be formulated for general (non-separable) lattices, see Figure 1.1. The figure shows the position of the atoms of a Gabor frame in the time-frequency plane for both a rectangular and a quinquix (hexagonal) lattice. We shall not work with these kind of Gabor frames in this thesis. They are described in the $L^2(\mathbb{R})$ setting in [41, Chapter 9] and discrete Gabor systems for non-separable lattice are described in the papers [102, 103, 32, 65].

1.1.3 Window classes

We define the Wiener space by

$$W(\mathbb{R}) = \left\{ f \left| \sum_{n\in\mathbb{Z}} \underset{x \in [0,1]}{\text{esssup}} |f(x+n)| < \infty \right. \right\}.$$

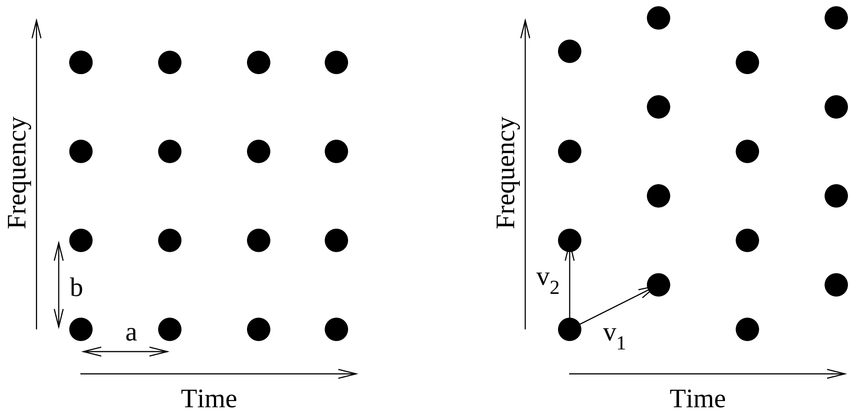


Figure 1.1: A rectangular and a skew (quinqux) grid in the time-frequency plane.

If both $f, \hat{f} \in W(\mathbb{R})$ then we write that $f \in (W \cap \mathcal{F}W)(\mathbb{R})$.

If $f \in (W \cap \mathcal{F}W)(\mathbb{R})$ then f is continuous and any regular sampling of f belongs to $l^1(\mathbb{Z})$. This can be stated formally by:

Definition 1.1.2. The sampling operator $S_\alpha : f \in (W \cap \mathcal{F}W)(\mathbb{R}) \mapsto l^1(\mathbb{Z})$ for $\alpha > 0$ is given by

$$(\mathcal{S}_\alpha f)(j) = \sqrt{\alpha} f(j\alpha), \quad \forall j \in \mathbb{Z}. \tag{1.14}$$

Very similarly, then a periodization of f will be absolutely convergent and the result will belong to $C(\mathbb{T})$, the space of continuous functions on the torus:

Definition 1.1.3. The periodization operator $\mathcal{P}_L : (W \cap \mathcal{F}W)(\mathbb{R}) \to \mathcal{A}([0, L])$ for $L > 0$ is given by

$$\mathcal{P}_L g(x) = \sum_{k \in \mathbb{Z}} g(x + kL), \quad x \in [0, L], \tag{1.15}$$

where $\mathcal{A}([0, L])$ is the space of functions on $[0, L]$ which have an absolutely convergent Fourier series.

This makes $(W \cap \mathcal{F}W)(\mathbb{R})$ a well suited space for working with samplings and periodizations.

Definition 1.1.4. A function $g \in L^2(\mathbb{R})$ belongs to *Feichtinger's algebra* $S_0(\mathbb{R})$ if

$$\|g\|_{S_0} = \int_{\mathbb{R}\times\mathbb{R}} \left| \int_{\mathbb{R}} g(t) \overline{(M_\omega T_x \varphi)(t)} dt \right| dx d\omega < \infty, \tag{1.16}$$

where $\varphi(t) = e^{-\pi t^2}$ is the Gaussian function.

Replacing the Gaussian φ in the definition of S_0 with another non-zero window in the Schwartz class of smooth, exponentially decaying functions, will give an equivalent norm.

It holds that $S_0 \subset W \cap \mathcal{F}W$. An example of a function in $W \cap \mathcal{F}W$ but not in S_0 is presented in [68].

Showing membership of the space $W \cap \mathcal{F}W$ is usually easy because it involves a separate condition on the function and its Fourier transform. It is much harder for the space S_0, but extensive research has been done on this yielding many characterizations of S_0, see e.g. [37].

1.1.4 Zak transforms and Gabor analysis

For $f \in L^2(\mathbb{R})$ the Zak transform $Z_\alpha f$ by

$$(Z_\alpha f)(x,\omega) \;=\; \sqrt{\alpha}\sum_{k\in\mathbb{Z}} f(x-l\alpha)\, e^{2\pi i\omega k}, \quad x,\omega \in \mathbb{R}. \tag{1.17}$$

The Zak transform is quasi-periodic in its first variable and periodic in the second:

$$(Z_\alpha f)\left(x+\alpha n, \omega + \frac{m}{\alpha}\right) = e^{2\pi i\alpha n\omega} Z_\alpha (x,\omega). \tag{1.18}$$

Because of the quasi-periodicity relation (1.18), the value of Z_α is completely determined by its value on the fundamental domain $[0,\alpha)\times[0,\frac{1}{\alpha})$.

The Zak transform is closely related to critically sampled Gabor systems. The canonial dual window can be computed by a pointwise inversion of the Zak-transform and the frame bounds of a Gabor system $(g,1,1)$ are given by

$$A \;=\; \operatorname*{ess\,inf}_{x,\omega\in[0;1)} |Zg(x,\omega)|^2 \tag{1.19}$$

$$B \;=\; \operatorname*{ess\,sup}_{x,\omega\in[0;1)} |Zg(x,\omega)|^2 \tag{1.20}$$

If $g \in W \cap \mathcal{F}W(\mathbb{R})$ then Zg is continuous. By using this and the quasi-periodizity relations (1.18) it can be shown that if $g \in W \cap \mathcal{F}W(\mathbb{R})$ then Zg contains a zero in the unit sqaure. This means that $(g,1,1)$ cannot be a Gabor frame, because by (1.19) then the lower frame bound is zero.

The finite, discrete Zak transform $Z_K h$ for $h \in \mathbb{C}^L$ and $K \in \{0,...,L-1\}$ such that $\frac{L}{K} \in \mathbb{N}$ is define by

$$(Z_K h)(r,s) \;=\; \sqrt{\frac{K}{L}} \sum_{l=0}^{L/K-1} h(r-lK)\, e^{2\pi i s l K/L}, \quad r,s \in \mathbb{Z}. \tag{1.21}$$

Just as the Zak transform for $L^2(\mathbb{R})$, the finite, discrete Zak transform is quasi-periodic in its first variable and periodic in the second,

$$(Z_K h)\left(r+kK, s+l\frac{K}{L}\right) \;=\; e^{2\pi i k s K/L}(Z_K h)(r,s). \tag{1.22}$$

The values $(Z_K h)(r,s)$ of a finite, discrete Zak-transform on the fundamental domain $r = 0,\dots,K-1$, $s = 0,\dots,K/L-1$ can be calculated efficiently by K FFT's of length K/L. To obtain values outside the fundamental domain, the quasi-periodicity relation (1.22) can be used.

1.2 From the continuous to the discrete

In this section we will present a small survey on the relationsship between Gabor frames for continuous and discrete spaces. This includes a small discussion of the results published in [92, 94] (presented in the thesis as Chapter 2 and 3.

The *Poisson summation formula* states that for $f \in W \cap \mathcal{F}W$ then:

$$\hat{f}(n) = \int_{\mathbb{T}} \left(\sum_{k\in\mathbb{Z}} f(x+k) \right) e^{2\pi i n x} dx, \quad n \in \mathbb{Z} \tag{1.23}$$

In words this means that in order to compute a sampling of the Fourier transform of a function f, we can do this by computing the Fourier coefficients of a sampling of the function. The condition that $f \in W \cap \mathcal{F}W$ is only a convenient, sufficient condition, the formula holds under much weaker conditions.

The Poisson summation formula shall be our standard tool for tool for studying the relation between Gabor system for various spaces. This is done in the paper [92], printed in this thesis as Chapter 2.

In the paper [92] (Chapter 2) relationsships between Gabor frames for the four spaces $L^2(\mathbb{R})$, $l^2(\mathbb{Z})$, $L^2(\mathbb{T})$ and $\mathbb{C}^L$. The work in this paper builds mainly upon the work in [79, 53] by Orr and Janssen and was done in parallel with similar work presented in [62] by Kaiblinger.

The main result of Janssen presented in [53] is that if the window of a Gabor frame (g, α, β) for $L^2(\mathbb{R})$ satisfies certain conditions, then a regular sampling of this window will generate a Gabor frame (g^s, a, b) for $l^2(\mathbb{Z})$. The frame bounds for (g, α, β) will also be frame bounds for (g^s, a, b) and the canonical dual windows of (g, α, β) and (g^s, a, b) will also be related by sampling. In the end of [53] it is hinted how to prove similar results going from Gabor frames for $l^2(\mathbb{Z})$ to $\mathbb{C}^L$ by periodization.

The paper [92] generalizes (trivially) Janssen's results to hold for a greater range of values of α, β and extends the results to also cover the transition $L^2(\mathbb{R}) \to L^2(\mathbb{T})$ by periodization and $L^2(\mathbb{T}) \to \mathbb{C}^L$ by sampling.

Furthermore, the relationsship between Gabor coffiecients computed for the four spaces are presented. This is an extension of work presented in [79] for critically sampled Gabor systems. Basically, if $c_{m,n} = \langle f, g_{m,n} \rangle_{L^2(\mathbb{R})}$ and $d_{m,n} = \left\langle f^s, g^s_{m,n} \right\rangle_{l^2(\mathbb{Z})}$ then the cofficients d will be a periodization in the frequency variable of the coefficients c. This is closely related to the aliasing phenomena for Fourier coefficients, where high-order frequencies are aliased to low order frequencies. For the relationsship $L^2(\mathbb{R}) \to L^2(\mathbb{T})$ the coefficients are periodized in the time variable instead. These results holds under the condition that both $f, g \in S_0(\mathbb{R})$ (or the similar space to S_0 on $\mathbb{Z}$ or $\mathbb{T}$), because the proof uses the Poisson summation formula. While this is a natural condition on the window g, it is sometimes a too strong condition on f.

The last type of result presented in [92] is an extension of the result on canonical duals to also hold for non-canonical dual windows: if γ is a dual window of (g, α, β) then γ^s is a dual window of (g^s, a, b).

The method for going all the way from $L^2(\mathbb{R})$ to $\mathbb{C}^L$ by sampling and periozation is summed up in 2.4.13, which we repeat below:

Theorem. *Let $g \in S_0(\mathbb{R})$, $\alpha\beta = \frac{a}{M} = \frac{b}{N}$ and $Mb = Na = L$ with $a, b, M, N, L \in \mathbb{N}$ and assume that (g, α, β) is a Gabor frame for $L^2(\mathbb{R})$ with canonical dual window γ^0. Then $\left(g^{sd}, a, b\right)$ is a Gabor frame for $\mathbb{C}^L$ with the same frame bounds and canonical dual window*

$\gamma^{0,sd}$. *The two functions are given by*

$$g^{sd}(l) = \sqrt{\frac{\alpha}{a}} \sum_{k\in\mathbb{Z}} g\left(\frac{\alpha}{a}(l-kL)\right), \tag{1.24}$$

$$\gamma^{0,sd}(l) = \sqrt{\frac{\alpha}{a}} \sum_{k\in\mathbb{Z}} \gamma^0\left(\frac{\alpha}{a}(l-kL)\right), \tag{1.25}$$

for $l = 0, \ldots, L-1$.

In [62] Kaiblinger has developed a method for "going back", that is to consider a window $g \in S_0(\mathbb{R})$ of a Gabor frame (g, α, β) for $L^2(\mathbb{R})$, sample and periodize it to get $g^{sd} \in \mathbb{C}^L$, compute the canonical dual $\gamma^{0,sd} \in \mathbb{C}^L$ of $\left(g^{sd}, a, b\right)$ using finite dimensional methods and use an interpolation scheme to construct an approximation $\gamma_L^0 \in L^2(\mathbb{R})$ of the canonical dual window γ^0 of (g, α, β). Kaiblinger uses a spline interpolation to accomplish this, in [92] a similar method using the original Gabor frame (g, α, β) for interpolating functions is presented.

Kaiblingers method and the Gabor frame interpolating methods are efficient tools for computing an approximation to the canonical dual of a Gabor frame for $L^2(\mathbb{R})$. It would be most beneficial to also apply them to the case of non-canonical windows. This is unfortunately not directly possible because the results in [92] only relates some dual window of (g, α, β) to some dual window of (g^s, a, b) and not specific windows as in the case of the canonical duals.

1.2.1 Window functions by Poisson summation

In this section we shall use the Poisson summation formula to construct appropriate window functions for finite discrete Gabor frames.

Using the appropriate Poisson summation formulas one can define the sampling and periodization operator $Q_L : S_0(\mathbb{R}) \mapsto \mathbb{C}^L$:

$$\left(Q_L f\right)(l) = L^{-1/4} \sum_{k\in\mathbb{Z}} f\left(l/\sqrt{L} + k\sqrt{L}\right). \tag{1.26}$$

In [2, 62] it is proven that for $f \in (W \cap \mathcal{F}W)(\mathbb{R})$ then

$$Q_L\left(\mathcal{F}_{\mathbb{R}} f\right) = \mathcal{F}_L\left(Q_L f\right), \tag{1.27}$$

so Q_L relates the Fourier transform on $\mathbb{R}$ to the Fourier transform on $\mathbb{C}^L$ through sampling and periodization. We shall use Q_L as the general tool to construct appropriate window functions for Gabor frames for $\mathbb{C}^L$ from windows in $(W \cap \mathcal{F}W)(\mathbb{R})$. The operator Q_L is also related to Gabor systems because by using the dilation operator $D_w : L^2(\mathbb{R}) \to L^2(\mathbb{R})$ given by

$$\left(D_w f\right)(t) = (w)^{-1/4} \quad f\left(\frac{t}{\sqrt{w}}\right), \quad t \in \mathbb{R}. \tag{1.28}$$

then (1.24) can be written as

$$\left(Q_L D_{\frac{1}{L}\frac{a^2}{\alpha^2}} g\right)(l) = \sqrt{\frac{\alpha}{a}} \sum_{k\in\mathbb{Z}} g\left(\frac{\alpha}{a}(l-kL)\right), \quad l = 0, ..., L-1. \tag{1.29}$$

While the operator Q_L works very nicely with Fourier transforms and Gabor analysis, it is not a unitary operator. This is the cause of several problems, because if $g \in (W \cap \mathcal{F}W)(\mathbb{R})$ has unit $L^2(\mathbb{R})$-norm, then $Q_L g$ is not guaranteed to have unit norm. For a window $g^{sd} = Q_L g \in \mathbb{C}^L$ comming from a unit norm window g one must decide whether to normalize g in accordance with Q and the Gabor sampling/periodization theory, or to make it unitary in the $\mathbb{C}^L$ sense.

Similar problems arise in connection with orthogonality and convolution, see discussion of Hermite and splines windows in Sections 1.2.1.3 and 1.2.1.2.

1.2.1.1 Gaussian windows.

In this section we use $\exp(x)$ for e^x to increase readability.

We wish to construct a periodic, discrete Gaussian function of length L. The starting point in the Gaussian $\varphi \in S_0(\mathbb{R})$ given by

$$\varphi(t) = \left(\frac{1}{2}\right)^{-1/4} \exp\left(-\pi t^2\right), \quad t \in \mathbb{R}. \tag{1.30}$$

In order to generate a suitable range of functions, we use the dilation operator (1.28):

$$(D_w\varphi)(t) = \varphi_w(t) = \left(\frac{w}{2}\right)^{-1/4} \exp\left(-\pi\frac{t^2}{w}\right). \tag{1.31}$$

Finally we obtain the sampled and periodized version of this using the sampling and periodization operator Q_L:

$$\begin{aligned}\varphi_w^D(l) &= \left(\frac{wL}{2}\right)^{-1/4} \sum_{k\in\mathbb{Z}} \exp\left(-\pi\frac{\left(\frac{l}{\sqrt{L}} + k\sqrt{L}\right)^2}{w}\right) &(1.32)\\ &= \left(\frac{wL}{2}\right)^{-1/4} \sum_{k\in\mathbb{Z}} \exp\left(-\pi\frac{(l+kL)^2}{wL}\right) &(1.33)\end{aligned}$$

for $l = 0, \ldots, L-1$. Because of (1.27) then the Discrete Fourier Transform of φ_w^D is $\varphi_{1/w}^D$.

The finite, discrete Gaussian constructed this way is whole point even (see Appendix A). For some applications it is beneficiel to construct a Gaussian that can be centered at different positions. We shift the dilated, continuous Gaussian (1.31) in time by $-c_t/\sqrt{L}$:

$$\varphi_{w,c_t}(t) = \left(\frac{w}{2}\right)^{-1/4} \exp\left(-\pi\frac{\left(t + \frac{c_t}{\sqrt{L}}\right)^2}{w}\right) \tag{1.34}$$

Finally, we sample and periodize this using Q_l:

$$\varphi^D_{w,c_f}(l) \quad = \quad \left(\frac{wL}{2}\right)^{-1/4} \sum_{k\in\mathbb{Z}} \exp\left(-\pi\frac{(l+c_t+kL)^2}{wL}\right) \qquad (1.35)$$

for $l = 0, \ldots, L-1$. This function is centered at $(-c_t, 0)$ in the finite, discrete TF-plane. It has exponential decay in both time and frequency, but it is only Fourier invariant if $c_t = 0$. If $c_t = \frac{1}{2}$, this function is half point even (see Appendix A).

The methods can also be used to construct finite, discrete window functions from hyperbolic secants. For a discussion of the role of hyperbolic secants in Gabor analysis, see [59].

1.2.1.2 Splines

In the paper [94] (Chapter 3 in this thesis), we define several classes of discrete, fractional splines. These are spline-like functions in $\mathbb{C}^L$ of possibly fractional orders. The paper builds on the research done by Unser and Blu in the papers [12, 13, 99, 100, 101] where they study fractional splines as window functions for wavelets and in connection with the study of fractional Brownian motion.

In [94] two classes of splines are developed:

1. The 'c' class. These splines are formed by sampling and periodization of their continuous counterparts using the operators D_w and Q_L. Because of the nature of Q_L, they do not satisfy the convolution properties normally associated to spline functions, but they do satisfy an important subsampling property.

2. The 'd' class. These splines are truly finite and discrete constructions made to satisfy the convolution properties by definition. On the other hand, they do not satisfy the subsampling property.

It is easy to show that any of the splines β^α defined in Definition 3.4.1 belong to $W \cap \mathcal{F}W$ for $\alpha > 0$ and that $\beta^\alpha \notin W \cap \mathcal{F}W$ for $\alpha \leq 0$.

Regarding $S_0(\mathbb{R})$ it can be easily shown using [37, Thm. 3.2.17] that $\beta^\alpha \in S_0(\mathbb{R})$ when $\alpha > \frac{\sqrt{2}-1}{2} \approx 0.207$. For $\alpha \leq 0$ then $\beta^\alpha \notin S_0(\mathbb{R})$ because of the similar result for $W \cap \mathcal{F}W$. This leaves the interval $\left(0, \frac{\sqrt{2}-1}{2}\right)$ for which it is currently not know if $\beta^\alpha \in S_0(\mathbb{R})$. If it should turn out that $\beta^\alpha \notin S_0(\mathbb{R})$ for any value $0 < \alpha < \frac{\sqrt{2}-1}{2}$ then it would produce a simple example of a function in $W \cap \mathcal{F}W$ but not in $S_0(\mathbb{R})$ to complement the only other known construction in [68].

1.2.1.3 Hermite functions

Hermite functions (sometimes called Gauss-Hermite functions) are a set of eigenfunctions for the Fourier transform given by the product of a Hermite polynomial and the Gaussian function. The zero'th order Hermite function is simply the Gaussian function.

The discrete Hermite functions are much harder to define. In the LTFAT toolbox we have included two definitions.

1. Define them as sampled and periodized versions of their continuous counterparts using the operator Q_L. These functions resemble the continuous ones, and are eigenfunctions of the DFT. However, they are not orthonormal, and the straightforward way of computing them is numerically unstable for large orders.

2. Define them as eigenfunctions of an almost tridiagonal matrix that commutes with the DFT. When the order is not a multiple of 4, then this matrix has distinct eigenvalues, and therefore its eigenfunctions are uniquely determined and orthonormal. If they are sorted according to the corresponding eigenvalue, then this ordering corresponds to the ordering of the continuous ones, namely after the number of zero-crossings. If the order is a multiple of 4, then one eigenvalue has multiplicity two. This is fixed by finding a basis consisting of an even and an odd eigenvector. The procedure is decribed in detail in [80].

The first approach is suitable for generating a few, lower order Hermite functions intended as window functions for Gabor frames, the second approach is suitable for generating a whole set of basis vectors for $\mathbb{C}^L$. It should be noted that there are other methods for generating discrete Hermite functions, this is still an active field of research. An attractive approach in presented in [75] because it makes it possible to construct dilated functions as is done for the Gaussian by the parameter w in (1.33).

1.2.2 Discrete Balian Low

The classical Balian-Low theorem (BLT) states that if the Gabor system (g, α, β) with $\alpha = \beta = 1$ forms an orthonormal basis for $L^2(\mathbb{R})$ then

$$\left(\int_{\mathbb{R}} |xg(x)|^2 \, dx\right)\left(\int_{\mathbb{R}} |\omega \hat{g}(\omega)|^2 \, d\omega\right) = \infty, \tag{1.36}$$

which basically means that the window g cannot have good concentration in time and frequency at the same time. This restriction does not apply for Gabor frames, and this is the main motivation for considering Gabor frames instead of Gabor bases. Several variations of BLT exists, see i.e. [8, 41].

The BLT is a statement about Gabor systems for $L^2(\mathbb{R})$, and it is not directly transferrable to the finite, discrete case. One, obvious reason is that a sum of a finite number of terms cannot sum to infinity, so doing the obvious substitution of the integral with finite sums makes no sense.

However, the problem consist not just in finding a new formulation of the BLT that will work for finite, discrete systems. It is possible to find windows in $\mathbb{C}^L$ having exponential behaviour in both time and frequency that generates finite, discrete Gabor frames. One such example is presented in [8]. Another closely related example is to consider $\varphi^D_{w,0}$ and $\varphi^D_{w,1/2}$ defined by (1.35). Both functions have exponential decay in both time and frequency, but $\varphi^D_{w,1/2}$ generates a Gabor frame and $\varphi^D_{w,0}$ does not.

In the following, we shall consider this relation between a critically sampled Gabor system $(g, 1, 1)$ and the Gabor system (g^{sp}, a, b) with $b = L/a$ for $\mathbb{C}^L$. The following has been derived from [111]. We wish to relate the Zak-transform of g with the finite, discrete

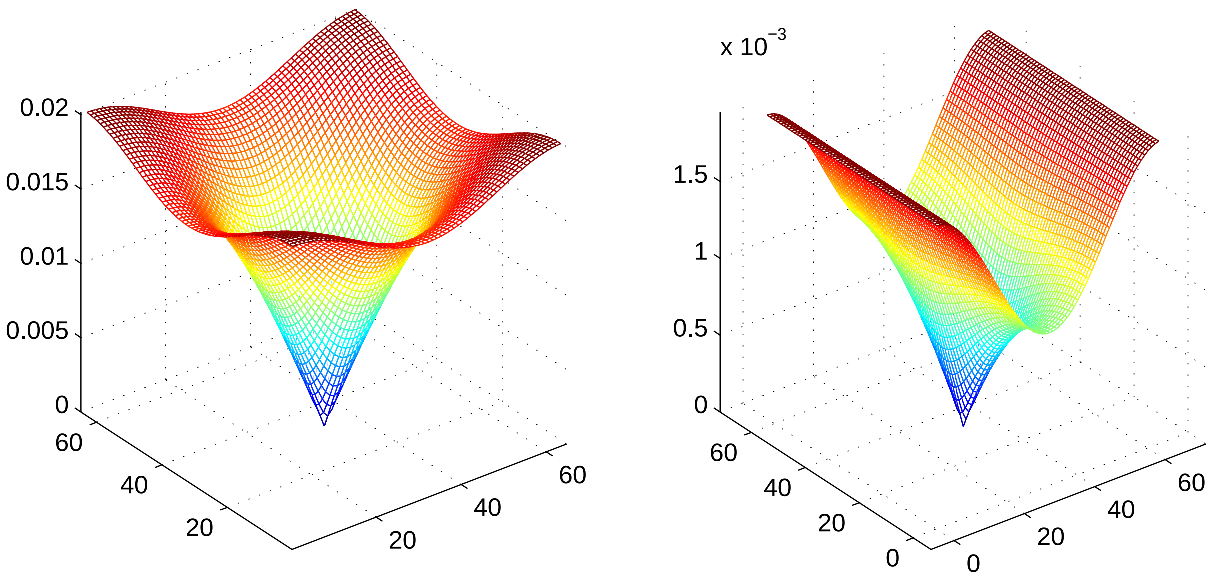


Figure 1.2: The figure shows the absolute value of the Zak transform of the finite, discrete Gaussian of length $L = 4096$ computed by `zak(pgauss(4096),64)` (to the left) and the Zak transform of a second order spline computed by `zak(pbspline('ec',4096,2,64),64)` (to the right). For both functions, the Zak transform is zero in a single point at $(32, 32)$.

Zak transform of

$$g^{sp}(k) = Q_L D_{a/b} g = \frac{1}{\sqrt{a}} \sum_{j\in\mathbb{Z}} g\left(\frac{k}{a} - jb\right), \quad k = 0, \ldots, L-1. \tag{1.37}$$

The Zak transform of g for $\alpha = 1$ is given by:

$$(Zg)(x, \omega) = \sum_{k\in\mathbb{Z}} g(x-k)\, e^{2\pi i\omega k}. \tag{1.38}$$

We consider the value of Zg sampled on a grid on the unit square:

$$(Zg)\left(\frac{r}{a}, \frac{s}{b}\right) = \sum_{k\in\mathbb{Z}} g\left(\frac{r}{a} - k\right) e^{2\pi iks/b}, \tag{1.39}$$

for $r = 0, \ldots, a-1$ and $s = 0, \ldots, b-1$. We split $k = l + jb$ with $l = 0, \ldots, b-1$, $l, r \in \mathbb{Z}$:

$$(Zg)\left(\frac{r}{a}, \frac{s}{b}\right) = \sum_{l=0}^{b-1} \sum_{j\in\mathbb{Z}} g\left(\frac{r}{a} - l - jb\right) e^{2\pi isl/b}$$

From this we can recognize $Z_K g^{sp}$:

$$(Zg)\left(\frac{r}{a}, \frac{s}{b}\right) = \sqrt{L} Z_a g^{sp}(r, s). \tag{1.40}$$

The relation (1.40) can be used two ways: It can be used to efficiently make a plot of the Zak transform of a continuous function, such as shown on Figure 1.2. This shows the

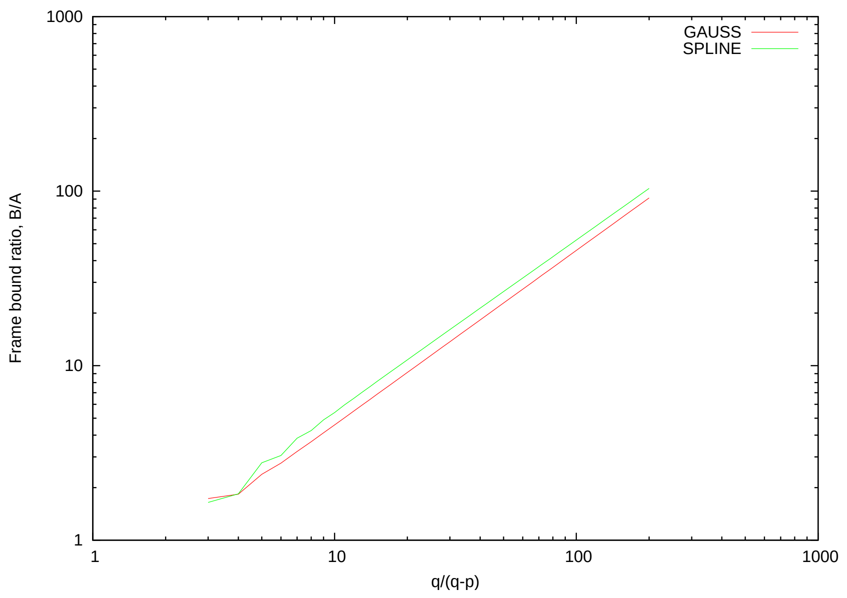


Figure 1.3: The figure show the connection between the inverse distance of the density to critical sampling given by $1/\left(1-\frac{p}{q}\right) = q/\left(q-p\right)$ and the frame bound ratio B/A. The experiment was done for Gabor frames $(g, a, a+1)$ for $\mathbb{C}^L$ with $L = (a+1)\, a$ for increasing a and windows `pgauss(L)` and `pbspline('ec'L,2,a)`.

finite, discrete Zak transform of a sampling and periodization of a Gaussian and a second order spline, and using (1.40) it can also be seen as a plot of the Zak transform of the functions on the unit square.

The relation (1.40) can also be used to explain the behaviour of the finite, discrete Zak transform based on knowledge of the continuous counterpart. This explains the single zero in the graph shown on Figure 1.2. This comes from the zero at $\left(\frac{1}{2}, \frac{1}{2}\right)$ in the unit square of the Zak transform of the Gaussian.

The relation (1.40) explains why the finite, discrete Gaussian does not generate a Gabor basis: It has a zero in its Zak-transform comming from the continuous Zak-transform. It also explains why $\varphi^D_{w,1/2}$ given by (1.35) do indeed generate a Gabor basis: The Zak transform of the continuous, shifted Gaussian is sampled just around the zero-point, but not in it. Therefore the Zak transform of $\varphi^D_{w,1/2}$ contains no zeros.

This discussion has shown that it is in fact quite easy to generate Gabor bases with windows constructed by sampling and periodization of $S_0(\mathbb{R})$ windows: We just have to avoid sampling the Zak-transform in a zero-point. This can be done by just shifting the function slightly in time or frequency. However, this presents no real solution, because we will still have to sample very close to the zero, and since the Zak-transform of $S_0(\mathbb{R})$ functions is continuous, we will get very small values in some sampling point, leading to a lower frame bound close to zero.

Another angle to consider is how the frame bound ratio B/A behaves for Gabor frames close to critical sampling. Figure 1.3 shows an investigation of what happens with the frame bound ratio B/A if we construct a series of Gabor frames with redundancy going to 1. Each point on the graph corresponds the frame bound ratio B/A of a Gabor frame (g^{sp}, a, a) for $\mathbb{C}^{a(a+1)}$ with a window g^{sp} formed by sampling and periodization of a continuous window. The two windows considered are φ_1^D given by (1.33) and $C_{L,a}^{2,eW}$ given by (3.30).

The density of such a frame is $\frac{p}{q} = \frac{a}{a+1}$ and goes to 1 as a increases. The first axis of Figure 1.3 shows the difference

$$\frac{1}{1-\frac{p}{q}} = \frac{q}{q-p} = a+1 \tag{1.41}$$

The figure shows that for $a \to \infty$ we have with strong correlation

$$\frac{B}{A} \approx C\frac{1}{1-\frac{p}{q}} = C\,(a+1)\,. \tag{1.42}$$

where C is some constant depending on the window. This shows that the frame bound ratio grows as we approach the limit of critical sampling.

1.3 Efficient algorithms

1.3.1 Factorization methods

One of the appealing aspects of Gabor analysis for finite, discrete systems is that there exists fast ($\mathcal{O}\,(L\log L)$) algorithms for analysis/synthesis and computation of canonical dual/tight windows. In this section we shall provide a little of the history of these methods, as well as a short discussion of the results published in [57, 93], Chapter 4 and 5 in this thesis.

The basic work was done by Zibulski and Zeevi in [111]. They showed that the Gabor frame operator for Gabor frames for $L^2\,(\mathbb{R})$ could be factorized into a set of products of $p\times q$ matrices, where $\frac{p}{q}$ is the redundancy of the frame written as an irreducible fraction. Each matrix depends continuously on two parameters $r, s \in \mathbb{R}$ and the elements in the matrices are given by elements of the Zak transform of the window. The approach naturally does not work for Gabor systems with irrational oversampling. They also provided a method for computing the Gabor coefficients, but since they worked in the $L^2\,(\mathbb{R})$ setting, this involves an integration, and it is not immidiatly apparent how to use this method in the $\mathbb{C}^L$ setting.

Zibulski and Zeevi's method was transferred to the finite, discrete case by Bastiaans and Geilen in [7] using the finite, discrete Zak transform. The finite, discrete Zak transform Z_a can be calculated efficiently on the fundamental domain $0,\ldots,a-1\times 0,\ldots,\frac{L}{a}-1$ by a series of DFTs. When values outside the fundamental domain is needed, the quasi-periodizity relation (1.22) must be used.

In [83] Prinz approached the problem in the more general context of Gabor systems with non-separable lattices, and he used matrix factorizations to present an incomplete factorization of the Gabor analysis and synthesis operators in the finite, discrete case. In the excellent survey paper [96] Strohmer improved this method for the Gabor frame operator, and ended up with a result similar to Bastians and Geilen and Zibulski and Zeevi: A system of products of $p \times q$ matrices.

The slight advantage Strohmer's method has is that is does not use the language of the Zak-transform, and so the use of (1.22) is not needed. Furthermore, it uses DFTs of shorter length than the Zak-transform approach.

In the paper [93] (Chapter 4) Strohmer's approach is extended to also cover the Gabor analysis and synthesis operators yielding the fastest known Gabor transform algorithm for general windows.

A further simple extension can be made on the algorithm presented in [93]. In [112] and in other places *multiwindow* Gabor systems are introduced. This is simply a regular Gabor system having not one, but several windows. All time-frequency shifts are still considered on the same grid. In the finite discrete case the definition of the multiwindow Gabor transform with R windows g_r, $r = 0, \ldots, R-1$ is

$$c(m,n,r) = \sum_{l=0}^{l=L-1} e^{-2\pi i l m b/L} f(l) \overline{g_r(l-na)}. \tag{1.43}$$

Extending the algorithm in [93] to multiwindows is very simple and it is shown as Algorithm 1.

If the window is a zero-extension of a relatively short window compared to the size of the signal then a more efficient approach is to use a filter bank approach. This simple algorithm is shown in (4.18). The fact that Gabor systems are related to filter banks was pointed out in [18].

To compute the canonical dual window of a Gabor frame a simple method is to calculate the Moore-Penrose pseudo-inverse of the Gabor analysis operator. This is a general method from frame theory, [21]. In the finite-discrete setting all operators are matrices, so this is simple to do. Because of the special structure of Gabor frames it is faster simply to invert the frame operator matrix and use the definition of the canonical dual window

$$g^d = S^{-1} g, \tag{1.46}$$

as a matrix inversion is computationally cheaper than a pseudo-inverse.

Taking the factorization method from [7, 93] into account, the picture changes: The $p \times q$ matrices of g^d are simpy the pseudo-inverses of the $p \times q$ matrices of g. This gives a very efficient algorithm for computing the canonical dual window.

Similar to this, it was shown in [57] that the $p \times q$ matrices of the canonical tight window g^t can be computed by the polar decomposition of the $p \times q$ matrices of the window g, (5.110). Computing the polar decomposition involves a Singular Value Decomposition (SVD) of the matrix.

1.3.2 Iterative algorithms

In order to reconstruct a signal f from its Gabor expansion, we might use the canonical dual window

$$f = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \langle f, g_{m,n} \rangle \, g^d. \tag{1.47}$$

One approach to find the window g^d is to compute it by a direct inversion $g^d = S^{-1} g$ or to consider it as the solution of the equation $S g^d = g$. Solving for g^d can be done by standard

Algorithm 1 Multisignal/multiwindow DGT by matrix/matrix products.

We wish to compute the DGT $c\left(m,n,t,w\right)\in\mathbb{C}^{M\times N\times R\times W}$ of $f\in\mathbb{C}^{L\times W}$ using the window $g\in\mathbb{C}^{L\times R}$ and lattice determined by a and M.

1. Define $\Psi^{f}_{r,s}\left(k,l\right)\in\mathbb{C}^{p\times qW}$ and $\Phi^{g}_{r,s}\left(k,l\right)\in\mathbb{C}^{p\times qR}$ for $r\in\langle c\rangle$, $s\in\langle d\rangle$, $t\in\langle R\rangle$ and $w\in\langle W\rangle$ by

$$\begin{aligned}\tilde{\Psi}^{f}_{r,\tilde{s}}\left(k,l+wq\right) &= f\left(r+kM+\tilde{s}pM-lh_a a,w\right),\\ \tilde{\Phi}^{g}_{r,\tilde{s}}\left(k,l+tq\right) &= \sqrt{Md}g\left(r+kM+\tilde{s}pM-la,t\right).\end{aligned}$$

2. Compute their DFTs along $\tilde{s}$:

$$\begin{aligned}\Psi^{f}_{r,s}\left(k,l+wq\right) &= \mathcal{F}_d\left(\tilde{\Psi}^{f}_{r,\cdot}\left(k,l+wq\right)\right)\left(s\right),\\ \Phi^{g}_{r,s}\left(k,l+tq\right) &= \mathcal{F}_d\left(\tilde{\Phi}^{g}_{r,\cdot}\left(k,l+tq\right)\right)\left(s\right).\end{aligned}$$

3. Multiply the matrices for each r,s:

$$\Upsilon_{r,s} = \left(\Phi^{g}_{r,s}\right)^{*}\Psi^{f}_{r,s}$$

4. Compute the inverse DFT of $\Upsilon_{r,s}$ along s:

$$\tilde{\Upsilon}_{r,\tilde{s}}\left(u+tq,l+wq\right)=\mathcal{F}_d^{-1}\left(\Upsilon_{r,\cdot}\left(u+tq,l+wq\right)\right)\left(\tilde{s}\right).$$

5. Compute $K\in\mathbb{C}^{M\times N\times W}$ as:

$$K\left(r+lc,u+\tilde{s}q-lh_a,t,w\right)=\tilde{\Upsilon}_{r,\tilde{s}}\left(u+tq,l+wq\right) \tag{1.44}$$

6. Finally, the result is given by a DFT of K along the first dimension:

$$c\left(m,n,t,w\right)=\mathcal{F}_M\left(K\left(\cdot,n,t,w\right)\right)\left(m\right). \tag{1.45}$$

iterative algorithms like the Neumann algorithm or the Conjugate Gradient Algorithm. The rationale is that it is much cheaper computationally to apply the frame operator S to a vector than it is to invert it. The Neumann algorithm using special preconditioning methods have been investigated in [4].

In the paper [57] (Chapter 5 in this thesis) a family of iterative algorithms with quadratic and quibic convergence is described. The work is based on previous results presented in [58, 55, 56]. The algorithms provides fast and simple methods to compute the canonical dual or the canonical tight window of a Gabor frame. They utilize the factorization methods presented in 1.3.1, in fact any of the factorization methods can be used. In [57] the factorization method used is that of Bastians and Geilen, mainly because [93] whas not yet written.

Chapter 2

Gabor frames by sampling and periodization

This chapter is the paper [92], which has been published online by Advances in Computational Mathematics, 2007.

2.1 Abstract

By sampling the window of a Gabor frame for $L^2(\mathbb{R})$ belonging to Feichtinger's algebra, $S_0(\mathbb{R})$, one obtains a Gabor frame for $l^2(\mathbb{Z})$. In this article we present a survey of results by R. Orr and A.J.E.M. Janssen and extend their ideas to cover interrelations among Gabor frames for the four spaces $L^2(\mathbb{R})$, $l^2(\mathbb{Z})$, $L^2([0,L])$ and $\mathbb{C}^L$. Some new results about general dual windows with respect to sampling and periodization are presented as well. This theory is used to show a new result of the Kaiblinger type to construct an approximation to the canonical dual window of a Gabor frame for $L^2(\mathbb{R})$.

2.2 Introduction

To compute an approximation to the Fourier transform of a function supported on an interval, a common technique is to sample the function regularly and compute the Discrete Fourier Transform of the obtained sequence. The result is an approximation to the Fourier transform of the original function computed at regular sampled points.

This relation was first transferred to the case of Gabor systems by Orr in [79]. He shows how the discrete Gabor coefficients of a sampled and periodized function can be calculated from a periodization in both time and frequency of their continuous counterparts. Orr only considers critically sampled Gabor systems.

In [53] Janssen shows that under certain conditions one can obtain a Gabor frame for $l^2(\mathbb{Z})$ by sampling the window function of a Gabor frame for $L^2(\mathbb{R})$ at the integers. Furthermore, it is shown that the canonical dual windows of the two Gabor frames are also related by sampling. Using the same techniques, he shows how to produce Gabor frames for $\mathbb{C}^L$ by periodizing the window function of a Gabor frame for $l^2(\mathbb{Z})$.

The results by Orr and Janssen covers the transition from the continuous setting to the discrete setting. Based on these results, Kaiblinger [62] has constructed a method to go backwards, that is to construct a continuous approximation to the canonical dual window of a Gabor frame for $L^2(\mathbb{R})$ from computations done purely in the discrete setting. He uses splines to reconstruct the canonical dual window from samples computed in the finite discrete setting.

The purpose of this article is to extend the results of Orr and Janssen to all of the four spaces $L^2(\mathbb{R})$, $l^2(\mathbb{Z})$, $L^2([0, L])$ and $\mathbb{C}^L$. We also provide new results about non-canonical dual windows. The situation is shown in Figure 2.1: We use these results to construct

$$\begin{tikzcd}
L^2(\mathbb{R}) \arrow[r, "sampling"] \arrow[d, "periodization"] & l^2(\mathbb{Z}) \arrow[d] \\
L^2([0, L]) \arrow[r] & \mathbb{C}^L
\end{tikzcd}$$

Figure 2.1: Relationship among the four different spaces for which we consider Gabor frames. Arrows to the right indicate sampling and arrows down indicate periodization.

a new method for computing approximations to the canonical dual window of a Gabor frame for $L^2(\mathbb{R})$. The method uses the same basic approach as that of Kaiblinger.

In Section 2.3 the required background and notation is presented. It includes the definitions of the various operators used, commutation relations between the operators, window classes for the Gabor frames, and definitions of frames and Gabor frames for each of the four spaces.

Section 2.4 presents a survey of how and under which conditions Gabor frames for the four spaces $L^2(\mathbb{R})$, $l^2(\mathbb{Z})$, $L^2([0, L])$ and $\mathbb{C}^L$ are interrelated by sampling and periodization operations.

For each relation in Figure 2.1, three types of results are presented:

1. If the window function of a Gabor frame belongs to S_0, then a sampling/periodization of the window generates a Gabor frame with the same parameters and frame bounds for one of the other spaces. The canonical dual windows of the two Gabor frames are similarly related by sampling/periodization. These are simple extensions of the results in [53]

2. For functions $f \in S_0$, simple relations hold between the expansion coefficients of f in a Gabor frame, and the expansion coefficients of the sampled/periodized function in the sampled/periodized Gabor frame. These are results of Orr from [79] made rigorous by Kaiblinger in [62].

3. If $f, \gamma \in S_0$ are dual windows of a Gabor frame, the sampled/periodized windows are also dual windows of the sampled/periodized Gabor frame. These are new results.

Proof of the statements in this section will be postponed (unless very short) to Sec. 2.6.

In the last section, Section 2.5, the new method to "go backwards" is presented. We show how to use the elements of a Gabor frame to construct an approximation to the canonical dual window of this frame.

2.3 Basic theory

The four spaces used in this article, $L^2(\mathbb{R})$,$L^2(\mathbb{Z})$,$L^2([0,L])$ and $\mathbb{C}^L$ share two common properties:

1. They are Hilbert spaces. Therefore common results from frame theory apply to them.
2. The domains $\mathbb{R}, \mathbb{Z}, [0,L]$ and $\mathbb{C}$ are locally compact Abelian (LCA) groups (the interval $[0,L]$ will always be thought of as a parameterization of the torus, $\mathbb{T}$). Most of the theory used in this article can be defined solely in terms of LCA-groups. This includes the Fourier transform, the translation and modulation operators, the sampling and periodization operators, Gabor systems and the space S_0. We will not use the LCA-definitions, but instead define everything for each of the four spaces.

2.3.1 The Fourier transforms

The proofs in this article will make frequent use of various types of Fourier transformations. They are defined by

$$\mathcal{F}_{\mathbb{R}} : \left(L^1 \cap L^2\right)(\mathbb{R}) \to L^2(\mathbb{R}) \quad : \quad \left(\mathcal{F}_{\mathbb{R}} f\right)(\omega) = \int_{\mathbb{R}} f(x) e^{-2\pi i \omega x} dx \tag{2.1}$$

$$\mathcal{F}_{\mathbb{Z}} : l^2(\mathbb{Z}) \to L^2([0,1]) \quad : \quad \left(\mathcal{F}_{\mathbb{Z}} f\right)(\omega) = \sum_{k \in \mathbb{Z}} f(k) e^{-2\pi i k x} \tag{2.2}$$

$$\mathcal{F}_{[0,L]} : L^2([0,L]) \to l^2(\mathbb{Z}) \quad : \quad \left(\mathcal{F}_{[0;L]} f\right)(k) = \frac{1}{\sqrt{L}} \int_0^L f(x) e^{-2\pi i k x / L} dx \tag{2.3}$$

$$\mathcal{F}_L : \mathbb{C}^L \to \mathbb{C}^L \quad : \quad \left(\mathcal{F}_L f\right)(k) = \frac{1}{\sqrt{L}} \sum_{j=0}^{L-1} f(j) e^{-2\pi i k j / L}. \tag{2.4}$$

$\mathcal{F}_{\mathbb{R}}$ is the Fourier transform, $\mathcal{F}_{\mathbb{Z}} c$ is known as the frequency domain representation of c, $\mathcal{F}_{[0,L]} f$ is the Fourier coefficients of f, and $\mathcal{F}_L$ is the Discrete Fourier Transform. The notation is heavy, but it helps to avoid confusion when several different transforms are involved.

The Fourier transform $\mathcal{F}_{\mathbb{R}}$ can be extended to a bounded, linear, unitary operator, $\mathcal{F}_{\mathbb{R}} : L^2(\mathbb{R}) \to L^2(\mathbb{R})$. With this extension, all the transforms are unitary operators.

For brevity, we shall use the notation $\mathcal{F}$ and $\hat{f}$ for $\mathcal{F}_{\mathbb{R}}$ and $\mathcal{F}_{\mathbb{R}} f$ *only*! We shall always write the other types of Fourier transformation with a subscript.

2.3.2 The translation, modulation and dilation operators

Gabor frames for the four spaces are defined in terms of the translation and modulation operators on the spaces.

Definition 2.3.1. The translation operators:

$$\begin{aligned}
T_t : L^2(\mathbb{R}) \to L^2(\mathbb{R}) \quad &: \quad (T_t f)(x) = f(x-t),\ x,t \in \mathbb{R} && (2.5)\\
T_j : l^2(\mathbb{Z}) \to l^2(\mathbb{Z}) \quad &: \quad (T_j f)(k) = f(k-j),\ k,j \in \mathbb{Z} && (2.6)\\
T_t : L^2([0,L]) \to L^2([0,L]) \quad &: \quad (T_t f)(x) = f(x-t),\ x,t \in [0,L] && (2.7)\\
T_j : \mathbb{C}^L \to \mathbb{C}^L \quad &: \quad (T_j f)(k) = f(k-j),\ k,j \in \{0,\dots,L-1\}. && (2.8)
\end{aligned}$$

The space $L^2([0,L])$ will always be thought of as a space of periodic functions, such that if $f \in L^2([0,L])$ then $f(x) = f(x+nL)$ for all $x \in \mathbb{R}$ and $n \in \mathbb{Z}$.

Definition 2.3.2. The modulation operators:

$$\begin{aligned}
M_\omega : L^2(\mathbb{R}) \to L^2(\mathbb{R}) \quad &: \quad (M_\omega f)(x) = e^{2\pi i\omega x} f(x),\ x,\omega \in \mathbb{R} && (2.9)\\
M_\omega : l^2(\mathbb{Z}) \to l^2(\mathbb{Z}) \quad &: \quad (M_\omega f)(j) = e^{2\pi i\omega j} f(j),\ j \in \mathbb{Z}, \omega \in [0,1] && (2.10)\\
M_k : L^2([0,L]) \to L^2([0,L]) \quad &: \quad (M_k f)(x) = e^{2\pi i kx/L} f(x),\ x \in [0,L],\ k \in \mathbb{Z} && (2.11)\\
M_k : \mathbb{C}^L \to \mathbb{C}^L \quad &: \quad (M_k f)(j) = e^{2\pi i kj/L} f(j),\ j,k \in \{0,\dots,L-1\}. && (2.12)
\end{aligned}$$

The space $L^2([0,L])$ only permit modulations with integer parameters, because the exponential factor must have period L. The modulation operator for $l^2(\mathbb{Z})$ is periodic in its parameter because $e^{2\pi i\omega j} = e^{2\pi i(\omega+n)j}$ for all integers j and n.

Definition 2.3.3. Let $d > 0$. The *dilation* operators

$$\begin{aligned}
D_d : L^2(\mathbb{R}) \to L^2(\mathbb{R}) \quad &: \quad (D_d f)(x) = \sqrt{d} f(dx), \quad \forall x \in \mathbb{R}. && (2.13)\\
D_d : L^2([0,L]) \to L^2([0,\frac{L}{d}]) \quad &: \quad (D_d g)(x) = \sqrt{d} g(dx), \quad x \in [0,\frac{L}{d}]. && (2.14)
\end{aligned}$$

2.3.3 Window classes

As a convenient window class for the four different types of Gabor frames we shall use the spaces $S_0(\mathcal{G})$, known as *Feichtinger's algebra,* which can be defined for LCA-groups, [30]. For the LCA-groups considered in this article these are the spaces $S_0(\mathbb{R})$, $l^1(\mathbb{Z})$, $\mathcal{A}([0,L])$ and $\mathbb{C}^L$. This gives window classes for $L^2(\mathbb{R})$, $l^2(\mathbb{Z})$, $L^2([0,L])$ and $\mathbb{C}^L$respectively.

Definition 2.3.4. A function $g \in L^2(\mathbb{R})$ belongs to *Feichtinger's algebra* $S_0(\mathbb{R})$ if

$$\|g\|_{S_0} = \int_{\mathbb{R}\times\mathbb{R}} \left| \int_{\mathbb{R}} g(t)\overline{(M_\omega T_x \varphi)(t)} dt \right| dx d\omega < \infty, \tag{2.15}$$

where $\varphi(t) = e^{-\pi t^2}$ is the Gaussian function.

Replacing the Gaussian φ in the definition of S_0 with another non-zero window in the Schwartz class of smooth, exponentially decaying functions, will give an equivalent norm. The space $S_0(\mathbb{R})$ is invariant under translation, modulation, dilation and the Fourier transform. Furthermore, $S_0(\mathbb{R}) \subseteq L^p(\mathbb{R})$ for $1 \le p \le \infty$. Functions in $S_0(\mathbb{R})$ are continuous. These and other useful properties of $S_0(\mathbb{R})$ can be found in [37] and [41]. There exists many useful conditions for membership of $S_0(\mathbb{R})$, see [37, 41, 76].

For Gabor frames on the interval $[0, L]$ we shall use the window class $S_0([0, L]) = \mathcal{A}([0, L]) = \mathcal{F}_{[0,L]}^{-1} l^1(\mathbb{Z})$. These are the functions on the interval $[0, L]$ having an absolutely convergent Fourier series. $\mathcal{A}([0, L])$ is a Banach space with respect to the norm

$$\|f\|_{\mathcal{A}([0,L])} = \left\| \mathcal{F}_{[0,L]}^{-1} f \right\|_{l^1(\mathbb{Z})}, \tag{2.16}$$

and is a Banach algebra with respect to point-wise multiplication. If $g \in \mathcal{A}([0, L])$ then g is a continuous function.

The Gabor coefficients of an S_0-function with respect to an S_0-window belongs to l^1. The precise relations are as follows, see [30, 37] and [41, Cor. 12.1.12].

Proposition 2.3.5. *Summability of Gabor coefficients.*

$S_0(\mathbb{R})$ *Let* $g, \gamma \in S_0(\mathbb{R})$ *and* $\alpha, \beta > 0$. *Then*

$$\sum_{m,n \in \mathbb{Z}} |\langle \gamma, M_{m\beta} T_{n\alpha} g \rangle| < \infty \tag{2.17}$$

$l^1(\mathbb{Z})$ *Let* $g, \gamma \in l^1(\mathbb{Z})$ *and* $a, M \in \mathbb{N}$. *Then*

$$\sum_{m=0}^{M-1} \sum_{n \in \mathbb{Z}} \left|\langle \gamma, M_{m/M} T_{na} g \rangle\right| < \infty \tag{2.18}$$

$\mathcal{A}([0, L])$ *Let* $g, \gamma \in \mathcal{A}([0, L])$ *and* $N, b \in \mathbb{N}$ *with* $L = Na$. *Then*

$$\sum_{m \in \mathbb{Z}} \sum_{n=0}^{N-1} |\langle \gamma, M_{mb} T_{na} g \rangle| < \infty \tag{2.19}$$

2.3.4 The sampling and periodization operators

The process of sampling is well-defined on $S_0(\mathbb{R})$:

Definition 2.3.6. Let $\alpha > 0$ and $a \in \mathbb{N}$. The sampling operators are given by

$$\begin{aligned}
\mathcal{S}_\alpha : S_0(\mathbb{R}) \to l^1(\mathbb{Z}) &: (\mathcal{S}_\alpha f)(j) = \sqrt{\alpha} f(j\alpha), \quad \forall j \in \mathbb{Z} && (2.20)\\
\mathcal{S}_a : l^1(\mathbb{Z}) \to l^1(\mathbb{Z}) &: (\mathcal{S}_a f)(j) = \sqrt{a} f(ja), \quad \forall j \in \mathbb{Z} && (2.21)\\
\mathcal{S}_\alpha : C([0; \alpha L]) \to \mathbb{C}^L &: (\mathcal{S}_\alpha f)(j) = \sqrt{\alpha} f(j\alpha), \quad j = 0, ..., L-1 && (2.22)\\
\mathcal{S}_a : \mathbb{C}^{aL} \to \mathbb{C}^L &: (\mathcal{S}_a f)(j) = \sqrt{a} f(ja), \quad j = 0, ..., L-1 && (2.23)
\end{aligned}$$

The fact that the sampling operator is a bounded operator from $S_0(\mathbb{R})$ into $l^1(\mathbb{Z})$ is proved in [37, Lemma 3.2.11]. The factor $\sqrt{\alpha}$ appearing in the definition of the sampling operator gives the sampling operators the following important properties:

- Composition with dilations:

$$\begin{aligned}
\mathcal{S}_a D_b &= \mathcal{S}_{ab} \text{ on } S_0(\mathbb{R}) && (2.24)\\
\mathcal{S}_a D_b &= \mathcal{S}_{ab} \text{ on } C([0, L]) && (2.25)
\end{aligned}$$

- If $f \in S_0(\mathbb{R})$ then $\|S_\alpha f\| < C \|f\|_{S_0}$ for some $C > 0$ independent of α. This is proved in [41, Prop. 11.1.4].

Definition 2.3.7. The periodization operators are given by

$$\mathcal{P}_L : S_0(\mathbb{R}) \to \mathcal{A}([0,L]) \quad : \quad \mathcal{P}_L g(x) = \sum_{k\in\mathbb{Z}} g(x+kL), \quad x \in [0,L] \tag{2.26}$$

$$\mathcal{P}_L : l^1(\mathbb{Z}) \to \mathbb{C}^L \quad : \quad \mathcal{P}_L g(j) = \sum_{k\in\mathbb{Z}} g(j+kL), \quad j = 0,...,L-1 \tag{2.27}$$

$$\mathcal{P}_M : \mathcal{A}([0,ML]) \to \mathcal{A}([0,L]) \quad : \quad \mathcal{P}_M g(x) = \sum_{k\in\mathbb{Z}} g(x+kM), \quad x \in [0,L] \tag{2.28}$$

$$\mathcal{P}_M : \mathbb{C}^{ML} \to \mathbb{C}^L \quad : \quad \mathcal{P}_M g(j) = \sum_{k\in\mathbb{Z}} g(j+kM), \quad j = 0,...,L-1 \tag{2.29}$$

The fact that $\mathcal{P}_L : S_0(\mathbb{R}) \to \mathcal{A}([0,L])$ is proved in [33] in the more general context of LCA-groups.

2.3.5 Frames for Hilbert spaces

Since we will deal with Gabor frames for four different spaces, it will be beneficial to note what can be said about frames for general Hilbert spaces.

Definition 2.3.8. A family of elements $\{e_j\}_{j\in J}$ in a separable Hilbert space $\mathcal{H}$ is called a frame if constants $0 < A \leq B < \infty$ exist such that

$$A \|f\|_{\mathcal{H}}^2 \leq \sum_{j\in J} \left|\langle f, e_j\rangle_{\mathcal{H}}\right|^2 \leq B \|f\|_{\mathcal{H}}^2, \quad \forall f \in \mathcal{H}. \tag{2.30}$$

The constants A and B are called lower and upper frame bounds, respectively.

Two frames for a Hilbert space $\mathcal{H}$, $\{e_j\}$ and $\{f_j\}$, are called *dual frames* if and only if $f = \sum_{j\in J} \langle f, e_j\rangle f_j, \quad \forall f \in \mathcal{H}$.

The *frame operator* of a frame $\{e_j\}_{j\in J}$ for a Hilbert space $\mathcal{H}$ is defined by

$$S : \mathcal{H} \to \mathcal{H} \quad : \quad Sf = \sum_j \langle f, e_j\rangle_{\mathcal{H}} e_j, \tag{2.31}$$

where the series defining Sf converges unconditionally for all $f \in \mathcal{H}$.

The condition (4.9) ensures that the frame operator is both bounded and invertible on $\mathcal{H}$.

The inverse frame operator can be used to give a decomposition of any function $f \in \mathcal{H}$:

$$f = \sum_j \left\langle f, S^{-1}e_j\right\rangle e_j, \quad \forall f \in \mathcal{H}. \tag{2.32}$$

The frame $\{S^{-1}e_j\}$ is known as the canonical dual frame.

2.3.6 Gabor systems

Gabor systems can be defined for the four spaces using the corresponding translation and modulation operators.

Definition 2.3.9. Gabor systems for each of the four spaces:

$L^2(\mathbb{R})$ A Gabor system (g, α, β) for $L^2(\mathbb{R})$ is defined as

$$(g, \alpha, \beta) = \{M_{m\beta} T_{n\alpha} g\}_{m,n \in \mathbb{Z}}, \tag{2.33}$$

where $g \in L^2(\mathbb{R})$ and $\alpha, \beta > 0$.

$l^2(\mathbb{Z})$ A Gabor system $\left(g, a, \frac{1}{M}\right)$ for $l^2(\mathbb{Z})$ is defined as

$$\left(g, a, \frac{1}{M}\right) = \left\{M_{m/M} T_{na} g\right\}_{m=0,\ldots,M-1, n \in \mathbb{Z}}, \tag{2.34}$$

where $g \in l^2(\mathbb{Z})$ and $a, M \in \mathbb{N}$.

$L^2([0, L])$ A Gabor system (g, a, b) for $L^2([0, L])$ is defined as

$$(g, a, b) = \{M_{mb} T_{na} g\}_{m \in \mathbb{Z}, n=0,\ldots,N-1}, \tag{2.35}$$

where $g \in L^2([0, L])$, $a, b, N \in \mathbb{N}$ and $L = Na$.

$\mathbb{C}^L$ A Gabor system for (g, a, b) for $\mathbb{C}^L$ is defined as

$$(g, a, b) = \{M_{mb} T_{na} g\}_{m=0,\ldots,M-1, n=0,\ldots,N-1}, \tag{2.36}$$

where $g \in \mathbb{C}^L$, $a, b, M, N \in \mathbb{N}$ and $Mb = Na = L$.

A *Gabor frame* is a Gabor system that is also a frame. For a good introduction to Gabor systems see [35, 36] and [41].

The Gabor frame operator (and its inverse) commute with the translation and modulation operators. If this is applied to the the decomposition formula (2.32), it shows that the canonical dual frame of a Gabor frame (g, α, β) is again a Gabor frame $(\gamma^0, \alpha, \beta)$, where $\gamma^0 = S^{-1} g$ is known as the *canonical dual window*.

If a Gabor frame is not a Riesz basis, it has more than one dual frame, and some of these dual frames will even be Gabor frames.

2.4 Between the spaces

The section is a survey of existing results of how Gabor frames behave with respect to samplings and periodization. Some new results about non-canonical dual window for Gabor frames are presented as well.

We present the results for the interrelations show on Fig. 2.1. This reason for this is that the four different types of Gabor systems each has their place in applications. Gabor frames for $l^2(\mathbb{Z})$ and $L^2([0, L])$ are less common in literature, but there are several good reasons for considering them:

- The space $l^2(\mathbb{Z})$ is a good model of a possibly infinite stream of discrete data, as it occurs in filter bank theory. It is also the correct place to study decay properties of discrete Gabor windows. Results about decay properties of windows for Gabor frames for $l^2(\mathbb{Z})$ can be transferred to the finite, discrete case by the results presented in Sec. 2.4.4.
- The space $L^2([0, L])$ is a good model of continuous phenomena with finite duration. An advantage of using $L^2([0, L])$ as opposed to $L^2(\mathbb{R})$ is that discretizing a Gabor frame is straightforward, see Sec. 2.4.3.

A generalization that will not be explicitly mentioned in the following is, that all statements about a canonical dual window also holds for the corresponding tight window. Proofs of this appear in [24].

2.4.1 From $L^2(\mathbb{R})$ to $l^2(\mathbb{Z})$.

In [53] Janssen proved, that if the window g of a Gabor frame $\left(g, a, \frac{1}{M}\right)$ for $L^2(\mathbb{R})$, $a, M \in \mathbb{N}$, satisfies the so-called *condition R* then by sampling it at the integers one obtains the window $\mathcal{S}_1 g$ of a Gabor frame $\left(\mathcal{S}_1 g, a, \frac{1}{M}\right)$ for $l^2(\mathbb{Z})$ with the same frame bounds. Furthermore he proved that if g additionally satisfies the so-called *condition A* then $\mathcal{S}_1\gamma^0$ will be the canonical dual window of $\left(\mathcal{S}_1 g, a, \frac{1}{M}\right)$.

Condition R is a regularity condition, that requires the function to have some smoothness around sampling points and decay as well. It is much weaker than requiring the function to be in $S_0(\mathbb{R})$. Most remarkably, it only places restrictions on g in a neighbourhood of the sampling points. The downside of condition R is that it depends on the position of the sampling points.

Condition A is a statement about the decay of inner products between g and certain time-frequency shifts of g. It depends on the parameters α, β, and is in general hard to verify.

All functions in $S_0(\mathbb{R})$ satisfies condition R and A for all sampling distances and for any combination of α and β. For condition R this is mentioned in [24]. For condition A this is a simple fact of Proposition 2.3.5. A direct proof is in [41, Corr. 12.1.12], along with a nice coverage of the topic.

In [62], new proofs of the combined transition $L^2(\mathbb{R}) \rightarrow \mathbb{C}^L$ is presented. These use $S_0(\mathbb{R})$ as a sufficient condition, and does not consider condition R and A.

The following result is a simple generalization of Janssen's sampling result to hold for Gabor frames with rational oversampling.

Theorem 2.4.1. *Let $g \in S_0(\mathbb{R})$, $\alpha, \beta > 0$ with $\alpha\beta = \frac{a}{M}$ for some $M, a \in \mathbb{N}$, and assume that (g, α, β) is a Gabor frame for $L^2(\mathbb{R})$ with canonical dual window γ^0. Then $\left(\mathcal{S}_{\alpha/a} g, a, \frac{1}{M}\right)$ is a Gabor frame for $l^2(\mathbb{Z})$ with the same frame bounds and canonical dual window $\mathcal{S}_{\alpha/a}\gamma^0$.*

In [79] Orr showed that the expansion coefficients $c(m, n)$ of a function f in a Gabor frame for $L^2(\mathbb{R})$ with dual window γ is related to the expansion coefficients $d(m, n)$ of the sampled function $\mathcal{S}_{\alpha/a} f$ in a Gabor frame for $l^2(\mathbb{Z})$ with the sampled dual window $\mathcal{S}_{\alpha/a}\gamma$. In [79] the result is stated without explicit conditions on the involved functions and only in the case of critical sampling (when $\alpha\beta = 1$). In [62] Kaiblinger has shown that $S_0(\mathbb{R})$ is a sufficient condition, and that the statements also holds for oversampled Gabor frames. The result [62, Lem. 9] covers the complete transition $L^2(\mathbb{R}) \to \mathbb{C}^L$. The proof of the following statement can be found by modifying this.

Proposition 2.4.2. *Let $f, \gamma \in S_0(\mathbb{R})$, $\alpha, \beta > 0$ with $\alpha\beta = \frac{a}{M}$ for some $M, a \in \mathbb{N}$. With*

$$\begin{aligned} c(m,n) &= \langle f, M_{m\beta} T_{n\alpha} \gamma \rangle_{L^2(\mathbb{R})}, \quad m,n \in \mathbb{Z}, & (2.37)\\ d(m,n) &= \left\langle \mathcal{S}_{\alpha/a} f, M_{m/M} T_{na} \mathcal{S}_{\alpha/a} \gamma \right\rangle_{l^2(\mathbb{Z})}, \quad m = 0, ..., M-1, n \in \mathbb{Z} & (2.38)\end{aligned}$$

then

$$d(m,n) = \sum_{j \in \mathbb{Z}} c(m - jM, n), \quad \forall m = 0, ..., M-1, n \in \mathbb{Z} \tag{2.39}$$

The sum in the previous proposition is well defined because of Prop. 2.3.5.

Theorem 2.4.1 shows that if γ is the canonical dual window of g, $\mathcal{S}_{\alpha/a}\gamma$ is the canonical dual window of $\mathcal{S}_{\alpha/a} g$. This relation holds not only for the canonical dual window, but in fact for ALL dual windows $\gamma \in S_0(\mathbb{R})$.

Proposition 2.4.3. *Let (g, α, β), $g \in S_0(\mathbb{R})$, $\alpha, \beta > 0$ with $\alpha\beta = \frac{a}{M}$ for some $M, a \in \mathbb{N}$, be a Gabor frame for $L^2(\mathbb{R})$ and let $\gamma \in S_0(\mathbb{R})$ be a dual window. Then $S_{\alpha/a}\gamma$ is a dual window of $\left(\mathcal{S}_{\alpha/a} g, a, \frac{1}{M}\right)$.*

In [52] it is shown that if g satisfies Condition A then the canonical dual window γ^0 of (g, α, β) also satisfies Condition A. In [31, 43] it is shown that if $g \in S_0(\mathbb{R})$ then also $\gamma^0 \in S_0(\mathbb{R})$. Because of these results, the only assumption needed in Theorem 2.4.1 is $g \in S_0(\mathbb{R})$. In Proposition 2.4.3 we need to impose $\gamma^0 \in S_0(\mathbb{R})$ as a separate condition.

2.4.2 From $L^2(\mathbb{R})$ to $L^2([0, L])$

The results from the previous section can be easily extended to prove how a Gabor frame for $L^2([0, L])$ can be obtained from a Gabor frame for $L^2(\mathbb{R})$ by periodizing the window function.

Gabor frames for $L^2([0, L])$ are not as widely studied as Gabor frames for $L^2(\mathbb{R})$. From a practical point of view, they are interesting because they can be used to model continuous phenomena of finite duration.

Interrelations between Wilson bases for $L^2(\mathbb{R})$ and $L^2([0, L])$ are described in [10, Corollary 9.3.6].

Theorem 2.4.4. *Let $g \in S_0(\mathbb{R})$, $\alpha, \beta > 0$ with $\alpha\beta = \frac{b}{N}$ for some $N, b \in \mathbb{N}$ and assume that (g, α, β) is Gabor frame for $L^2(\mathbb{R})$ with canonical dual window γ^0. Then $\left(\mathcal{P}_{b/\beta} g, \alpha, b\right)$ is a Gabor frame for $L^2([0, \frac{b}{\beta}])$ with the same frame bounds and canonical dual window $\mathcal{P}_{b/\beta}\gamma^0$.*

The proof of the following can by modifying [62, Lem. 9].

Proposition 2.4.5. *Let $f, \gamma \in S_0(\mathbb{R})$, $\alpha, \beta > 0$ with $\alpha\beta = \frac{b}{N}$ for some $N, b \in \mathbb{N}$. With*

$$\begin{aligned} c(m,n) &= \langle f, M_{m\beta} T_{n\alpha} \gamma \rangle_{L^2(\mathbb{R})}, \quad m,n \in \mathbb{Z} & (2.40)\\ d(m,n) &= \left\langle \mathcal{P}_{b/\beta} f, M_{mb} T_{na} \mathcal{P}_{b/\beta} \gamma \right\rangle_{L^2([0,\frac{b}{\beta}])}, \quad m \in \mathbb{Z}, n = 0, ..., N-1 & (2.41)\end{aligned}$$

then

$$d(m,n) = \sum_{j \in \mathbb{Z}} c(m, n - jN), \quad \forall m \in \mathbb{Z}, n = 0, ..., N-1. \tag{2.42}$$

The sum in the previous proposition is well defined because of Prop. 2.3.5.

Proposition 2.4.6. *Let (g, α, β) with $g \in S_0(\mathbb{R})$, $\alpha, \beta > 0$ and $\alpha\beta = \frac{b}{N}$ for some $N, b \in \mathbb{N}$ be a Gabor frame for $L^2(\mathbb{R})$ and let $\gamma \in S_0(\mathbb{R})$ be a dual window. Then $\mathcal{P}_{b/\beta}\gamma$ is a dual window of $(\mathcal{P}_{b/\beta}g, \alpha, b)$.*

The proof has the same structure as that of Proposition 2.4.3, using Proposition 2.4.5 as the main ingredient.

2.4.3 From $L^2([0, L])$ to $\mathbb{C}^L$

A Gabor frame for $\mathbb{C}^L$ can be obtained by sampling the window function of a Gabor frame for $L^2([0, L])$. The proofs are very similar to the proofs presented in Section 2.4.1 for the $L^2(\mathbb{R}) \to l^2(\mathbb{Z})$ case.

Theorem 2.4.7. *Let $g \in \mathcal{A}([0, L_1])$, $L_2, M, N, a_2, b \in \mathbb{N}$ with $L_1 = Na_1$ and $L_2 = Mb = Na_2$. Assume that (g, a_1, b) is a Gabor frame for $L^2([0, L_1])$ with canonical dual window γ^0. Then $(\mathcal{S}_{L_1/L_2}g, a_2, b)$ is a Gabor frame for $\mathbb{C}^{L_2}$ with the same frame bounds and canonical dual window $\mathcal{S}_{L_1/L_2}\gamma^0$.*

The proof can be found by carefully modifying the proof in [53] for the $L^2(\mathbb{R})$-case. The modifications consist in replacing integration over $\mathbb{R}$ with integrations over $[0, L]$ and similar changes.

The proof of the following can by modifying [62, Lem. 9].

Proposition 2.4.8. *Let $f, \gamma \in \mathcal{A}([0, L_1])$, $L_2, M, N, a_2, b \in \mathbb{N}$ with $L_1 = Na_1$ and $L_2 = Mb = Na_2$. With*

$$\begin{aligned} c(m,n) &= \langle f, M_{mb}T_{na_1}\gamma \rangle_{L^2([0,L_1])}, \quad m \in \mathbb{Z}, n = 0, ..., N-1 & (2.43) \\ d(m,n) &= \langle \mathcal{S}_{L_1/L_2}f, M_{mb}T_{na_2}\mathcal{S}_{L_1/L_2}\gamma \rangle_{\mathbb{C}^L}, \ m = 0, ..., M-1, n = 0, ..., N-1 & (2.44) \end{aligned}$$

then

$$d(m,n) = \sum_{j \in \mathbb{Z}} c(m - jM, n), \quad \forall m = 0, ..., M-1, n = 0, ..., N-1 \tag{2.45}$$

The sum in the previous proposition is well defined because of Prop. 2.3.5.

Proposition 2.4.9. *Let (g, a_1, b) $g \in \mathcal{A}([0, L_1])$, $L_2, M, N, a_2, b \in \mathbb{N}$ with $L_1 = Na_1$ and $L_2 = Mb = Na_2$, be a Gabor frame for $L^2([0, L_1])$ and let $\gamma \in \mathcal{A}([0, L_1])$ be a dual window. Then $\mathcal{S}_{L_1/L_2}\gamma$ is a dual window of $(\mathcal{S}_{L_1/L_2}g, a_2, b)$.*

The proof has the same structure as that of Proposition 2.4.3, using Proposition 2.4.8 as the main ingredient.

2.4.4 From $l^2(\mathbb{Z})$ to $\mathbb{C}^L$

Similarly to the results presented in the previous sections, one can obtain a Gabor frame for $\mathbb{C}^L$ by periodizing the window function of a Gabor frame for $l^2(\mathbb{Z})$.

The following result appears in[53].

Theorem 2.4.10. *Let $g \in l^1(\mathbb{Z})$, $M, N, a, b \in \mathbb{N}$ with $Mb = Na = L$ and assume that $\left(g, a, \frac{1}{M}\right)$ is a Gabor frame for $l^2(\mathbb{Z})$ with canonical dual window γ^0. Then $(\mathcal{P}_L g, a, b)$ is a Gabor frame for $\mathbb{C}^L$ with the same frame bounds and canonical dual window $\mathcal{P}_L \gamma^0$.*

The proof of the following can by modifying [62, Lem. 9].

Proposition 2.4.11. *Let $f, \gamma \in l^1(\mathbb{Z})$, $M, N, a, b \in \mathbb{N}$ with $Mb = Na = L$. With*

$$\begin{aligned} c(m,n) &= \langle f, M_{mb} T_{na} \gamma \rangle_{l^2(\mathbb{Z})}, \quad m = 0, ..., M-1, n \in \mathbb{Z} && (2.46) \\ d(m,n) &= \langle \mathcal{P}_L f, M_{mb} T_{na} \mathcal{P}_L \gamma \rangle_{\mathbb{C}^L}, \; m = 0, ..., M-1, n = 0, ..., N-1, && (2.47) \end{aligned}$$

then

$$d(m,n) = \sum_{j \in \mathbb{Z}} c(m, n - jN), \quad \forall m = 0, ..., M-1, n = 0, ..., N-1 \qquad (2.48)$$

The sum in the previous proposition is well defined because of Prop. 2.3.5.

Proposition 2.4.12. *Let $\left(g, a, \frac{1}{M}\right)$ $g \in l^1(\mathbb{Z})$, $M, N, a, b \in \mathbb{N}$ with $Mb = Na = L$, be a Gabor frame for $l^2(\mathbb{Z})$ and let $\gamma \in l^1(\mathbb{Z})$ be a dual window. Then $\mathcal{P}_L \gamma$ is a dual window of $(\mathcal{P}_L g, a, b)$.*

The proof has the same structure as that of Proposition 2.4.3, using Proposition 2.4.11 as the main ingredient.

2.4.5 From $L^2(\mathbb{R})$ to $\mathbb{C}^L$

With the results from the previous sections, any Gabor frame for $L^2(\mathbb{R})$, (g, α, β) with $g \in S_0(\mathbb{R})$ and rational sampling, $\alpha\beta \in \mathbb{Q}$, can be sampled and periodized to obtain a Gabor frame for $\mathbb{C}^L$ with corresponding sampled and periodized canonical dual window. Direct proofs of the two first statements are given in [62]. The proofs of the three following statements can also found by simply combining the similar from the previous sections. Going through $L^2([0, \frac{b}{\beta}])$ or $l^2(\mathbb{Z})$ produces the same result, because $\mathcal{P}_L \mathcal{S}_{\alpha/a} = \mathcal{S}_{b/\beta/L} \mathcal{P}_{b/\beta}$ on $S_0(\mathbb{R})$. This shows that the two directions on Fig. 2.1 commute.

Theorem 2.4.13. *Let $g \in S_0(\mathbb{R})$, $\alpha\beta = \frac{a}{M} = \frac{b}{N}$ and $Mb = Na = L$ with $a, b, M, N, L \in \mathbb{N}$ and assume that (g, α, β) is a Gabor frame for $L^2(\mathbb{R})$ with canonical dual window γ^0. Then $\left(\mathcal{P}_L \mathcal{S}_{\alpha/a} g, a, b\right)$ is a Gabor frame for $\mathbb{C}^L$ with the same frame bounds and canonical dual window $\mathcal{P}_L \mathcal{S}_{\alpha/a} \gamma^0$.*

Fully written out then $\mathcal{P}_L \mathcal{S}_{\alpha/a} g$ is

$$\left(\mathcal{P}_L \mathcal{S}_{\alpha/a}\right) g(j) = \sqrt{\frac{\alpha}{a}} \sum_{k \in \mathbb{Z}} g\left(\frac{\alpha}{a}(j - kL)\right), \quad j = 0, ..., L-1. \qquad (2.49)$$

Proposition 2.4.14. *Let $f, \gamma \in S_0(\mathbb{R})$, $\alpha, \beta > 0$ with $\alpha\beta = \frac{a}{M} = \frac{b}{N}$, $L = Mb = Na$ with $a, b, M, N, L \in \mathbb{N}$. With*

$$\begin{aligned} c(m,n) &= \langle f, M_{m\beta}T_{n\alpha}\gamma\rangle_{L^2(\mathbb{R})}, \quad m,n \in \mathbb{Z} && (2.50)\\ d(m,n) &= \left\langle \mathcal{P}_L\mathcal{S}_{\alpha/a}f, M_{mb}T_{na}\mathcal{P}_L\mathcal{S}_{\alpha/a}\gamma\right\rangle_{\mathbb{C}^L}, \; m = 0,...,M-1, n = 0,...,N-1 && (2.51)\end{aligned}$$

then

$$d(m,n) = \sum_{j,k\in\mathbb{Z}} c(m-kM, n-jN), \quad \forall = 0,...,M-1, n = 0,...,N-1. \tag{2.52}$$

The sum in the previous proposition is well defined because of Prop. 2.3.5.

Proposition 2.4.15. *Let $(g, \alpha\beta)$, $\alpha\beta = \frac{a}{M} = \frac{b}{N}$ and $Mb = Na = L$ with $a, b, M, N, L \in \mathbb{N}$, be a Gabor frame for $L^2(\mathbb{R})$ and let $\gamma \in S_0(\mathbb{R})$ be a dual window. Then $\mathcal{P}_L\mathcal{S}_{\alpha/a}\gamma$ is a dual window of $\left(\mathcal{P}_L\mathcal{S}_{\alpha/a}g, a, b\right)$.*

2.5 Going back

Theorem Theorem 2.4.13 provides an easy way to compute samples of the canonical dual window γ^0 of a Gabor frame (g, α, β) for $L^2(\mathbb{R})$ using finite-dimensional methods. The question is whether it is possible to use this to construct a sequence of functions converging to γ^0 in some norm.

Kaiblinger showed in [62] that this is indeed possible. By using a standard interpolation scheme to interpolate the computed samples, one can construct functions that converges to γ^0 in the $S_0(\mathbb{R})$-norm. This implies convergence for all L^p-spaces, $1 \le p \le \infty$.

Some other methods of approximating the inverse frame operator of Gabor frame is presented in [22, 24].

The following method works the same way as the method proposed by Kaiblinger, but it uses another interpolation scheme. We will use the Gabor atoms of the Gabor frame (g, α, β).

The result is presented only for M, N being even. The case of odd M, N is similar.

Theorem 2.5.1. *Let $g \in S_0(\mathbb{R})$, $\alpha, \beta > 0$ with $\alpha\beta = \frac{a}{M}$ and assume that (g, α, β) is a Gabor frame for $L^2(\mathbb{R})$ with canonical dual window γ^0.*

For each even $M, N \in \mathbb{N}$ such that $L = Mb = Na$ with $a, b, L \in \mathbb{N}$ denote the canonical dual window of $\left(\mathcal{P}_L\mathcal{S}_{\alpha/a}g, a, b\right)$ by $\varphi_{M,N}$. Define $d_{M,N} \in \mathbb{C}^{M\times N}$ by $d_{M,N}(m,n) = \langle\varphi_{M,N}, M_{mb}T_{na}\varphi_{M,N}\rangle_{\mathbb{C}^L}$ and $\gamma^0_{M,N} \in S_0(\mathbb{R})$ by

$$\gamma^0_{M,N} = \sum_{n=-N/2}^{N/2-1}\sum_{m=-M/2}^{M/2-1} d_{M,N}(m,n)M_{m\beta}T_{n\alpha}g. \tag{2.53}$$

Then $\left\|\gamma^0 - \gamma^0_{M,N}\right\|_{S_0} \to 0$ as $M, N \to \infty$.

Define $c \in l^1(\mathbb{Z}\times\mathbb{Z})$ by $c(m,n) = \langle\gamma^0, M_{m\beta}T_{n\alpha}\gamma^0\rangle_{L^2(\mathbb{R})}$. The crucial fact that $c \in l^1(\mathbb{Z}\times\mathbb{Z})$ follows from Prop. 2.3.5. Then

$$\gamma^0 = \sum_{m,n\in\mathbb{Z}} c(m,n)M_{m\beta}T_{n\alpha}g. \tag{2.54}$$

This is the standard frame expansion, since γ^0 and g are dual windows.

By (2.53) and (2.54) both γ^0 and $\gamma^0_{M,N}$ can be written using the frame (g, α, β). Subtracting them gives

$$\gamma^0 - \gamma^0_{M,N} = \sum_{m,n\in\mathbb{Z}} r_{M,N}(m,n) M_{m\beta} T_{n\alpha} g, \tag{2.55}$$

where

$$r_{M,N}(m,n) = \begin{cases} c(m,n) - d_{M,N}(m,n) & \text{if } -\frac{M}{2} \leq m \leq \frac{M}{2} - 1 \text{ and} \\ & \quad -\frac{N}{2} \leq n \leq \frac{N}{2} - 1 \\ c(m,n) & \text{otherwise} \end{cases} \tag{2.56}$$

By Proposition 2.4.14,

$$d_{M,N}(m,n) = \sum_{j,k\in\mathbb{Z}} c(m - kM, n - jN), \tag{2.57}$$

for all $-\frac{M}{2} \leq m \leq \frac{M}{2} - 1$ and $-\frac{N}{2} \leq n \leq \frac{N}{2} - 1$. This gives the following expression of the residual r:

$$\begin{aligned} r_{M,N}(m,n) &= \\ &= \begin{cases} -\sum_{j,k\in\mathbb{Z}\setminus\{0\}} c_{M,N}(m - kM, n - jN) & \text{if } -\frac{M}{2} \leq m \leq \frac{M}{2} - 1 \text{ and} \\ & \quad -\frac{N}{2} \leq n \leq \frac{N}{2} - 1 \\ c(m,n) & \text{otherwise} \end{cases} \end{aligned}$$

From 2.3.4 it can be seen that the translation and modulation operators are unitary on $S_0(\mathbb{R})$. Using this, we get the following estimate.

$$\begin{aligned} \left\|\gamma^0 - \gamma^0_{M,N}\right\|_{S_0} &= \left\| \sum_{m,n\in\mathbb{Z}} r_{M,N}(m,n) M_{m\beta} T_{n\alpha} g \right\|_{S_0} & (2.58) \\ &\leq \|g\|_{S_0} \sum_{m,n\in\mathbb{Z}} |r_{M,N}(m,n)| & (2.59) \\ &\leq \|g\|_{S_0} \sum_{n=-N/2}^{N/2-1} \sum_{m=-M/2}^{M/2-1} \sum_{j,k\in\mathbb{Z}\setminus\{0\}} |c_{M,N}(m - kM, n - jN)| + & (2.60) \\ &\quad + \|g\|_{S_0} \sum_{m\notin\left\{-\frac{M}{2},\dots,\frac{M}{2}-1\right\}} \sum_{n\notin\left\{-\frac{N}{2},\dots,\frac{N}{2}-1\right\}} |c(m,n)| & (2.61) \end{aligned}$$

In the last term, only the coefficients outside the rectangle indexed by $m \in \left\{-\frac{M}{2}, \dots, \frac{M}{2} - 1\right\}$ and $n \in \left\{-\frac{N}{2}, \dots, \frac{N}{2} - 1\right\}$ appears, and they all appear twice. From this

$$\left\|\gamma^0 - \gamma^0_{M,N}\right\|_{S_0} \leq 2\|g\| \sum_{m\notin\left\{-\frac{M}{2},\dots,\frac{M}{2}-1\right\}} \sum_{n\notin\left\{-\frac{N}{2},\dots,\frac{N}{2}-1\right\}} |c(m,n)| \tag{2.62}$$

When $M, N \to \infty$, the last term goes to zero.

Remark 2.5.2. Since each $\gamma^0_{M,N}$ is a finite linear combination of Gabor atoms from (g,α,β), they inherit properties from g: Since $g \in S_0(\mathbb{R})$ then each $\gamma^0_{M,N} \in S_0(\mathbb{R})$. Similarly, if g or $\hat{g}$ has exponential decay, then so does $\gamma^0_{M,N}$ or $\hat{\gamma}^0_{M,N}$. This is a main difference to the method of Kaiblinger [62], where the smoothness properties of the constructed approximation depend on the interpolation method.

Remark 2.5.3. If additionally $g \in \mathcal{S}(\mathbb{R})$, where $\mathcal{S}(\mathbb{R})$ is the Schwartz-space of rapidly decaying, smooth functions, then also $\gamma^0 \in \mathcal{S}(\mathbb{R})$, [51]. Is this case then by Remark 2.5.2 each $\gamma^0_{M,N} \in \mathcal{S}(\mathbb{R})$, so the convergence of $\gamma^0_{M,N}$ to γ^0 is purely within $\mathcal{S}(\mathbb{R})$.

To use this method, the two main numerical calculations to be carried out are the inversion of the frame operator for $(g_{M,N},a,b)$ and the calculation of the coefficients $d_{M,N} \in \mathbb{C}^{M\times N}$. Algorithms based on FFTs and matrix-factorisations can be found in [96]. These calculations can be performed in $\mathcal{O}(Lq) + \mathcal{O}(NM\log M)$ time, where $\frac{q}{p}$ is the oversampling factor written as an irreducible fraction.

2.6 Proofs

2.6.1 Some additional theory.

The dilation operators are unitary operators for all $d > 0$, and the following commutation relations between the translation and modulation operators holds:

$$D_dM_\omega T_t \;=\; M_{d\omega}T_{\frac{t}{d}}D_d \text{ on } L^2(\mathbb{R}) \tag{2.63}$$

$$D_dM_kT_t \;=\; M_kT_{\frac{t}{d}}D_d \text{ on } L^2([0,L]) \tag{2.64}$$

Notice that the parameter of the modulation operator is unchanged in (2.64), as opposed to (2.63). This is caused by the definition of the modulation operator.

For the various Fourier transforms and the modulation and translation operators the following commutation relations hold:

$$\mathcal{F}_{\mathbb{R}}M_\omega T_t \;=\; e^{2\pi it\omega}M_{-t}T_\omega\mathcal{F}_{\mathbb{R}} \text{ on } L^2(\mathbb{R}) \tag{2.65}$$

$$\mathcal{F}_{\mathbb{Z}}M_\omega T_j \;=\; e^{2\pi it\omega}M_{-j}T_\omega\mathcal{F}_{\mathbb{Z}} \text{ on } l^2(\mathbb{Z}) \tag{2.66}$$

$$\mathcal{F}_{[0,L]}M_kT_t \;=\; e^{2\pi itk/L}M_{-t/L}T_k\mathcal{F}_{[0,L]} \text{ on } L^2([0,L]) \tag{2.67}$$

$$\mathcal{F}_LM_kT_j \;=\; e^{2\pi ijk/L}M_{-j}T_k\mathcal{F}_L \text{ on } \mathbb{C}^L \tag{2.68}$$

We shall need the Poisson summation formula on $L^2(\mathbb{R})$ stated as a relation between Fourier transformation and the sampling- and periodization operators. A proof can be found in [41, p. 105].

Theorem 2.6.1. *The Poisson summation formula:*

$$\mathcal{P}_M \;=\; \mathcal{F}^{-1}_{[0,M]}\mathcal{S}_{1/M}\mathcal{F}_{\mathbb{R}} \text{ on } S_0(\mathbb{R}) \tag{2.69}$$

The following simple lemma [23, Lemma 5.3.3] shall be used frequently.

Lemma 2.6.2. *Let T be a unitary operator on $\mathcal{H}$, and assume that $\{e_j\}$ is a frame for $\mathcal{H}$. Then $\{Te_j\}$ is also a frame for $\mathcal{H}$ with the same frame bounds.*

The dual frames that have Gabor structure are characterized by the Wexler-Raz relations:

Theorem 2.6.3 (Wexler-Raz). *If g and γ both generates Gabor systems as in Definition 2.3.9 with finite upper frame bounds, then they are dual windows if and only if*

$L^2(\mathbb{R})$	$\frac{1}{\alpha\beta}\left\langle \gamma, M_{m/\alpha}T_{n/\beta}g\right\rangle = \delta_m\delta_n, \quad m,n\in\mathbb{Z}$
$l^2(\mathbb{Z})$	$\frac{M}{a}\left\langle \gamma, M_{m/a}T_{nM}g\right\rangle = \delta_m\delta_n, \quad m=0,...,a-1, n\in\mathbb{Z}$
$L^2([0,L])$	$\frac{N}{b}\left\langle \gamma, M_{mN}T_{nM/L}g\right\rangle = \delta_m\delta_n, \quad m\in\mathbb{Z}, n=0,...,b-1$
$\mathbb{C}^L$	$\frac{MN}{L}\left\langle \gamma, M_{mN}T_{nM}g\right\rangle = \delta_m\delta_n, \quad m=0,...,a-1, n=0,...,b-1$

Proof of the original result for $L^2(\mathbb{R})$ and $\mathbb{C}^L$ can be found in [107]. More rigorous proofs with a minimal sufficient condition appear in [51, 26] and equally in [50]. Also see [54, Subsecs. 1.4.2 and 1.6.4].

Combining Lemma 2.6.2 with the relations (2.63), (2.64) and (2.65)-(2.68) yields the following well known results. The first relation appears as [41, 6.36].

Lemma 2.6.4. *Dual Gabor frames under Fourier transforms and dilations.*

$\mathcal{F}_{\mathbb{R}}$: Let γ^0 be the canonical dual window of (g,α,β), $g\in L^2(\mathbb{R})$. The canonical dual window of $(\mathcal{F}_{\mathbb{R}}g,\beta,\alpha)$ is $\mathcal{F}_{\mathbb{R}}\gamma^0$.

$\mathcal{F}_{\mathbb{Z}}$: Let γ^0 be the canonical dual window of $\left(g,a,\frac{1}{M}\right)$, $g\in l^2(\mathbb{Z})$. The canonical dual window of $\left(\mathcal{F}_{\mathbb{Z}}g,\frac{1}{M},a\right)$ is $\mathcal{F}_{\mathbb{Z}}\gamma^0$.

$\mathcal{F}_{[0,L]}$: Let γ^0 be the canonical dual window of (g,a,b), $g\in L^2([0,L])$. The canonical dual window of $\left(\mathcal{F}_{[0,L]}g,b,\frac{a}{L}\right)$ is $\mathcal{F}_{[0,L]}\gamma^0$.

$\mathcal{F}_L$: Let γ^0 be the canonical dual window of (g,a,b), $g\in\mathbb{C}^L$. The canonical dual window of $(\mathcal{F}_Lg,b,a)$ is $\mathcal{F}_L\gamma^0$.

D_d on $L^2(\mathbb{R})$: Let γ^0 be the canonical dual window of (g,α,β), $g\in L^2(\mathbb{R})$. The canonical dual window of $\left(D_dg,\frac{\alpha}{d},\beta d\right)$ is $D_d\gamma^0$.

D_d on $L^2([0,L])$: Let γ^0 be the canonical dual window of (g,a,b), $g\in L^2([0,L])$. The canonical window of $\left(D_dg,\frac{a}{d},b\right)$ is $D_d\gamma^0$.

The proofs can be found by direct calculation. They are very simple, and almost identical. The only difference is which of the commutation relations (2.63), (2.64) and (2.65)-(2.68) to use.

Proof of Proposition 2.4.3. Define $c\in l^2(\mathbb{Z}\times\mathbb{Z})$ and $d\in l^2(\{0,...,a-1\}\times\mathbb{Z})$ by

$$\begin{aligned} c(m,n) &= \left\langle \gamma, M_{m/\alpha}T_{n/\beta}g\right\rangle_{L^2(\mathbb{R})}, \quad m,n\in\mathbb{Z}, && (2.70)\\ d(m,n) &= \left\langle \mathcal{S}_{\alpha/a}\gamma, M_{m/a}T_{nM}\mathcal{S}_{\alpha/a}g\right\rangle_{l^2(\mathbb{Z})} && (2.71)\\ &= \left\langle \mathcal{S}_{1/M\beta}\gamma, M_{m/a}T_{nM}\mathcal{S}_{1/M\beta}g\right\rangle_{l^2(\mathbb{Z})}, \quad m=0,...,M-1, n\in\mathbb{Z} && (2.72)\end{aligned}$$

It is well known, that if $g, \gamma \in S_0(\mathbb{R})$ then (g, α, β) and (γ, α, β) have finite upper frame bounds, see e.g. [41, Chapter 6]. Since g, γ are dual windows, they satisfy the Wexler-Raz condition for $L^2(\mathbb{R})$, Theorem 2.6.3:

$$\frac{1}{\alpha\beta} c(m,n) = \delta_m \delta_n, \quad m, n \in \mathbb{Z}. \tag{2.73}$$

We wish to show that $\mathcal{S}_{\alpha/a} g$ and $\mathcal{S}_{\alpha/a}\gamma$ satisfies the Wexler-Raz condition for $l^2(\mathbb{Z})$:

$$\begin{aligned} & \frac{M}{a} \left\langle \mathcal{S}_{1/M\beta}\gamma, M_{m/a} T_{nM} \mathcal{S}_{1/M\beta} g \right\rangle_{l^2(\mathbb{Z})} && (2.74) \\ = & \frac{M}{a} d(m,n) && (2.75) \\ = & \delta_m \delta_n, \quad m = 0, ..., a-1, n \in \mathbb{Z} && (2.76) \end{aligned}$$

By Theorem 2.4.1, $\left(\mathcal{S}_{\alpha/a} g, a, \frac{1}{M}\right)$ and $\left(\mathcal{S}_{\alpha/a}\gamma, a, \frac{1}{M}\right)$ also has finite upper frame bounds. By Proposition 2.4.2:

$$\begin{aligned} \frac{M}{a} d(m,n) &= \sum_{j \in \mathbb{Z}} \frac{1}{\alpha\beta} c(m - ja, n) && (2.77) \\ &= \sum_{j \in \mathbb{Z}} \delta_{m-ja} \delta_n && (2.78) \\ &= \delta_m \delta_n, \quad \forall m = 0, ..., a-1, n \in \mathbb{Z}. && (2.79) \end{aligned}$$

This shows that $\mathcal{S}_{\alpha/a} g$ and $\mathcal{S}_{\alpha/a}\gamma$ satisfies the Wexler-Raz condition for $l^2(\mathbb{Z})$, and therefore they are dual windows. □

2.6.2 Proofs of section 2.4.2

Proof of Theorem 2.4.4. By Lemma 2.6.2 and Lemma 2.6.4, $(\mathcal{F}g, \beta, \alpha)$ is also a Gabor frame for $L^2(\mathbb{R})$ with frame bounds A and B and canonical dual window $\mathcal{F}\gamma^0$.

Since $\hat{g} \in S_0(\mathbb{R})$, Theorem 2.4.1 can be used: $\left(\mathcal{S}_{\beta/b}\mathcal{F}g, b, \frac{1}{N}\right)$ is a Gabor frame for $l^2(\mathbb{Z})$ with frame bounds A and B and canonical dual window $\mathcal{S}_{\beta/b}\mathcal{F}\gamma^0$.

By Lemma 2.6.2 and Lemma 2.6.4, $\left(\mathcal{F}^{-1}_{[0,\frac{b}{\beta}]}\mathcal{S}_{\beta/b}\mathcal{F}g, a, b\right)$ is a Gabor frame for $L^2([0, \frac{b}{\beta}])$ with frame bounds A and B and canonical dual window $\mathcal{F}^{-1}_{[0,\frac{b}{\beta}]}\mathcal{S}_{\beta/b}\mathcal{F}\gamma^0$.

By the Poisson summation formula (2.69) then $\mathcal{F}^{-1}_{[0,\frac{b}{\beta}]}\mathcal{S}_{\beta/b}\mathcal{F} = \mathcal{P}_{b/\beta}$. From this the result follows. □

Chapter 3

Symmetric, discrete fractional splines and Gabor systems

This chapter is the paper [94], which has been submitted to International Journal of Wavelets, Multiresolution and Information Processing, 2007.

3.1 Abstract

In this paper we consider fractional splines as windows for Gabor frames. We introduce two new types of symmetric, fractional splines in addition to one found by Unser and Blu. For the finite, discrete case we present two families of splines: One is created by sampling and periodizing the continuous splines, and one is a truly finite, discrete construction. We discuss the properties of these splines and their usefulness as windows for Gabor frames and Wilson bases..

3.2 Introduction

Fractional splines are a simple generalization of regular B-splines to fractional orders. They have been developed by Unser and Blu in a series of papers [101, 12, 13] mostly in the context of wavelets and fractional Brownian motion.

Fractional splines are interesting in relation to Gabor frames, because they provide a smooth parameter family of functions ranging from the rectangular box function, which generates an orthonormal Gabor basis, to the Gaussian function, which is the function with the best time/frequency concentration. The fact that splines converge to the Gaussian as the order grows is shown in [99] along with some considerations of B-splines as Gabor windows for continuous Gabor frames. A result about the non-existence of certain Gabor frames with spline windows was reported in [42].

Fractional splines retain most of the well-known properties of regular B-splines, but they are no longer compactly supported. This makes them less useful for applications where data is continuously produced and analyzed (e.g. processing long music signals). However, in many applications (e.g. image processing) all data are available at the time of processing, and fast algorithms exists for these kinds of Gabor systems, [111, 7, 96].

A regular B-spline of order n can be defined as n convolutions of the rectangular function. Because a convolution of two functions can be computed by pointwise multiplication of their Fourier transforms, we get the classical expression for the B-spline β^n of order $n = 0, 1, 2, \ldots$ in the Fourier domain:

$$\widehat{\beta^n}(\omega) \;=\; \operatorname{sinc}(\omega)^{n+1}, \quad \omega \in \mathbb{R}. \tag{3.1}$$

This definition extends naturally to fractional orders $\alpha > -\frac{1}{2}$:

$$\widehat{\beta_+^\alpha}(\omega) = \operatorname{sinc}(\omega)^{\alpha+1}, \quad \omega \in \mathbb{R}. \tag{3.2}$$

This is a simple shift of the β_+^α spline defined in [101]. This spline is not even around $x = 0$, because is has a complex valued Fourier transform.

Symmetry is an important goal for window construction because symmetric windows are a requirement for Wilson bases [25, 15] and the modified discrete cosine transform (MDCT) [81, 82]. Both Wilson bases and the MDCT are cosine modulated filter banks closely related to Gabor frames of twice the redundancy. In [12] Blu and Unser's presents a family of symmetric, fractional splines. In this paper, we present two other definitions, yielding splines that generates tighter (better conditioned) Gabor frames than Blu and Unsers definition.

A central property of B-splines is that they form a partition of unity, PU, meaning that the sum of integer translates of a B-spline is a constant function. This property can be important i.e. in image processing, where it is advantageous to be able to represent a large, almost constant area of an image using few coefficients. In relation to local, trigonometric bases, this has been investigated in [60, 9].

In [12, 101] Unser and Blu have shown methods to generate discrete fractional splines by sampling. In this paper, we present two other methods for producing discrete and finite splines. In Section 3.5 we present a method to obtain finite, discrete fractional splines by sampling and periodizing their continuous counterparts. In Section 3.6 we present another method where we transfer the definition of the continuous splines to the finite, discrete setting. Finally, in Section 3.7 we make numerical comparisons of the different types of splines and study their usefullness as windows for Gabor / Wilson / MDCT frames and bases.

3.3 Definitions

We define the Fourier transform $\mathcal{F} : L^2(\mathbb{R}) \mapsto L^2(\mathbb{R})$ as

$$(\mathcal{F}f)(\omega) = \hat{f}(\omega) = \int_{x\in\mathbb{R}} f(x)\, e^{-2\pi i\omega x} dx, \quad \omega \in \mathbb{R}, \tag{3.3}$$

and the Discrete Fourier Transform (DFT) of $h \in \mathbb{C}^L$ as

$$(\mathcal{F}h)(k) = \hat{h}(k) = \frac{1}{\sqrt{L}} \sum_{l=0}^{L-1} h(k)\, e^{-2\pi ikl/L}, \quad k = 0, \ldots, L-1, \tag{3.4}$$

where i denotes the imaginary unit. The sinc function is given by

$$\operatorname{sinc}(x) = \begin{cases} \frac{\sin(\pi x)}{\pi x} & x \in \mathbb{R} \setminus \{0\} \\ 1 & x = 0 \end{cases}. \tag{3.5}$$

The sinc function is the Fourier transform of the box function β_+^0, which is explicitly given by

$$\beta_+^0(x) = \begin{cases} 1 & \text{if } |x| < \frac{1}{2} \\ \frac{1}{2} & \text{if } |x| = \frac{1}{2} \\ 0 & \text{if } |x| > \frac{1}{2}. \end{cases} \tag{3.6}$$

Convolution of two functions is given by

$$\begin{aligned} (f * g)(x) &= \int_{y\in\mathbb{R}} f(y)\overline{g(x-y)}dy, \quad x \in \mathbb{R},\ f, g \in L^1(\mathbb{R}) &(3.7)\\ (f * g)(l) &= \sum_{k=0}^{L-1} f(k)\overline{g(l-k)}, \quad l = 0, \dots, L-1,\ f, g \in \mathbb{C}^L. &(3.8)\end{aligned}$$

The convolution of two functions can be computed in the Fourier domain:

$$\begin{aligned} \widehat{f * g}(\omega) &= \widehat{f}(\omega)\widehat{g}(\omega), \quad \omega \in \mathbb{R}, \forall f, g \in L^1(\mathbb{R}), &(3.9)\\ \widehat{f * g}(k) &= \sqrt{L}\widehat{f}(k)\widehat{g}(k), \quad k = 0, \dots, L-1, \forall f, g \in \mathbb{C}^L. &(3.10)\end{aligned}$$

A family of elements $\{e_j\}_{j\in J}$ in a separable Hilbert space $\mathcal{H}$ is called a frame if constants $0 < A \leq B < \infty$ exist such that

$$A\|f\|_{\mathcal{H}}^2 \leq \sum_{j\in J} \left|\langle f, e_j\rangle_{\mathcal{H}}\right|^2 \leq B\|f\|_{\mathcal{H}}^2, \quad \forall f \in \mathcal{H}. \tag{3.11}$$

The constants A and B are called lower and upper frame bounds, respectively.

We define the Wiener space by

$$W(\mathbb{R}) = \left\{ f \,\middle|\, \sum_{n\in\mathbb{Z}} \operatorname*{ess\,sup}_{x\in[0,1]} |f(x+n)| < \infty \right\}. \tag{3.12}$$

The Wiener space is a subspace of $L^1(\mathbb{R})$ and $L^2(\mathbb{R})$.

We define Gabor systems for $L^2(\mathbb{R})$ and $\mathbb{C}^L$ by

Definition 3.3.1. A Gabor system (g, α, β) for $L^2(\mathbb{R})$ with $g \in L^2(\mathbb{R})$ and $\alpha, \beta > 0$ is given by

$$g_{m,n}(x) = e^{2\pi i m\beta x} g(x - na), \quad , m, n \in \mathbb{Z}. \tag{3.13}$$

Definition 3.3.2. A Gabor system $\left(g^D, a, b\right)$ for $\mathbb{C}^L$ with $g \in \mathbb{C}^L$, $a, b \in \mathbb{N}$ is given by

$$g_{m,n}^D(l) = e^{2\pi i m b l/L} g(l - na), \quad , m = 0, \dots, \frac{L}{b} - 1, n = 0, \dots, \frac{L}{a} - 1. \tag{3.14}$$

For more information about Gabor systems and frames, see the books [41, 23, 35, 36].

We consider the following symmetries of discrete signals:

Definition 3.3.3. Let $g \in \mathbb{C}^L$. We say that g is *whole point even* (WPE) if

$$g(l) = \overline{g(-l)} = \overline{g(L-l)}$$

for $l = 0, \dots, L-1$. This implies that $g(0)$ must always be real, and so must $g\left(\frac{L}{2}+1\right)$ if L is even.

Definition 3.3.4. Let $g \in \mathbb{C}^L$. We say that g is *half point even* (HPE) if

$$g(l) = \overline{g(L-1-l)}$$

for $l = 0, \ldots, L-1$. This implies that $g\left(\frac{L-1}{2}\right)$ must be real if L is odd.

If g is HPE then

$$\hat{g}(k) = h(k)\, e^{\pi i k/L}, \tag{3.15}$$

where h is some real-valued signal: $h \in \mathbb{R}^L$.

It is relevant to consider these two symmetries, because WPE and HPE windows are a requirement for discrete Wilson bases [15] and the MDCT [81, 82].

Definition 3.3.5. The basis functions $w_{m,n} \in \mathbb{C}^L$ of a Wilson basis for $\mathbb{C}^L$ with $M \in \mathbb{N}$ channels and where $g \in \mathbb{C}^L$ is WPE if $c_t = 0$ or HPE if $c_t = \frac{1}{2}$ are given by:

If $m = 0$:

$$w_{0,n}(l) = g(l - 2nM)$$

If m is odd and less than M:

$$\begin{aligned} w_{m,n}(l) &= \sqrt{2}\sin(2\pi \frac{m}{2M}(l + c_t))g(l - 2nM) \\ w_{m+M,n}(l) &= \sqrt{2}\cos(2\pi \frac{m}{2M}(l + c_t))g(l - (2n+1)M) \end{aligned}$$

If m is even and less than M:

$$\begin{aligned} w_{m,n}(l) &= \sqrt{2}\cos\left(2\pi \frac{m}{2M}(l + c_t)\right) g(l - 2nM) \\ w_{m+M,n}(l) &= \sqrt{2}\sin\left(2\pi \frac{m}{2M}(l + c_t)\right) g(l - (2n+1)M) \end{aligned}$$

If $m = M$ and M is even:

$$w_{M,n}(l) = (-1)^l g(l - 2nM)$$

else if $m = M$ and M is odd:

$$w_{M/2,n}(l) = (-1)^l g(l - (2n+1)M).$$

3.4 Symmetric splines for the real line

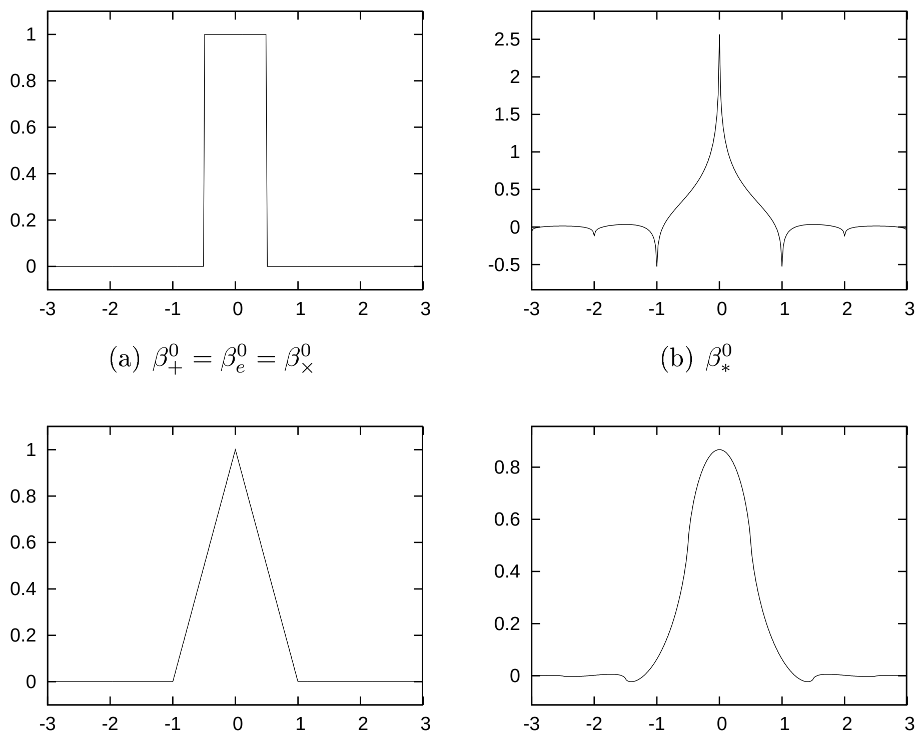
The β_+^α spline defined by (3.2) is not symmetric for α not being an integer, which means in cannot be used as a window function with Wilson bases. In the following, we shall present three ways to overcome this problem.

The first way is to add β_+^α to its own reverse:

$$\beta_e^\alpha(x) = \frac{\beta_+^\alpha(x) + \beta_+^\alpha(-x)}{2}. \tag{3.16}$$

In the Fourier domain, this has the expression

$$\widehat{\beta_e^\alpha} = \frac{\widehat{\beta_+^\alpha} + \overline{\widehat{\beta_+^\alpha}}}{2} = \Re\left(\widehat{\beta_+^\alpha}\right). \tag{3.17}$$



(a) $\beta_+^0 = \beta_e^0 = \beta_\times^0$ (b) β_*^0

(c) $\beta_+^1 = \beta_e^1 = \beta_*^1$ (d) $\beta_\times^1$

Figure 3.1: The figure shows the B-splines of order 0 and 1.

Another way of looking at this is, that β_e^α is the even part of β_+^α. This can be seen from (3.16).

In [12] Blu and Unser's presents a family of symmetric, fractional splines, β_*^α. These splines coincides with the regular B-splines only when α is an odd integer. In particular, β_*^0 is not the box function. For our purpose, namely Gabor analysis, this is a downside, because the box function often (depending on the parameters α and β) generates a tight Gabor frame. We will therefore define another family of B-splines, $\beta_\times^\alpha$, which coincides with the regular B-splines for α being an even integer.

The definition of β_*^α and $\beta_\times^\alpha$ comes from substituting the normal power function x^α with its signed and unsigned $|x|^\alpha$ counterparts. The *signed power* of a real number is given by:

$$s^{\langle\alpha\rangle} = |s|^{\alpha-1} s = |s|^\alpha \operatorname{sign}(s), \quad \forall s \in \mathbb{R}. \tag{3.18}$$

We can now define the three types of B-splines:

Definition 3.4.1. Let $\alpha > -\frac{1}{2}$. Then $\beta_e^\alpha, \beta_*^\alpha, \beta_\times^\alpha \in L^2(\mathbb{R})$ are given in the Fourier domain by

$$\widehat{\beta_e^\alpha}(\omega) = \Re\left(\operatorname{sinc}(\omega)^{\alpha+1}\right), \quad \omega \in \mathbb{R}, \tag{3.19}$$

$$\widehat{\beta_*^\alpha}(\omega) = |\operatorname{sinc}(\omega)|^{\alpha+1}, \quad \omega \in \mathbb{R}, \tag{3.20}$$

$$\widehat{\beta_\times^\alpha}(\omega) = \operatorname{sinc}(\omega)^{\langle\alpha+1\rangle}, \quad \omega \in \mathbb{R}. \tag{3.21}$$

We shall use the notation β^α in results that hold for all three type of B-splines. When

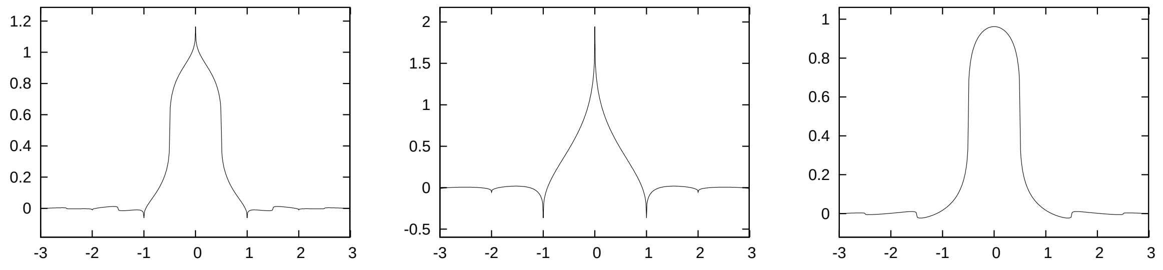


Figure 3.2: The three splines of order $\alpha = 0.3$. From left to right: β_e^α, β_*^α and $\beta_\times^\alpha$.

$\alpha > -\frac{1}{2}$ then sinc raised to $\alpha+1$ is an $L^2(\mathbb{R})$ function, and because the Fourier transform maps $L^2(\mathbb{R})$ to $L^2(\mathbb{R})$, then $\beta^\alpha \in L^2(\mathbb{R})$.

It is clear from the definition that $\beta_e^\alpha = \beta_\times^\alpha$ for even values of α, and $\beta_e^\alpha = \beta_*^\alpha$ for odd values of α. Contrary to the regular B-splines then $\beta_\times^\alpha$ is not compactly supported for odd values of α, and β_*^α is not compactly supported for even values of α. For integer values of α, β_e^α is equal to the normal, compactly supported B-splines. Figure 3.1 shows the B-splines of order 0 and 1, where these differences are most visible. All the splines quickly start to resemble the Gaussian function as $\alpha \to \infty$. Figure 3.2 shows the three symmetric B-splines for order $\alpha = 0.3$.

From the definition and (3.9) it can be seen that the splines have the following simple convolution properties for $\alpha > -\frac{1}{2}$:

$$\beta_e^{\alpha+1} = \beta_e^\alpha * \beta_+^0 \tag{3.22}$$

$$\beta_*^{\alpha+1} = \beta_*^\alpha * \beta_*^0 \tag{3.23}$$

$$\beta_\times^{\alpha+1} = \beta_*^\alpha * \beta_+^0 \tag{3.24}$$

In [101], pure time domain expressions are given for β_+^α and β_*^α. By adapting the proof, the same decay result can be shown for β_e^α and $\beta_\times^\alpha$:

$$|\beta^\alpha(x)| \le C_t |x|^{-(\alpha+2)}, \tag{3.25}$$

where C_t is a constant independent of x.

The decay in frequency is straightforward to show from the definition:

$$\left|\widehat{\beta^\alpha}(\omega)\right| \le C_f |\omega|^{-(\alpha+1)}, \tag{3.26}$$

where C_f is a constant independent of ω. Using the decay results, it is easy to show that $\beta^\alpha \in W(\mathbb{R})$ when $\alpha \ge 0$ and $\widehat{\beta^\alpha} \in W(\mathbb{R})$ when $\alpha > 0$.

The three types of splines all form a partition of unity (PU):

$$\sum_{k\in\mathbb{Z}} \beta^\alpha(x+k) = 1, \quad a.e. x \in \mathbb{R}. \tag{3.27}$$

Since $\beta^\alpha \in L^1(\mathbb{R})$, this is a consequence of the fact that

$$\widehat{\beta^\alpha}(n) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } n \in \mathbb{Z}\setminus\{0\} \end{cases} \tag{3.28}$$

For a simple proof, see [105, Cor. 7.54].

3.5 Discrete splines by sampling and periodization

In the papers [79, 53, 62, 92] a theory for finite, discrete Gabor frames obtained from sampled and periodized Gabor frames for $\mathbb{C}^L$ has been developed. The main results are as follows: Suppose that $g \in L^2(\mathbb{R})$ (satisfying certain conditions) generates a Gabor frame for $L^2(\mathbb{R})$. Then

- $g^D \in \mathbb{C}^L$ obtained from g by sampling and periodization will generate a Gabor frame for $\mathbb{C}^L$ with the same (or better) frame bounds.
- The canonical dual/tight windows of these two Gabor systems are related by sampling and periodization as well.
- If $f \in L^2(\mathbb{R})$ and $f^D \in \mathbb{C}^L$ are also related by sampling and periodization then

  $$\left\langle f^D, g^D_{m,n} \right\rangle = \sum_{r\in\mathbb{Z}} \sum_{s\in\mathbb{Z}} \left\langle f, g_{m+\frac{rL}{b}, n+\frac{sL}{a}} \right\rangle . \tag{3.29}$$

  This corresponds to an aliasing in time and frequency of the expansion coefficients from the continuous Gabor system, very similar to the well known phenomena for Fourier series and the DFT.

This theory is the reason for trying to define finite, discrete fractional splines by sampling and periodization.

We define the finite, discrete splines by sampling and periodization of their continuous counterparts. We shall denote splines that are WPE by a capital 'W' and similarly denote splines that are HPE by a capital 'H', see Definition 3.3.3 and 3.3.4:

Definition 3.5.1. We define the discrete splines $C^{\alpha,+W}_{L,a}$, $C^{\alpha,eW}_{L,a}$, $C^{\alpha,*W}_{L,a}$, $C^{\alpha,\times W}_{L,a}$, $C^{\alpha,+H}_{L,a}$, $C^{\alpha,eH}_{L,a}$, $C^{\alpha,*H}_{L,a}$, $C^{\alpha,\times H}_{L,a}$ by sampling and periodization of their continuous counterparts:

$$C^{\alpha,W}_{L,a}(l) = \frac{1}{a} \sum_{n\in\mathbb{Z}} \beta^{\alpha}\left(\frac{l}{a} + nN\right), \quad l = 0, \dots, L-1 \tag{3.30}$$

$$C^{\alpha,H}_{L,a}(l) = \frac{1}{a} \sum_{n\in\mathbb{Z}} \beta^{\alpha}\left(\frac{2l+1}{2a} + nN\right), \quad l = 0, \dots, L-1. \tag{3.31}$$

The splines $C^{\alpha,+W}_{L,a}$, $C^{\alpha,eW}_{L,a}$, $C^{\alpha,\times W}_{L,a}$ $C^{\alpha,+H}_{L,a}$, $C^{\alpha,eH}_{L,a}$ $C^{\alpha,\times H}_{L,a}$ are defined for $\alpha \geq 0$, while $C^{\alpha,*W}_{L,a}$ and $C^{\alpha,*H}_{L,a}$ are only defined for $\alpha > 0$. This is because the periodization of β^0_* is divergent.

To find Fourier domain expressions for $\alpha > 0$ for these splines, the main tool will be the Poisson summation formula:

Lemma 3.5.2 (The Poisson summation formula). . *Let $f \in L^2(\mathbb{R})$ then*

$$\sum_n f(x + nN) = \frac{1}{L} \sum_k \hat{f}\left(\frac{k}{N}\right) e^{2\pi i k x/N}, \quad a.e.\ x \in \mathbb{R}. \tag{3.32}$$

When both $f, \hat{f} \in W(\mathbb{R})$, the Poisson summation formula hold with pointwise convergence everywhere, [41], and this is exactly the case for β^α when $\alpha > 0$.

We shall make use of *Hurwitz' zeta function* [47], also known as the generalized zeta function:

$$\zeta(z, v) = \sum_{k=0}^{\infty} \frac{1}{(k+v)^z}, \quad z > 1,\, v > 0 \tag{3.33}$$

In the following, we set $\beta = \alpha + 1$ to shorten the formulas. Using the Poisson summation formula (3.32) and Hurwitz' zeta function, we obtain Fourier domain expressions for for the splines defined in Definition 3.5.1:

Proposition 3.5.3. *For $\alpha > 0$ we have the following. For even a and $m = 1, \ldots, L-1$.*

$$\widehat{C_{L,a}^{\alpha,eW}}(m) = \frac{1}{\sqrt{L}} \Re \left(\left(\frac{\sin\left(\frac{\pi m}{N}\right)}{\pi a} \right)^{\beta} \left(\zeta\left(\beta, \frac{m}{L}\right) + e^{-\pi i \beta} \zeta\left(\beta, 1 - \frac{m}{L}\right) \right) \right) \tag{3.34}$$

$$\widehat{C_{L,a}^{\alpha,*W}}(m) = \frac{1}{\sqrt{L}} \left| \frac{\sin\left(\frac{\pi m}{N}\right)}{\pi a} \right|^{\beta} \left(\zeta\left(\beta, \frac{m}{L}\right) + \zeta\left(\beta, 1 - \frac{m}{L}\right) \right) \tag{3.35}$$

$$\widehat{C_{L,a}^{\alpha,\times W}}(m) = \frac{1}{\sqrt{L}} \left(\frac{\sin\left(\frac{\pi m}{N}\right)}{\pi a} \right)^{\langle\beta\rangle} \left(\zeta\left(\beta, \frac{m}{L}\right) - \zeta\left(\beta, 1 - \frac{m}{L}\right) \right) \tag{3.36}$$

For odd a and $m = 1, \ldots, L-1$ we get the expressions

$$\begin{aligned} \widehat{C_{L,a}^{\alpha,eW}}(m) &= \frac{1}{\sqrt{L}} \Re \Bigg(\left(\frac{\sin\left(\frac{\pi m}{N}\right)}{2\pi a} \right)^{\beta} \Bigg((-1)^{-\beta} \zeta\left(\beta, 1 - \frac{m}{2L}\right) + \zeta\left(\beta, \frac{m}{2L}\right) + \\ &\quad + \zeta\left(\beta, \frac{1}{2} - \frac{m}{2L}\right) + (-1)^{\beta} \zeta\left(\beta, \frac{1}{2} + \frac{m}{2L}\right) \Bigg) \Bigg) \end{aligned} \tag{3.37}$$

$$\begin{aligned} \widehat{C_{L,a}^{\alpha,\times W}}(m) &= \frac{1}{\sqrt{L}} \left(\frac{\sin\left(\frac{\pi m}{N}\right)}{2\pi a} \right)^{\langle\beta\rangle} \Bigg(\zeta\left(\beta, \frac{m}{2L}\right) - \zeta\left(\beta, 1 - \frac{m}{2L}\right) + \\ &\quad + \zeta\left(\beta, \frac{1}{2} - \frac{m}{2L}\right) - \zeta\left(\beta, \frac{1}{2} + \frac{m}{2L}\right) \Bigg). \end{aligned} \tag{3.38}$$

The expression (3.35) holds for both even and odd a. For $m = 0$ and all values of a we get

$$\widehat{C_{L,a}^{\alpha,eW}}(0) = \widehat{C_{L,a}^{\alpha,*W}}(0) = \widehat{C_{L,a}^{\alpha,\times W}}(0) = \frac{1}{\sqrt{L}}. \tag{3.39}$$

For a derivation, see Sec. 3.9.

The discrete splines defined in this manner does not satisfy the discrete equivalent of the convolution properties (3.22)-(3.24). Instead, because these splines are formed from sampling and periodization, they have the subsampling property

$$C_{L,a}^{\alpha,W}(k) = \sqrt{c}\, C_{Lc,ac}^{\alpha,W}(ck), \quad k = 0, \ldots, L-1, \tag{3.40}$$

where $c \in \mathbb{N}$. This means, that if one picks out a regularly spaced subsequence of the splines containing the first element, then this sequence is again a spline of the same order and type. Another consequence is that interleaving a WPE spline with the corresponding HPE spline will form the WPE spline (differently scaled) of twice L and a.

3.6 Discrete splines by the DFT

In the previous section we defined finite, discrete splines by means of sampling and periodization. In this section we will define finite, discrete splines by transferring the continuous definition to the discrete case. We wish to create both WPE and HPE spline functions. For this the following simple observations can be used:

- The convolution of two WPE functions is again a WPE function.
- The convolution of an HPE and a WPE function is a again an HPE function.

However, it does not hold that the convolution of two HPE functions is again a HPE function. Contrary, it is a WPE function shifted by one sample.

We can construct splines by repeated convolution of a rectangular function, just as in the continuous case. The zeroth order splines is defined by a sampling and periodization of the box spline. Because the box spline is compactly supported, the periodization consists of only two terms:

Definition 3.6.1. Let $a = \{0, \ldots, L-1\}$. We define $D_{L,a}^{0,+W}, D_{L,a}^{0,+H} \in \mathbb{C}^L$ by:

$$D_{L,a}^{0,+W}(l) = \frac{1}{\sqrt{a}}\left(\beta_+^0\left(\frac{l-L}{a}\right) + \beta_+^0\left(\frac{l}{a}\right)\right), \tag{3.41}$$

$$D_{L,a}^{0,+H}(l) = \frac{1}{\sqrt{a}}\left(\beta_+^0\left(\frac{l+\frac{1}{2}-L}{a}\right) + \beta_+^0\left(\frac{l+\frac{1}{2}}{a}\right)\right), \tag{3.42}$$

for $l = 0, \ldots, L-1$.

By standard trigonometric formulas, we find that the DFT of $D_{L,a}^{0,W}$ is

$$\widehat{D_{L,a}^{0,+W}}(k) = \frac{2}{\sqrt{L}}\left(\sum_{l=1}^{\lceil (a-1)/2\rceil} b(l)\cos(2\pi k l/L)\right) \tag{3.43}$$

$$= \begin{cases} \frac{\sin\left(a\frac{\pi l}{L}\right)}{\sin\left(\frac{\pi l}{L}\right)} & \text{if } a \text{ is odd} \\ \frac{\sin\left((a-1)\frac{\pi l}{L}\right)+\sin\left((a+1)\frac{\pi l}{L}\right)}{2\sin\left(\frac{\pi l}{L}\right)} & \text{if } a \text{ is even} \end{cases} \tag{3.44}$$

where

$$b(l) = \begin{cases} \frac{1}{2} & \text{if } l = 0 \text{ or } l = \frac{a}{2} \\ 1 & \text{otherwise.} \end{cases} \tag{3.45}$$

We define the discrete, fractional WPE B-splines in the same way as their continuous counterparts.

Definition 3.6.2. Let $\alpha > -1$ and $a = \{0, \ldots, L-1\}$. We define $D^{\alpha,eW}_{L,a}, D^{\alpha,*W}_{L,a}, D^{\alpha,\times W}_{L,a} \in \mathbb{C}^L$

$$\widehat{D^{\alpha,eW}_{L,a}} = L^{\frac{\alpha}{2}} \Re\left(\left(\widehat{D^{\alpha,W}_{L,a}}\right)^{\alpha+1}\right) \tag{3.46}$$

$$\widehat{D^{\alpha,*W}_{L,a}} = L^{\frac{\alpha}{2}} \left|\widehat{D^{\alpha,W}_{L,a}}\right|^{\alpha+1} \tag{3.47}$$

$$\widehat{D^{\alpha,\times W}_{L,a}} = L^{\frac{\alpha}{2}} \left(\widehat{D^{\alpha,W}_{L,a}}\right)^{\langle\alpha+1\rangle} \tag{3.48}$$

The constant term $L^{\frac{\alpha}{2}}$ is necessary to ensure the proper normalization. It comes from the $\sqrt{L}$ appearing in (3.10).

The Fourier domain methods used so far need to be modified for defining the HPE splines, because the Fourier transform of an HPE function is not real valued. Instead we will use the convolution properties (3.22)-(3.24) as the definition. We define the zero-order splines by:

The unsigned power spline of order zero can be defined in the Fourier domain using (3.15):

Definition 3.6.3. Let $a = \{0, \ldots, L-1\}$. We define $D^{0,*H}_{L,a} \in \mathbb{C}^L$ in the Fourier domain by

$$\widehat{D^{0,*H}_{L,a}}(k) = \begin{cases} \left|\widehat{B^{0,+HD}_{L,a}}(k)\right| e^{\pi i k/L} & \text{if } 0 \leq k < \left\lfloor \frac{L}{2} \right\rfloor \\ -\left|\widehat{B^{0,+HD}_{L,a}}(k)\right| e^{\pi i k/L} & \text{otherwise} \end{cases}.$$

Using the two zero order HPE splines just defined, we can define all the HPE splines:

Definition 3.6.4. Let $a = \{0, \ldots, L-1\}$ and $\alpha > 0$. We define $D^{\alpha,eH}_{L,a}$, $D^{\alpha,*H}_{L,a}$, $D^{\alpha,\times H}_{L,a} \in \mathbb{C}^L$ in the Fourier domain by:

$$\widehat{D^{\alpha,eH}_{L,a}} = L^{\frac{\alpha}{2}} \Re\left(\left(\widehat{D^{0,W}_{L,a}}\right)^{\alpha}\right) \widehat{D^{0,+H}_{L,a}}, \tag{3.49}$$

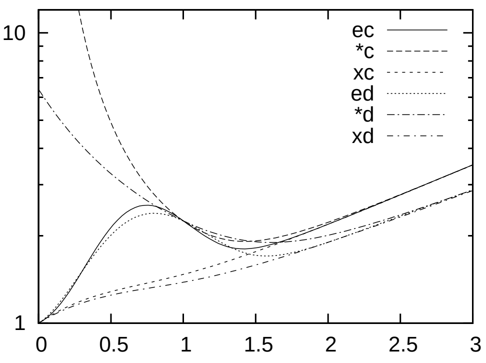
$$\widehat{D^{\alpha,*H}_{L,a}} = L^{\frac{\alpha}{2}} \left|\widehat{D^{0,W}_{L,a}}\right|^{\alpha} \widehat{D^{0,*H}_{L,a}}, \tag{3.50}$$

$$\widehat{D^{\alpha,\times H}_{L,a}} = L^{\frac{\alpha}{2}} \left|\widehat{D^{0,W}_{L,a}}\right|^{\alpha} \widehat{D^{0,H}_{L,a}}. \tag{3.51}$$

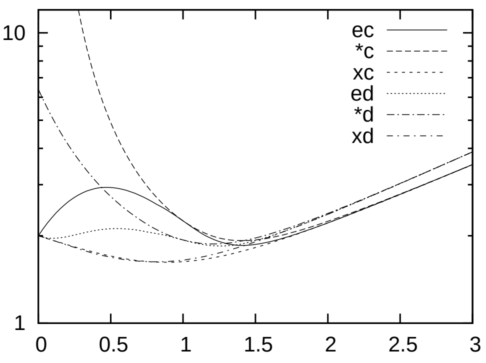
3.7 Numerical results

In this section we consider Gabor frames and Wilson bases as defined in Definition 3.3.2 and 3.3.5.

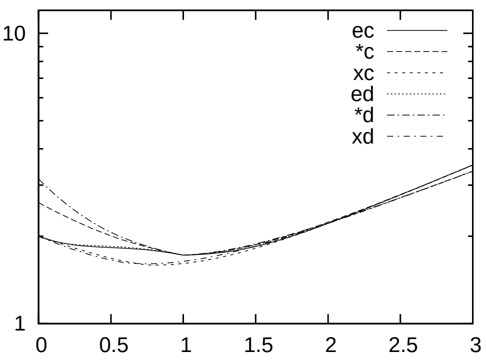
The six different types of splines (both in WPE and HPE variation) have been implemented in the Linear Time Frequency Toolbox (LTFAT) available from `http://www.`



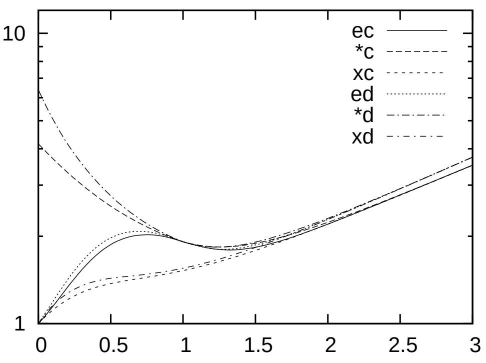
(a) odd a, WPE



(b) even a, WPE

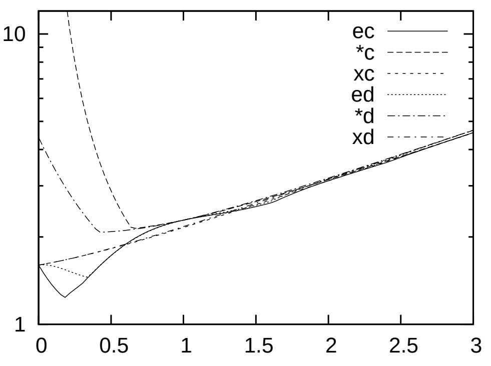


(c) odd a, HPE

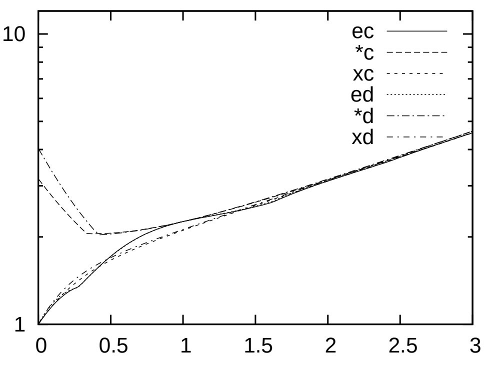


(d) even a, HPE

Figure 3.3: The figure show the frame bound ratio $\frac{B}{A}$ of a Gabor frame using one of the six types of splines as window function. The parameters for the left plot are $L = 48$, $a = 3$ and $M = 4$. For the right plot they are $L = 96$, $a = 6$ and $M = 8$. The top row shows WPE windows and the bottom row shows HPE windows.



(a) WPE



(b) HPE

Figure 3.4: The figure show the frame bound ratio $\frac{B}{A}$ of a Wilson basis as defined in Definition 3.3.5 using one of the six types of splines as window function. The parameters are $L = 36$ using $M = 6$ channels. The left figure shows a Wilson basis with $c_t = 0$ using WPE splines as window. The right plot show a Wilson basis with $c_t = \frac{1}{2}$ using HPE splines as windows.

`univie.ac.at/nuhag-php/ltfat` as the function `pbspline` (Periodic B-spline). Implementations of Gabor systems, Wilson bases and frame bound calculations are available as well.

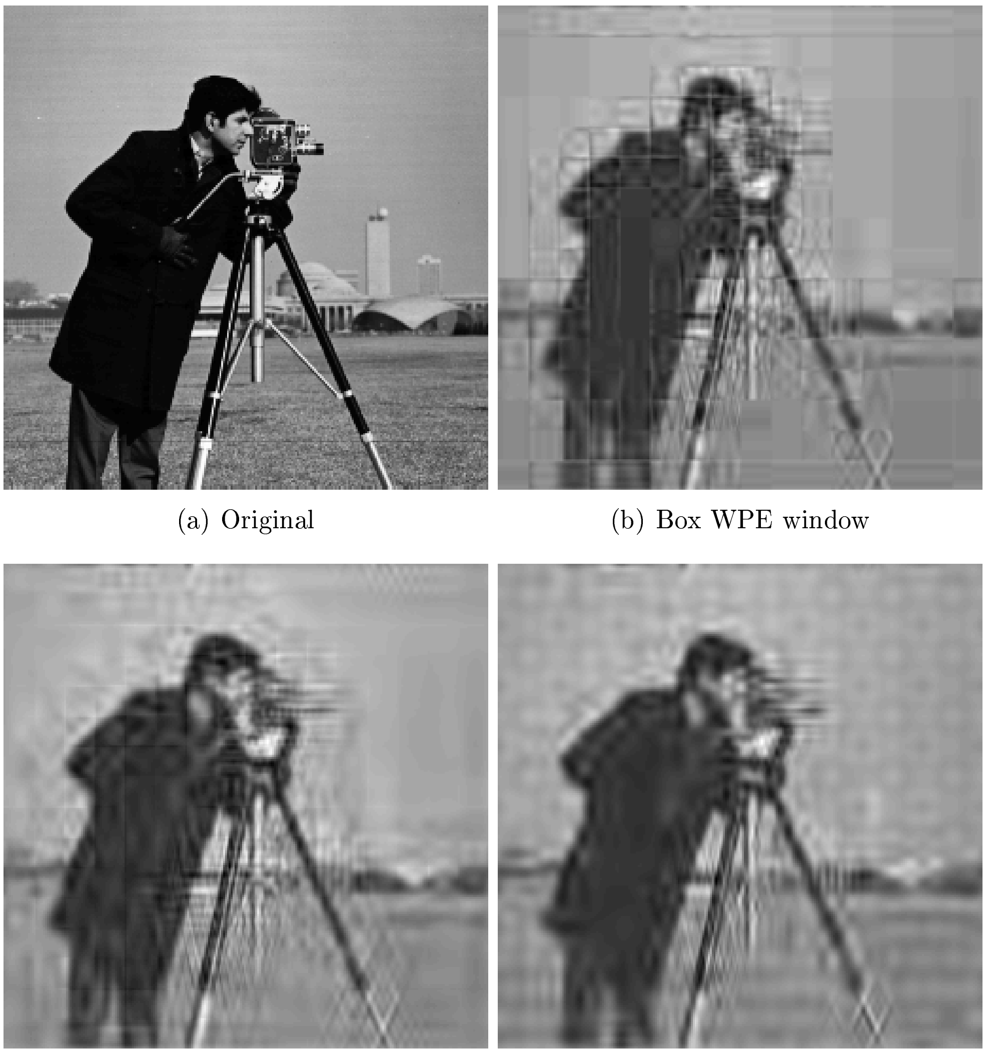
Implementations of Hurwitz' zeta function needed for the 'C' splines can (among other places) be found in GSL (the GNU Scientific Library), Maple and Mathematica. Octave uses the GSL implementation and Matlab uses the Maple implementation.

Figure 3.3 show a comparison of the frame bound ratio $\frac{B}{A}$ of Gabor frames using splines windows with different values of α. The windows that consistently generates the lowest frame bound ratios are the signed power splines, $C^{\alpha,\times}$ and $D^{\alpha,\times}$, and the unsigned power splines $C^{\alpha,*}$ and $D^{\alpha,*}$ generate the highest. The even splines, $C^{\alpha,e}$ and $D^{\alpha,e}$, oscillates between these two, as would be expected.

For values of α close to zero, using a WPE spline when a is odd and an HPE spline when a is even gives much lower frame bounds that doing the opposite. The cause of this is that $D_{L,a}^{0,+W}$ for a odd and $D_{L,a}^{0,+H}$ for a even consists of only the values 0 and 1. Because $D_{L,a}^{0,+W}$ forms a PU, then also $g(k) = \left|D_{L,a}^{0,+W}(k)\right|^2$ forms a PU. This is exactly the requirement for generating a tight Gabor frame for this case.

Figure 3.4 show the same investigation for Wilson bases. We have constructed a Wilson basis for which the unmodulated terms form a PU. This means that we must choose the parameter a for the splines as $a = 2M$, and therefore the value of a cannot be odd. We see similar result as for Gabor frames, except that the 'e' splines may sometimes generate frames with a lower frame bound ratio than those generated by the 'x' splines.

Figure 3.5 shows the effect of using a window that forms a PU in image compression. The image used is a standard test image, the 'cameraman', which has a large background area of almost constant color. For each of the three windows $C^{0,eW}$, $C^{0.4,eW}$ and the Gaussian, the test image has been heavily compressed so as to produce visible artifacts of the compression. The two spline windows that form a PU provides a smooth resolution of the background, while the Gaussian produces a visibly disturbing pattern in the background.


(a) Original (b) Box WPE window

(c) 'ec' WPE spline window with $\alpha = .4$ (d) Gauss window

Figure 3.5: The figure shows a test done on a standard test image, the Cameraman. The image in the upper left corner is the original 256x256 greyscale image. The other 3 images have been compressed using a 2D Wilson basis with $M = 16$ channels and different window functions as specified below each image. The specified window was used for synthesis and the corresponding canonical dual window was used for analysis. The 1% largest coefficients have been kept, and all other coefficients set to zero.

The box function on the other hand produces a sharp image but with visible horizontal and vertical lines, because of the sharp cutoff.[1] The image produced by the fractional spline $C^{0.4,eW}$ shows a smoother background without disturbing lines or patterns.

3.8 Conclusion

We have introduced two families of finite, discrete, symmetric B-splines suitable for discrete Gabor analysis:

- A family of discrete splines constructed by sampling and periodization of the continuous splines. These splines are suitable for Gabor analysis if we are working with signals coming from a sampling of continuous signals, and desire a precise estimate of sampling errors. To compute these splines, evaluation of Hurwitz' zeta function is needed, which can be slow if not properly implemented.
- A family of discrete splines defined in the same manner as their continuous counterparts. These splines are very similar to the splines produced by sampling and periodization, but lack the close relationship to the continuous splines. They are very fast to compute, requiring only a single DFT.

For Gabor systems, the splines formed by raising the sinc function to a signed power consistently generated Gabor frames with a lower frame bound ratio than the other types of splines.

An implementation of the different type of splines discussed in this paper is available in the LTFAT toolbox as the function `pbspline`. By default this function will return a spline of type $B_{L,a}^{e,D}$, as this spline coincides with the normal B-splines for all integer orders, and because it is fast and reliable to compute (computation does not depend on an external library to be available).

3.9 Derivation of the periodic, discrete fractional splines by sampling and periodization

We wish to find an explicit expression for the DFT of

$$C_{L,a}^{\alpha,+W}(l) = \frac{1}{a}\sum_{n} \beta_{+}^{\alpha}\left(\frac{l}{a} + nN\right). \tag{3.52}$$

To derive this, we will need to consider the following two sums: For the first one, we expand the sum in (3.33) to run through all whole numbers:

[1]When using this window a 2D Wilson transform is almost the same transform as the 2D block DCT that is used in JPEG encoding of images.

$$\begin{aligned}\sum_{k\in\mathbb{Z}}(z+k)^{-\beta} &= \sum_{k=1}^{\infty}(z-k)^{-\beta}+\sum_{k=0}^{\infty}(z+k)^{-\beta} && (3.53)\\ &= \sum_{k=0}^{\infty}(z-(k+1))^{-\beta}+\sum_{k=0}^{\infty}(z+k)^{-\beta} && (3.54)\\ &= (-1)^{-\beta}\zeta(\beta,1-z)+\zeta(\beta,z) && (3.55)\end{aligned}$$

For the second one, we consider a sum with alternating signs. We split it for k even and k odd and use (3.55):

$$\begin{aligned}\sum_{k\in\mathbb{Z}}\frac{\left((-1)^k\right)^{\beta}}{(z+k)^{\beta}} &= \sum_{k\in\mathbb{Z}}\frac{1}{(z+2k)^{\beta}}+\sum_{k\in\mathbb{Z}}\frac{(-1)^{\beta}}{(z+2k+1)^{\beta}} && (3.56)\\ &= 2^{-\beta}\left(\sum_{k\in\mathbb{Z}}\left(\frac{z}{2}+k\right)^{-\beta}+(-1)^{\beta}\sum_{k\in\mathbb{Z}}\left(\frac{z+1}{2}+k\right)^{-\beta}\right) && (3.57)\\ &= 2^{-\beta}\Bigg((-1)^{-\beta}\zeta\left(\beta,1-\frac{z}{2}\right)+\zeta\left(\beta,\frac{z}{2}\right)+\\ &+ \zeta\left(\beta,\frac{1}{2}-\frac{z}{2}\right)+(-1)^{\beta}\zeta\left(\beta,\frac{1}{2}+\frac{z}{2}\right)\Bigg) && (3.58)\end{aligned}$$

Using the Poisson summation formula (3.32) and the well-known aliasing of high-mode waves to low mode waves (also a form of Poisson-summation) we obtain

$$\begin{aligned}C_{L,a}^{\alpha,+W}(l) &= \frac{1}{a}\sum_{n}\beta_{+}^{\alpha}\left(\frac{l}{a}+nN\right) && (3.59)\\ &= \frac{1}{L}\sum_{k\in\mathbb{Z}}\operatorname{sinc}\left(\frac{k}{N}\right)^{\alpha+1}e^{2\pi ikl/L} && (3.60)\\ &= \frac{1}{L}\sum_{m=0}^{L-1}\left(\sum_{k\in\mathbb{Z}}\operatorname{sinc}\left(\frac{m+kL}{N}\right)^{\alpha+1}\right)e^{2\pi ilm/L} && (3.61)\\ &= \frac{1}{L}\sum_{m=0}^{L-1}\left(\sum_{k\in\mathbb{Z}}\operatorname{sinc}\left(\frac{m}{N}+ka\right)^{\alpha+1}\right)e^{2\pi ilm/L} && (3.62)\end{aligned}$$

for $j=0,\dots,L-1$ and $\alpha>0$. The infinite sum is absolute convergent because we have assumed $\alpha>0$. From the last expression we recognize the discrete Fourier transform of $C_{L,a}^{\alpha,+W}$:

$$\widehat{C_{L,a}^{\alpha,+W}}(m)=\frac{1}{\sqrt{L}}\sum_{k\in\mathbb{Z}}\operatorname{sinc}\left(\frac{m}{N}+ka\right)^{\alpha+1},\quad m=0,\dots,L-1. \qquad (3.63)$$

To get rid of the infinite sum, we rewrite for $m=1,\dots,L-1$:

$$\widehat{C^{\alpha,+W}_{L,a}}(m) = \frac{1}{\sqrt{L}} \sum_{k\in\mathbb{Z}} \operatorname{sinc}\left(\frac{m}{N} + ka\right)^{\alpha+1} \tag{3.64}$$

$$= \frac{1}{\sqrt{L}} \sum_{k\in\mathbb{Z}} \left(\frac{\sin\left(\frac{\pi m}{N} + \pi ka\right)}{\frac{\pi m}{N} + \pi ka}\right)^{\alpha+1} \tag{3.65}$$

$$= \frac{1}{\sqrt{L}} \sin\left(\frac{\pi m}{N}\right)^{\alpha+1} \sum_{k\in\mathbb{Z}} \frac{\left((-1)^{ka}\right)^{\alpha+1}}{\left(\frac{\pi m}{N} + \pi ka\right)^{\alpha+1}} \tag{3.66}$$

$$= \frac{1}{\sqrt{L}} \left(\frac{\sin\left(\frac{\pi m}{N}\right)}{\pi a}\right)^{\alpha+1} \sum_{k\in\mathbb{Z}} \frac{\left((-1)^{ka}\right)^{\alpha+1}}{\left(\frac{m}{L} + k\right)^{\alpha+1}} \tag{3.67}$$

The last expression (3.67) simplifies when a is even. We therefore first treat the case when a is even and use (3.55) with $\beta = \alpha + 1$:

$$\widehat{B^{\alpha,+WC}_{L,a}}(m) \tag{3.68}$$

$$= \frac{1}{\sqrt{L}} \left(\frac{\sin\left(\frac{\pi m}{N}\right)}{\pi a}\right)^{\alpha+1} \sum_{k\in\mathbb{Z}} \frac{1}{\left(\frac{m}{L} + k\right)^{\alpha+1}} \tag{3.69}$$

$$= \frac{1}{\sqrt{L}} \left(\frac{\sin\left(\frac{\pi m}{N}\right)}{\pi a}\right)^{\alpha+1} \left((-1)^{-(\alpha+1)} \zeta\left(\alpha+1, 1 - \frac{m}{L}\right) + \zeta\left(\alpha+1, \frac{m}{L}\right)\right). \tag{3.70}$$

To treat the case when a is odd, we use use (3.58) on (3.67):

$$\widehat{B^{\alpha,+WC}_{L,a}}(m) = \frac{1}{\sqrt{L}} \left(\frac{\sin\left(\frac{\pi m}{N}\right)}{\pi a}\right)^{\alpha+1} \sum_{k\in\mathbb{Z}} \frac{\left((-1)^{k}\right)^{\alpha+1}}{\left(\frac{m}{L} + k\right)^{\alpha+1}} \tag{3.71}$$

$$= \frac{1}{\sqrt{L}} \left(\frac{\sin\left(\frac{\pi m}{N}\right)}{2\pi a}\right)^{\alpha+1} \left((-1)^{-(\alpha+1)} \zeta\left(\alpha+1, 1 - \frac{m}{2L}\right) + \zeta\left(\alpha+1, \frac{m}{2L}\right) + \right.$$
$$\left. + \zeta\left(\alpha+1, \frac{1}{2} - \frac{m}{2L}\right) + (-1)^{\alpha+1} \zeta\left(\alpha+1, \frac{1}{2} + \frac{m}{2L}\right)\right) \tag{3.72}$$

For $m = 0$ we get trivially from the properties of the sinc function that

$$\widehat{C^{\alpha,+W}_{L,a}}(0) = \frac{1}{\sqrt{L}}. \tag{3.73}$$

The derivations of $\widehat{C^{\alpha,*W}_{L,a}}$ and $\widehat{C^{\alpha,\times W}_{L,a}}$ are very similar.

Chapter 4

An Efficient Algorithm for the Discrete Gabor Transform using full length Windows

This chapter is the paper [93], which has been submitted to IEEE Signal Processing Letters, 2007.

4.1 Abstract

This paper extends the efficient factorization of the Gabor frame operator developed by Strohmer in [96] to the Gabor analysis/synthesis operator. This provides a fast method for computing the discrete Gabor transform (DGT) and several algorithms associated with it. The algorithm is used for the case when the involved window and signal have the same length.

4.2 Introduction

The finite, discrete Gabor transform (DGT) of a signal f of length L is given by

$$c\left(m,n,w\right)=\sum_{l=0}^{L-1}f(k,w)\overline{g\left(l-an\right)}e^{-2\pi iml/M}. \tag{4.1}$$

Here g is a window that localizes the signal in time and in frequency. The DGT is equivalent to a Fourier modulated filter bank with M channels and decimation in time a, [18].

Efficient computation of a DGT can be done by several methods: If the window g has short support, a filter bank based approach can be used. We shall instead focus on the case when g and f are equally long. In this case a factorization approach developed by Zibulski and Zeevi in [111] for the case of the frame operator of Gabor frames for $L^2\left(\mathbb{R}\right)$ can be used. The method was adapted for the finite, discrete setting by Bastiaans and Geilen in [7], and extended to cover also the analysis/synthesis operator. A simple, but not so efficient, method was developed for the Gabor analysis/synthesis operator by

Prinz in [83] and later extended by Strohmer [96] to provide the fastest known method for computing the Gabor frame operator. This paper extends Prinz' and Strohmer's method to also cover the Gabor analysis and synthesis methods on multisignals.

The advantage of the method developed in this paper as compared to the one developed in [7], is that it works with FFTs of shorter length, and do not require multiplication by complex exponentials. The asymptotic running time of the two method are the same.

We shall study the DGT applied to multiple signals at once. This is a common case for instance when computing a multidimensional, separable DGT, then this can be done by applying several multisignal DGTs. The DGT defined by (4.1) works on a multisignal $f \in \mathbb{C}^{L\times W}$, where $W \in \mathbb{N}$ is the number of signals.

4.3 Definitions

We shall denote the set of integers between zero and some number L by

$$\langle L\rangle = 0, \ldots, L-1. \tag{4.2}$$

The Discrete Fourier Transform (DFT) of a signal $f \in \mathbb{C}^L$ is defined by

$$\left(\mathcal{F}_L f\right)(k) \quad = \quad \frac{1}{\sqrt{L}} \sum_{l=0}^{L-1} f(l) e^{-2\pi i k l / L}. \tag{4.3}$$

We shall use the $\cdot$ notation in conjunction with the DFT to denote the variable over which the transform is to be applied.

The folding $f * g$ of two functions $f, g \in \mathbb{C}^L$ and the involution f^* is given by

$$\left(f * g\right)(l) \quad = \quad \sum_{k=0}^{L-1} f\left(k\right) g\left(l-k\right), \quad l \in \langle L\rangle \tag{4.4}$$

$$f^*\left(l\right) \quad = \quad \overline{f\left(-l\right)}, \quad l \in \langle L\rangle\,. \tag{4.5}$$

Both folding and involution has special properties with respect to the Fourier transform

$$\widehat{f * g} \quad = \quad \sqrt{L}\hat{f}\hat{g} \tag{4.6}$$

$$\widehat{f^*} \quad = \quad \overline{\hat{f}} \tag{4.7}$$

The Poisson summation formula in the finite, discrete setting is given by

$$\mathcal{F}_M\left(\sum_{k=0}^{b-1} g(\cdot + kM)\right)(m) \quad = \quad \sqrt{b}\left(\mathcal{F}_L g\right)(mb), \tag{4.8}$$

where $g \in \mathbb{C}^L$, $L = Mb$ with $b, M \in \mathbb{N}$.

A family of vectors e_j, $j \in \langle J\rangle$ of length L is called a *frame* if constants $0 < A \leq B$ exist such that

$$A\left\|f\right\|^2 \leq \sum_{j=0}^{J-1} \left|\langle f, e_j\rangle\right|^2 \leq B\left\|f\right\|^2, \quad \forall f \in \mathbb{C}^L. \tag{4.9}$$

The constants A and B are called lower and upper frame bounds. If $A = B$, the frame is called *tight.* If $J > L$, the frame is redundant (oversampled). Finite- and infinite dimensional frames are described in [23].

A finite, discrete *Gabor system* (g, a, M) is a family of functions $g_{m,n} \in \mathbb{C}^L$ of the following form

$$g_{m,n}(l) = e^{2\pi i l m/M} g(l - na), \tag{4.10}$$

for $m \in \langle M \rangle$ and $n \in \langle N \rangle$ where $L = aN$ and $M/L \in \mathbb{N}$. A Gabor system that is also a frame is called a *Gabor frame.* The analysis operator $C_g : \mathbb{C}^L \mapsto \mathbb{C}^{M\times N}$ associated to a Gabor system (g, a, M) is given by

$$c(m, n) = C_g f = \sum_{l=0}^{L-1} f(k) e^{-2\pi i m l/M} \overline{g(l - an)}. \tag{4.11}$$

This is exactly the DGT from (4.1). The adjoint operator is the Gabor synthesis operator $D_\gamma : \mathbb{C}^{M\times N} \mapsto \mathbb{C}^L$ associated to a Gabor system (γ, a, M) given by

$$f = D_\gamma c = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} c(m, n) e^{2\pi i m l/M} \gamma(l - an). \tag{4.12}$$

In (4.1) it must hold that $L = Na = Mb$ for some $M, N \in \mathbb{N}$. Additionally, we define $c, d, p, q \in \mathbb{N}$ by

$$c = \gcd(a, M) \quad , \quad d = \gcd(b, N), \tag{4.13}$$

$$p = \frac{a}{c} = \frac{b}{d} \quad , \quad q = \frac{M}{c} = \frac{N}{d}, \tag{4.14}$$

where GCD denotes the greatest common divisor of two natural numbers. With these numbers, the redundancy of the transform can be written as $L/(ab) = q/p$, where q/p is an irreducible fraction. It holds that $L = cdpq$.The Gabor *frame operator* $S_g : \mathbb{C}^L \mapsto \mathbb{C}^L$ of a Gabor frame (g, a, M) is given by the composition of the analysis and synthesis operators $S_g = D_g C_g$. The Gabor frame operator is important because it can be used to find the *canonical dual window* $g^d = S_g^{-1} g$ and the *canonical tight window* $g^t = S_g^{-1/2} g$ of a Gabor frame. The canonical dual window is important because C_g and D_{g^t} are each others inverses. This gives an easy way to construct the inverse transform of the DGT. Similarly, then C_{g^t} and D_{g^t} are each others inverses. For more information on Gabor systems and properties of the operators C, D and S see [41, 35, 36].

4.4 The method

We wish to make an efficient calculation of all the coefficients of the DGT. Using (4.1) literally to compute all coefficients $c(m, n, w)$ would require $8MNLW$ flops.

To derive a faster DGT, an approach would be to consider the analysis operator C_g as a matrix, and derive a faster algorithm through unitary matrix factorizations of this matrix. This is indeed the approach taken by [85, 96]. Unfortunately, this approach tends to introduce many permutation matrices and Kronecker product matrices, so in

Algorithm 2 Multisignal DGT by matrix/matrix products.

We wish to compute the DGT $c(m,n,w) \in \mathbb{C}^{M\times N\times W}$ of $f \in \mathbb{C}^{L\times W}$ using the window $g \in \mathbb{C}^{L}$ and lattice determined by a and M.

1. Define $\Psi^{f}_{r,s}(k,l) \in \mathbb{C}^{p\times qW}$ and $\Phi^{g}_{r,s}(k,l) \in \mathbb{C}^{p\times q}$ for $r \in \langle c\rangle$, $s \in \langle d\rangle$ and $w \in \langle W\rangle$ by

$$\begin{aligned}\tilde{\Psi}^{f}_{r,\tilde{s}}(k,l+qw) &= f(r+kM+\tilde{s}pM-lh_a a,w),\\ \tilde{\Phi}^{g}_{r,\tilde{s}}(k,l) &= \sqrt{M}dg(r+kM+\tilde{s}pM-la).\end{aligned}$$

2. Compute their DFTs along $\tilde{s}$:

$$\begin{aligned}\Psi^{f}_{r,s}(k,l+qw) &= \mathcal{F}_d\left(\tilde{\Psi}^{f}_{r,\cdot}(k,l+qw)\right)(s),\\ \Phi^{g}_{r,s}(k,l) &= \mathcal{F}_d\left(\tilde{\Phi}^{g}_{r,\cdot}(k,l)\right)(s).\end{aligned}$$

3. Multiply the matrices for each r,s:

$$\Upsilon_{r,s} = \left(\Phi^{g}_{r,s}\right)^{*}\Psi^{f}_{r,s}$$

4. Compute the inverse DFT of $\Upsilon_{r,s}$ along s:

$$\tilde{\Upsilon}_{r,\tilde{s}}(u,l+wq) = \mathcal{F}_d^{-1}\left(\Upsilon_{r,\cdot}(u,l+wq)\right)(\tilde{s}).$$

5. Compute $K \in \mathbb{C}^{M\times N\times W}$ as:

$$K(r+lc,u+\tilde{s}q-lh_a,w) = \tilde{\Upsilon}_{r,\tilde{s}}(u,l+wq) \tag{4.15}$$

6. Finally, the result is given by a DFT of K along the first dimension:

$$c(m,n,w) = \mathcal{F}_M\left(K(\cdot,n,w)\right)(m). \tag{4.16}$$

this paper we have chosen to derive the algorithm by directly manipulating the sums of the definition.

To find more efficient algorithms, the first step is to recognize that the summation and the modulation term in (4.1) can be expressed as a DFT:

$$c(m,n,w)=\sqrt{L}\mathcal{F}_L\left(f(\cdot,w)\overline{g(\cdot-an)}\right)(mb). \tag{4.17}$$

We can improve on this because we do not need all the coefficients computed by the Fourier transform appearing in (4.17), only every b'th coefficient. Therefore, we can rewrite by the Poisson summation formula (4.8):

$$\begin{aligned} c(m,n,w) &= \sqrt{M}\mathcal{F}_M\left(\sum_{\tilde{m}=0}^{b-1} f(\cdot+\tilde{m}M,w)\overline{g(\cdot+\tilde{m}M-an)}\right)(m) \\ &= \left(\mathcal{F}_M K(\cdot,n,w)\right)(m), \end{aligned} \tag{4.18}$$

where

$$K(j,n,w)=\sqrt{M}\sum_{\tilde{m}=0}^{b-1} f(j+\tilde{m}M,w)\,\overline{g}(j+\tilde{m}M-na), \tag{4.19}$$

for $j\in\langle M\rangle$ and $n\in\langle N\rangle$. From (4.18) it can be seen that computing the DGT of a signal f can be done by computing K followed by a DFT along the first dimension of K.

We split j as $j=r+lc$ with $r\in\langle c\rangle$, $l\in\langle q\rangle$ and introduce $h_a,h_M\in\mathbb{Z}$ such that the following is satisfied:

$$c=h_M M-h_a a. \tag{4.20}$$

The two integers h_a, h_M can be found by the extended Euclid algorithm for computing the GCD of a and M.

Using (4.20) and the splitting of j we can express (4.19) as

$$\begin{aligned} K(r+lc,n,w) &= \sqrt{M}\sum_{\tilde{m}=0}^{b-1} f(r+lc+\tilde{m}M,w)\times \\ &\quad \times\overline{g}(r+l(h_M M-h_a a)+\tilde{m}M-na) \qquad (4.21)\\ &= \sqrt{M}\sum_{\tilde{m}=0}^{b-1} f(r+lc+\tilde{m}M,w)\times \\ &\quad \times\overline{g}(r+(\tilde{m}+lh_m)M-(n+lh_a)a) \qquad (4.22) \end{aligned}$$

We set $\tilde{m}'=\tilde{m}+lh_m$ and $n'=n+lh_a$ and get

$$\begin{aligned} & K(r+lc,n'-lh_a,w) \\ &= \sqrt{M}\sum_{\tilde{m}'=0}^{b-1} f(r+lc+(\tilde{m}'-lh_m)M,w)\,\overline{g}(r+\tilde{m}'M-n'a) \qquad (4.23)\\ &= \sqrt{M}\sum_{\tilde{m}'=0}^{b-1} f(r+\tilde{m}'M+l(c-h_m M),w)\,\overline{g}(r+\tilde{m}'M-n'a) \qquad (4.24) \end{aligned}$$

For simplicity, we continue without the primes in (4.24). We split $\tilde{m} = k + \tilde{s}p$ with $k \in \langle p \rangle$ and $\tilde{s} \in \langle d \rangle$ and $n = u + sq$ with $u \in \langle q \rangle$ and $s \in \langle d \rangle$ and use that $M = cq$, $a = cp$ and $c - h_m M = -h_a a$:

$$\begin{aligned} & K(r + lc, u + sq - lh_a, w) \\ = & \sqrt{M} \sum_{k=0}^{p-1} \sum_{\tilde{s}=0}^{d-1} f(r + kM + \tilde{s}pM - lh_a a, w) \times \\ & \times \overline{g}(r + kM - ua + (\tilde{s} - s) pM) \end{aligned} \tag{4.25}$$

Define

$$\tilde{\Psi}^f_{r,\tilde{s}}(k, l + wq) = f(r + kM + \tilde{s}pM - lh_a a, w), \tag{4.26}$$

$$\tilde{\Phi}^g_{r,\tilde{s}}(k, u) = \sqrt{M} g(r + kM + \tilde{s}pM - ua), \tag{4.27}$$

We can then write (4.25) as

$$\begin{aligned} & K(r + lc, u + sq - lh_a, w) \\ = & \sum_{k=0}^{p-1} \sum_{\tilde{s}=0}^{d-1} \tilde{\Psi}^f_{r,\tilde{s}}(k, l + wq) \overline{\tilde{\Phi}^g_{r,\tilde{s}-s}(k, u)} \end{aligned} \tag{4.28}$$

The sum over $\tilde{s}$ can be seen as a special form of convolution. The combination (4.6) and (4.7) yields

$$(f * g^*)(l) = \sum_{k=0}^{L-1} f(k) \overline{g}(k - l) \tag{4.29}$$

$$\widehat{f * g^*} = \sqrt{L} \hat{f} \hat{\overline{g}}, \tag{4.30}$$

and it is the kind of convolution as in (4.29) the we shall use. Fully written out (4.30) is

$$(f * g^*)(l) = \sqrt{L} \mathcal{F}_L^{-1} \left(\hat{f}(\cdot) \hat{\overline{g}}(\cdot) \right)(l).$$

Define the Fourier transforms along $\tilde{s}$ of $\tilde{\Psi}$ and $\tilde{\Phi}$ by

$$\Psi^f_{r,s}(k, l) = \left(\mathcal{F}_d \tilde{\Psi}^f_{r,\cdot}(k, l) \right)(s) \tag{4.31}$$

$$\Phi^g_{r,s}(k, u) = \left(\mathcal{F}_d \tilde{\Phi}^g_{r,\cdot}(k, u) \right)(s) \tag{4.32}$$

Using (4.30) we can now write (4.28) as

$$\begin{aligned} & K(r + lc, u + \tilde{s}q - lh_a, w) \\ = & \sqrt{d} \sum_{k=0}^{p-1} \mathcal{F}_d^{-1} \left(\Psi^f_{r,\cdot}(k, l + wq) \overline{\Phi^g_{r,\cdot}(k, u)} \right)(\tilde{s}) \end{aligned} \tag{4.33}$$

$$= \sqrt{d} \mathcal{F}_d^{-1} \left(\sum_{k=0}^{p-1} \Psi^f_{r,\cdot}(k, l + wq) \overline{\Phi^g_{r,\cdot}(k, u)} \right)(\tilde{s}) \tag{4.34}$$

If we consider $\Psi^f_{r,s}$ and $\Phi^g_{r,s}$ as matrices for each r and s, then sum over k in the last line can be written as matrix products. Algorithm 2 follows from this.

4.5 Extensions

The algorithm just developed can also be used to calculate the synthesis operator D_γ. This is done by applying Algorithm 2 in the reverse order and inverting each step in the algorithm. All the steps can be trivially inverted except step 3, which becomes

$$\Psi^f_{r,s} = \left(\Phi^\gamma_{r,s}\right) \Upsilon_{r,s}. \tag{4.35}$$

If the matrices $\Phi^\gamma_{r,s}$ are all left-inverses of the matrices $\Phi^g_{r,s}$ then (4.35) will invert step 3 in Algorithm 2. This is the case if γ is a dual Gabor window of the Gabor frame (g, a, M). It also holds that all dual Gabor windows γ of a Gabor frame (g, a, M) must satisfy that $\Phi^\gamma_{r,s}$ are left-inverses of $\Phi^g_{r,s}$. This criterion was reported in [52, 54].

A special left-inverse in the *Moore-Penrose pseudo-inverse*. Taking the pseudo-inverses of $\Phi^g_{r,s}$ yields the factorization associated with the canonical dual window of (g, a, M), [21]. Taking the polar decomposition of each matrix in $\Phi^g_{r,s}$ yields a factorization of the canonical tight window (g, a, M). For more information on these methods, as well as iterative methods for computing the canonical dual/tight windows, see [57].

4.6 Special cases

We shall consider some special cases of the algorithm:

1. Integer oversampling. When the redundancy is an integer then $p = 1$. Because of this we see that $c = a$ and $d = b$. This gives (4.20) the appearance

   $$a = h_M q a - h_a a,$$

   indicating that $h_M = 0$ and $h_a = -1$ solves the equation for all a and q. The algorithm simplifies accordingly, and reduces to the well known Zak-transform algorithm for this case, [49].

2. Short time Fourier transform. In this case $a = b = 1$, $M = N = L$, $c = d = 1$, $p = 1$, $q = L$ and as in the previous special case $h_M = 0$ and $h_a = -1$. In this case the algorithm reduces to the very simple, and well known algorithm, for computing the STFT.

4.7 Efficient implementation

The reason for defining the algorithm on multisignals, is that the multiple signals can be handled at once in the matrix product in step 3. This is a matrix product of two matrices size $q \times p$ and $p \times qW$, so the second matrix grows when multiple signals are involved. Doing it this way reuse the $\Phi^g_{r,s}$ matrices as much as possible, and this is an advantage on standard, general purpose computers with a deep memory hierarchy, see [27, 108].

The are several ways to execute Algorithm 2. One is of course two execute the steps in the order as they are written. Another possibility comes from the fact that step 1 - 5 can be done in parallel over the variable r. This may be exploited as follows:

1. The algorithm can be split such that each processor on a parallel machine handles step 1-5 for a specific range of values for r.

2. One may loop over step 1-5 for each value of r. This lowers the memory requirement because only a subset of the matrices $\Psi_{r,s}^{f}$ and $\Phi_{r,s}^{g}$ needs to be kept in memory at once.

3. One may do the algorithm as it is written, doing the loop over r as the innermost loop. This make the memory access more efficient, because the values indexed by r are stored consecutively in memory. On typical computing machinery, elements stored in consecutive memory locations can be loaded much faster than scattered elements.

Any combination of the above three methods can be done, depending on the number of processors and memory that is available on the machine.

Implementations of the algorithms described in this paper can be found in the Linear Time Frequency Toolbox (LTFAT) available from `http://www.univie.ac.at/nuhag-php/ltfat/`.

Chapter 5

Iterative algorithms to approximate canonical Gabor windows: Computational aspects

This chapter is the paper [57] which has been published online by Journal of Fourier Analysis and Applications. This is joint work with A.J.E.M Janssen.

5.1 Abstract

In this paper we investigate the computational aspects of some recently proposed iterative methods for approximating the canonical tight and canonical dual window of a Gabor frame (g, a, b). The iterations start with the window g while the iteration steps comprise the window g, the k^{th} iterand γ_k, the frame operators S and S_k corresponding to (g, a, b) and (γ_k, a, b), respectively, and a number of scalars. The structure of the iteration step of the method is determined by the envisaged convergence order m of the method. We consider two strategies for scaling the terms in the iteration step: norm scaling, where in each step the windows are normalized, and initial scaling where we only scale in the very beginning. Norm scaling leads to fast, but conditionally convergent methods, while initial scaling leads to unconditionally convergent methods, but with possibly suboptimal convergence constants. The iterations, initially formulated for time-continuous Gabor systems, are considered and tested in a discrete setting in which one passes to the appropriately sampled-and-periodized windows and frame operators. Furthermore, they are compared with respect to accuracy and efficiency with other methods to approximate canonical windows associated with Gabor frames.

5.2 Introduction

We consider in this paper iterative schemes for the approximation of the canonical tight and canonical dual windows associated with a Gabor frame. There are several motivations for this study:

- Fast algorithms for computing dual/tight windows allow for more flexibility in choosing the windows. Instead of working with fixed, precomputed windows, fast algor ithms allow for changing the windows on the fly. This can lead to more robust applications, that be tter adapt to a larger variety of problems.

- When designing Gabor windows meeting an optimality criterion, it is often necessary to generate sequences of windows, and then speed is important.

We refer to [23, Ch. 8-10] and [41, Ch. 5-9, 11-13] for recent and comprehensive treatments of the theory of Gabor systems and frames; to fix notations and conventions we briefly give here the main features. We denote for $g \in L^2(\mathbb{R})$ and $a > 0$, $b > 0$ by (g, a, b) the collection of time-frequency shifted windows

$$g_{na,mb}, \quad m, n \in \mathbb{Z}, \tag{5.1}$$

where for $x, y \in \mathbb{R}$ we denote

$$g_{x,y} = e^{2\pi i y t} g(t - x), \quad t \in \mathbb{R}. \tag{5.2}$$

We refer to g as the *window* and to a and b as the *time-shift* and *frequency-shift* parameters, respectively, of the Gabor system (g, a, b). When there are $A > 0$, $B < \infty$, called the *lower* and *upper frame bound*, respectively, such that for all $f \in L^2(\mathbb{R})$ there holds

$$A \|f\|^2 \leq \sum_{m,n=-\infty}^{\infty} |(f, g_{na,mb})|^2 \leq B \|f\|^2, \tag{5.3}$$

we call (g, a, b) a *Gabor frame*. When in (5.3) the second inequality holds for all $f \in L^2(\mathbb{R})$, we have that

$$f \in L^2(\mathbb{R}) \mapsto Sf := \sum_{m,n=-\infty}^{\infty} (f, g_{na,mb})\, g_{na,mb} \tag{5.4}$$

is well-defined as a bounded, positive, semi-definite linear operator of $L^2(\mathbb{R})$. We call S the *frame operator* of (g, a, b). The frame operator commutes with all relevant shift operators, i.e., we have for all $f \in L^2(\mathbb{R})$

$$Sf_{na,mb} = (Sf)_{na,mb}, \quad m, n \in \mathbb{Z}. \tag{5.5}$$

We shall assume in the remainder of this paper that (g, a, b) is a Gabor frame. Thus the frame operator S is positive definite and therefore boundedly invertible. There are two windows canonically associated to the Gabor frame (g, a, b). These are the canonical tight window g^t and the canonical dual window g^d, defined by

$$g^t = S^{-1/2} g \quad , \quad g^d = S^{-1} g, \tag{5.6}$$

respectively. The practical relevance of these windows is that they give rise to Gabor series representations of arbitrary $f \in L^2(\mathbb{R})$ according to

$$f = \sum_{m,n=-\infty}^{\infty} (f, g^t_{na,mb})\, g^t_{na,mb} = \sum_{m,n=-\infty}^{\infty} (f, g^d_{na,mb})\, g_{na,mb}, \tag{5.7}$$

where both series are $L^2(\mathbb{R})$-convergent. Furthermore, the Gabor systems (g^t, a, b) and $\left(g^d, a, b\right)$ are Gabor frames themselves with frame operators equal to the identity I and S^{-1}, respectively.

The computation of g^t and g^d according to (5.6) requires taking the inverse square root and the inverse of the frame operator S, respectively. In the often occurring practical case that ab is a rational number, the frame operator is highly structured which allows relatively efficient methods for computing S^{-1}, see [86]. The computation of $S^{-\frac{1}{2}}$ is much more awkward, even in the case that ab is rational, and often requires advanced techniques from numerical linear algebra, see for instance [45].

In [58] the calculus of Gabor frame operators was combined with a use of the spectral mapping theorem and Kantorovich's inequality to analyze an iteration scheme for the approximation of g^t that was proposed around 1995 by Feichtinger and Strohmer (independently of one another). In this iteration scheme one sets $\gamma_0 = g$ and for $k = 0, 1, \ldots$

I. $$\gamma_{k+1} = \frac{1}{2}\alpha_k\gamma_k + \frac{1}{2}\beta_k S_k^{-1}\gamma_k; \quad \alpha_k = \frac{1}{\|\gamma_k\|},\ \beta_k = \frac{1}{\|S_k^{-1}\gamma_k\|}, \tag{5.8}$$

where S_k is the frame operator corresponding to (γ_k, a, b). It was shown in [58] that (γ_k, a, b) is indeed a Gabor frame, and that $\frac{\gamma_k}{\|\gamma_k\|}$ converges (at least) quadratically to $\frac{g^t}{\|g^t\|}$. From the numerical results in [58] for the case that g is the standard Gaussian window $2^{1/4}\exp(-\pi t^2)$ and $a = b = 1/\sqrt{2}$ it appears that the resulting method compares favorably with other iterative techniques for computing inverse square roots, [64, 66, 91].

The investigations in [58] were followed by the introduction in [55, 56] of two families of iterative algorithms for the approximation of g^t and g^d, in which the iteration step involves the initial window g and the frame operator S as well as the current window γ_k and frame operator S_k, but frame operator inversion as in (5.8) do not occur. The following instances of these two families were analyzed in [55, 56]. Again we set $\gamma_0 = g$, and for $k = 0, 1, \ldots$

II. $$\gamma_{k+1} = \frac{3}{2}\alpha_k\gamma_k - \frac{1}{2}\beta_k S_k\gamma_k; \quad \alpha_k = \frac{1}{\|\gamma_k\|},\ \beta_k = \frac{1}{\|S_k\gamma_k\|}, \tag{5.9}$$

III. $$\begin{aligned}\gamma_{k+1} &= \frac{15}{8}\varepsilon_{k0}\gamma_k - \frac{5}{4}\varepsilon_{k1}S_k\gamma_k + \frac{3}{8}\varepsilon_{k2}S_k^2\gamma_k;\\ \varepsilon_{k0} &= \frac{1}{\|\gamma_k\|},\ \varepsilon_{k1} = \frac{1}{\|S_k\gamma_k\|},\ \varepsilon_{k2} = \frac{1}{\|S_k^2\gamma_k\|},\end{aligned} \tag{5.10}$$

for the approximation of g^t, and

IV. $$\gamma_{k+1} = 2\alpha_k\gamma_k - \beta_k S_k g; \quad \alpha_k = \frac{1}{\|\gamma_k\|},\ \beta_k = \frac{1}{\|S_k g\|}, \tag{5.11}$$

V. $$\begin{aligned}\gamma_{k+1} &= 3\delta_{k0}\gamma_k - 3\delta_{k1}S_k g + \delta_{k2}SS_k\gamma_k;\\ \delta_{k0} &= \frac{1}{\|\gamma_k\|},\ \delta_{k1} = \frac{1}{\|S_k g\|},\ \delta_{k2} = \frac{1}{\|SS_k\gamma_k\|},\end{aligned} \tag{5.12}$$

for the approximation of g^d.

The algorithms II-V are, contrary to algorithm I, conditionally convergent in the sense that the frame bound ratio $\frac{A}{B}$ of the initial Gabor frame (g, a, b) should exceed a certain lower bound. Accordingly, in algorithm II and III we have that $\frac{\gamma_k}{\|\gamma_k\|}$ converges to $\frac{g^t}{\|g^t\|}$ quadratically and cubically when $\frac{A}{B} > \frac{1}{2}$ and $\frac{A}{B} > \frac{3}{7}$, respectively. In algorithm IV and V we have that $\frac{\gamma_k}{\|\gamma_k\|}$ converges to $\frac{g^d}{\|g^d\|}$ quadratically and cubically when $\frac{A}{B} > \frac{1}{2}\left(\sqrt{5}-1\right)$ and $\frac{A}{B} > 0.513829766\ldots$, respectively. A remarkable phenomenon that emerged from the preliminary experiments done with the algorithms around 2002, was the fact that the lower bounds for the algorithms II, III seem far too pessimistic while those for the algorithms IV and V appear to be realistic.

In the algorithms just presented, all computed windows are normalized. We shall refer to this as *norm scaling.* Another possibility that we will investigate, is to replace all scalars α's, β's, ε's and δ's that occur in (5.9-5.12) by 1, and then initially scale the windows by replacing

$$g \text{ by } g/\hat{B}^{1/2} \quad , \quad S \text{ by } S/\hat{B}. \tag{5.13}$$

We shall refer to this scaling strategy as *initial scaling.* If $\hat{B}$ is (an estimate for) the best upper frame bound $\max \sigma(S)$ then the algorithms will be unconditionally convergent, with guaranteed desired convergence order, but with convergence constants that may not be as good as the ones that can be obtained by using the norm scaling as described by (5.9-5.12).

Matrix versions of algorithms II-III without any scaling have been treated in [11]. In [46, 63] the matrix version of algorithm II is considered using norm scaling and a scaling method that approximates the optimal scaling. The matrix version of algorithm IV is known as a Schulz iteration, see [90]. The fact that S, and therefore $\varphi(S)$ with φ continuous and positive on the spectrum of S, commutes with all relevant shift operators, allows us to formulate the iteration steps on the level of the windows themselves.

In this paper we investigate the algorithms II-V, using both norm and initial scaling, with more emphasis on computational aspects than in [58, 55, 56]. Here it is necessary to consider sampled-and-periodized Gabor systems in the style of [53]. This allows for a formulation and analysis of the algorithms I-V in an entirely similar way as was done in [58, 55, 56]. Thanks to the fact that the involved (canonical) windows and frame operators behave so conveniently under the operations of sampling and periodization, the observations done on the sampled-and-periodized systems are directly relevant to the time-continuous systems. We must restrict here to rational values of ab, and this gives the frame operator additional structure which can be exploited in the computations as dictated by the recursion steps, also see [5, 86, 92] for this matter.

The notions "smart" (or, rather, "smart but risky") and "safe" (or, rather, "safe but conservative") were introduced in a casual way in [56] to distinguish between cases where the stationary point(s) of the function transforming (frame) operators according to (5.22), (5.29) has a good chance to be well-placed in the middle of and on the "safe" side of the relevant spectral set, respectively. In the present paper, we choose to refer to the strategies leading to smart and safe modes as "norm scaling" and "initial scaling", respectively, and discard the terms "smart" and "safe" altogether.

5.3 Paper outline and results

In Section 5.4 and 5.5 we present the basic results of [55, 56] on transforming g into $\gamma = \varphi(S)\, g$, where φ is a function positive and continuous on $\sigma(S)$, so as to obtain a window γ whose frame operator S_γ (for approximating g^t) or the operator $(SS_\gamma)^{1/2} =: Z_\gamma$ (for approximating g^d) is closer to (a multiple of) the identity I than S itself. Here we recall that (g^t, a, b) has frame operator I and that $\left(g^d, a, b\right)$ has frame operator S^{-1}. Thus we relate the operators S and S_γ and their frame bounds, and we present a bound for the distance between (the normed) γ and g^t in terms of the frame bounds of S_γ. Similarly, we relate the frame bounds of (g, a, b) and the minimum and maximum of $\sigma(Z_\gamma)$, and we present a bound for the difference between (the normed) γ and g^d in terms of the latter minimum and maximum. Next, in Section 5.5, the choice of φ is specified so as to accommodate the recursions of type II, III and of type IV, V which gives us a means to monitor the frame bound ratio $\frac{A_k}{B_k}$ (for g^t) and of the ratio between minimum and maximum of $\sigma(Z_k)$ (for g^d) during the iteration process.

In Section 5.6 we present the algorithms using only initial scaling. In Section 5.7 we elaborate on the observation that all algorithms take place in the closed linear span $\mathcal{L}_g$ of the adjoint orbit $\left\{g_{j/b,l/a} \,\middle|\, j, l \in \mathbb{Z}\right\}$. Here the dual lattice representation of frame operators is relevant as well as an operator norm to measure the distance of S_k (for g^t) and of $(SS_k)^{1/2}$ (for g^d) from (a multiple of) the identity. The consideration of the algorithms in the space $\mathcal{L}_g$ reveals a fundamental difference between the algorithms for computing g^t and g^d that manifests itself in the totally different after-convergence behaviour of the two families of algorithms.

In Section 5.8 we give some considerations in the Zak transform domain, so as to produce examples of Gabor frames for which a specific algorithm diverges.

In Section 5.9 we discuss the discretization and finitization aspects (through sampling and periodization) that have to be taken into account since the algorithms are to be tested numerically.

In Section 5.10 we show how the algorithms can be expressed for discrete, finite Gabor systems, and show that the algorithms are scalar iterations of the singular values of certain matrices. We present an efficient implementation of the iterative algorithms, and we list the window functions we have used to test the algorithms.

In Section 5.11 we present our experimental results, compare them with what the theory predicts and with other methods to compute tight and dual windows. We provide examples that show the quadratic and cubic convergence of the algorithms, and the exponential divergence of the dual iterations after the initial convergence. We give an example that breaks the norm scaling schemes for both the tight and dual iterations, and show how various error norms of the iteration step behave. Comparisons with other methods are made: we show that the tight iterations are competitive with respect to computing time and superior with respect to precision. Finally, we show that the number of iterations needed for full convergence of the algorithms are dependent on the frame bound ratio, but independent of the structural properties of the discretization. For initial scaling, we show that it is easy to choose a scaling parameter that gives almost optimal convergence.

5.4 Frame operator calculus and basic inequalities

The basic theory to analyze the recursions appears somewhat scattered in [58, 55, 56]; for the reader's convenience, we give in Section 5.4 and 5.5 a concise yet comprehensive summary of the basic results and ideas. We let (g, a, b) be a Gabor frame with frame operator S and best frame bounds $A = \min \sigma(S) > 0$, $B = \max \sigma(S)$, where $\sigma(S)$ is the spectrum of S. In this section we present the basic inequalities expressing the approximation errors in terms of the (frame) bounds on the involved (frame) operators. These inequalities are a consequence of the calculus of Gabor frame operators, the spectral mapping theorem and Kantorovich's inequality.

Proposition 5.4.1. *Let φ be continuous and positive on $[A, B]$, and set $\gamma := \varphi(S) g$. The following holds.*

(i) *(γ, a, b) is a Gabor frame with frame operator $S_\gamma := S\varphi^2(S)$ and best frame bounds*

$$A_\gamma := \min_{s\in\sigma(S)} s\varphi^2(s) \quad , \quad B_\gamma := \max_{s\in\sigma(S)} s\varphi^2(s). \tag{5.14}$$

Furthermore,

$$g^t = S^{-1/2} g = S_\gamma^{-1/2}\gamma = \gamma^t, \tag{5.15}$$

and

$$\left\| \frac{\gamma}{\|\gamma\|} - \frac{g^t}{\|g^t\|} \right\| \le \left(1 - Q_\gamma^{1/4}\right)\sqrt{\frac{2}{1+Q_\gamma}}; \quad Q_\gamma = \frac{A_\gamma}{B_\gamma}. \tag{5.16}$$

(ii) *Let $Z_\gamma := (SS_\gamma)^{1/2} = S\varphi(S)$, and*

$$E_\gamma := \min \sigma(Z_\gamma) = \min_{s\in\sigma(S)} s\varphi(s), \tag{5.17}$$

$$F_\gamma := \max \sigma(Z_\gamma) = \max_{s\in\sigma(S)} s\varphi(s). \tag{5.18}$$

Then

$$g^d = Z_\gamma^{-1}\gamma \quad , \quad Z_\gamma\gamma = S_\gamma g, \tag{5.19}$$

and

$$\left\| \frac{\gamma}{\|\gamma\|} - \frac{g^d}{\|g^d\|} \right\| \le \left(1 - R_\gamma^{1/2}\right)\sqrt{\frac{2}{1+R_\gamma}}; \quad R_\gamma = \frac{E_\gamma}{F_\gamma}. \tag{5.20}$$

The basic result (i) gives us a clue how to produce a good approximation $\frac{\gamma}{\|\gamma\|}$ of $\frac{g^t}{\|g^t\|}$: take φ such that $s\varphi^2(s)$ is flat on $\sigma(S) \subset [A, B]$ so that the number Q_γ in (5.16) is close to 1. Similarly, by the basic result (ii), the number R_γ in (5.20) is close to 1 when φ is such that $s\varphi(s)$ is flat on $\sigma(S) \subset [A, B]$, and then we obtain a good approximation of $\frac{g^d}{\|g^d\|}$. In the next two subsections, we use this basic result repeatedly with polynomials φ of fixed degree m so as to obtain iterative approximations of g^t and g^d.

5.5 Norm scaling

5.5.1 Iterations for approximating g^t

We consider iteration schemes

$$\gamma_0 \;=\; g; \quad \gamma_{k+1} = \varphi_k\left(S_k\right)\gamma_k, \quad k = 0, 1, \ldots, \tag{5.21}$$

for the approximation of g^t, where S_k is the frame operator of (γ_k, a, b). We use here the basic result (i) repeatedly with

$$g \;=\; \gamma_k, \quad S = S_k \text{ and } \gamma = \gamma_{k+1}, \quad S_\gamma = S_{k+1} = S_k\varphi_k^2\left(S_k\right). \tag{5.22}$$

For $k = 0, 1, \ldots$ we have that $\gamma_k^t = g^t$ and that

$$\left\|\frac{\gamma_k}{\|\gamma_k\|} - \frac{g^t}{\|g^t\|}\right\| \le \left(1 - Q_k^{1/4}\right)\sqrt{\frac{2}{1+Q_k}}; \quad Q_k = \frac{A_k}{B_k}, \tag{5.23}$$

where A_k and B_k are the best frame bounds of S_k. The numbers A_k, B_k can be computed and estimated recursively according to $A_0 = A$, $B_0 = B$ and

$$A_{k+1} \;=\; \min_{s\in\sigma(S_k)} s\varphi_k^2\left(s\right) \ge \min_{s\in[A_k,B_k]} s\varphi_k^2\left(s\right), \tag{5.24}$$

$$B_{k+1} \;=\; \max_{s\in\sigma(S_k)} s\varphi_k^2\left(s\right) \le \max_{s\in[A_k,B_k]} s\varphi_k^2\left(s\right), \tag{5.25}$$

for $k = 0, 1, \ldots$.

We should choose φ_k such that $s\varphi_k^2\left(s\right)$ is flat on $\sigma\left(S_k\right)$. To that end there is proposed in [56, Subsec. 5.1] for $m = 2, 3, \ldots$ the choice

$$\varphi_k\left(s\right) = \sum_{j=0}^{m-1} a_{mj}\alpha_{kj}s^j, \quad \alpha_{kj} = \left\|S_k^j\gamma_k\right\|^{-1}, \tag{5.26}$$

where the a_{mj} are defined by

$$\sum_{l=0}^{m-1}(-1)^l\begin{pmatrix} -1/2 \\ l \end{pmatrix}(1-x)^l = \sum_{j=0}^{m-1} a_{mj}x^j, \quad x > 0. \tag{5.27}$$

The motivation for this choice is as follows. The left-hand side of (5.27) is the $(m-1)^{\text{th}}$ order Taylor approximation of $x^{-1/2}$ around $x = 1$, while (when Q_k is sufficiently close to 1)

$$\alpha_{kj}S_k^j \;\approx\; \left(\frac{S_k}{\|S_k\|}\right)^j\frac{1}{\|\gamma_k\|}, \quad j = 0, \ldots, m-1. \tag{5.28}$$

Hence $s\varphi_k^2\left(s\right) = \left(s^{1/2}\varphi_k\left(s\right)\right)^2$ should be expected to be flat on $\sigma\left(S_k\right)$, with $1 - Q_{k+1}$ potentially of order $\left(1-Q_k\right)^m$.

When $m = 2, 3$ we get the iterations II, III in (5.9), (5.10). It is shown in [55, Sec. 4] and [56, Sec. 6] that for $m = 2$ the quantity $Q_k = \frac{A_k}{B_k}$ increases to 1 and that $\frac{\gamma_k}{\|\gamma_k\|} \to \frac{g^t}{\|g^t\|}$ quadratically when $k \to \infty$, provided that $\frac{A}{B} > \frac{1}{2}$. For $m = 3$ it is shown in [55, Sec. 8] that Q_k increases to 1 and that $\frac{\gamma_k}{\|\gamma_k\|} \to \frac{g^t}{\|g^t\|}$ cubically when $k \to \infty$, provided that $\frac{A}{B} > \frac{3}{7}$. In [55, 56] it was observed for $m = 2, 3$ that the choice of α_{kj} causes $s\varphi_k^2\left(s\right)$ to have one or more stationary points in $[A_k, B_k]$ so that the odds for flatness of $s\varphi_k^2\left(s\right)$ on $[A_k, B_k]$ are favourable.

5.5.2 Iterations for approximating g^d

We consider iteration schemes

$$\gamma_0 \;=\; g; \quad \gamma_{k+1} = \varphi_k(Z_k)\,\gamma_k, \quad k = 0, 1, \ldots, \tag{5.29}$$

for the approximation of g^d, where $Z_k = (SS_k)^{1/2}$ with S_k the frame operator of (γ_k, a, b). It is seen from (5.29) by induction that $\gamma_k = \psi_k(S)\,g$ for some function ψ_k. Hence by the basic result (i) with $\varphi = \psi_k$, we have that

$$S_k = S\psi_k^2(S) \quad , \quad Z_k = S\psi_k(S), \tag{5.30}$$

and, by the basic result (ii), that

$$\left\| \frac{\gamma_k}{\|\gamma_k\|} - \frac{g^d}{\|g^d\|} \right\| \le \left(1 - R_k^{1/2}\right) \sqrt{\frac{2}{1 + R_k}}; \quad R_k = \frac{E_k}{E_k}, \tag{5.31}$$

where $E_k = \min \sigma(Z_k)$, $F_k = \max \sigma(Z_k)$. A further use of the calculus of frame operators as given by the basic result (i), yields

$$Z_{k+1} = Z_k \varphi_k(Z_k). \tag{5.32}$$

Consequently, the numbers E_k, F_k can be computed and estimated recursively according to $E_0 = A$, $F_0 = B$ and

$$E_{k+1} \;=\; \min_{z\in\sigma(Z_k)} z\varphi_k(z) \ge \min_{z\in[E_k,F_k]} z\varphi_k(z), \tag{5.33}$$

$$F_{k+1} \;=\; \max_{z\in\sigma(Z_k)} z\varphi_k(z) \le \max_{z\in[E_k,F_k]} z\varphi_k(z), \tag{5.34}$$

for $k = 0, 1, \ldots$.

We should choose φ_k such that $z\varphi_k(z)$ is flat on $\sigma(Z_k)$. To that end there is proposed in [56, Subsec. 5.2] for $m = 2, 3, \ldots$ the choice

$$\varphi_k(z) = \sum_{j=0}^{m-1} b_{mj}\beta_{kj} z^j, \quad \beta_{kj} = \left\| Z_k^j \gamma_k \right\|^{-1}, \tag{5.35}$$

where the b_{mj} are defined by

$$\sum_{l=0}^{m-1} (1 - x)^l = \sum_{j=0}^{m-1} b_{mj} x^j, \quad x > 0. \tag{5.36}$$

The motivation for the proposal is similar to the one for the choice of φ_k in (5.26) in Subsec. 5.5.1; we now note that the left-hand side of (5.36) is the $(m-1)^{\text{th}}$ order Taylor approximation of x^{-1} around $x = 1$. The implementation of the resulting recurrence step

$$\gamma_{k+1} \;=\; \sum_{j=0}^{m-1} b_{mj} \frac{Z_k^j \gamma_k}{\left\| Z_k^j \gamma_k \right\|}, \quad Z_k = (SS_k)^{1/2}, \tag{5.37}$$

is made feasible by the observation that, thanks to the second item in (5.19), $Z_k\gamma_k = S_k g$ so that

$$Z_k^{2r}\gamma_k \;=\; (SS_k)^r\,\gamma_k, \quad Z_k^{2r+1}\gamma_k = (SS_k)^r\,S_k g, \quad r = 0,1,\ldots. \tag{5.38}$$

When $m = 2,3$ we get the recurrences IV, V in (5.11) and (5.12). It is shown in [55, Sec. 5] and [56, Sec. 7], that for $m = 2$ the quantity $R_k = \frac{E_k}{F_k}$ increases to 1 and that $\frac{\gamma_k}{\|\gamma_k\|} \to \frac{g^d}{\|g^d\|}$ quadratically when $k \to \infty$, provided that $\frac{A}{B} > \frac{1}{2}\left(\sqrt{5}-1\right)$. For $m = 3$ it is shown in [55, Sec. 9] that R_k increases to 1 and that $\frac{\gamma_k}{\|\gamma_k\|} \to \frac{g^d}{\|g^d\|}$ cubically when $k \to \infty$, provided that $\frac{A}{B} > 0.513829766\ldots$. In [55, 56] it was observed for $m = 2,3$ that the choice of β_{kj} causes $z\varphi_k(z)$ to have one or more stationary points in $[E_k, F_k]$.

5.6 Initial scaling

The algorithms II-V are guaranteed to converge when the lower bound ratio $\frac{A}{B}$ of (g,a,b) exceeds a certain value. The proofs, as given in [55] and [56], require a careful analysis of the extreme values of the functions φ_k on the spectra of the relevant operators and can become quite complicated, especially in the cases of algorithms III, V. However, the algorithms are efficient in the sense that the envisaged convergence order m is realized with favourable convergence constants. In practice, as the experiments in Section 5.11 show, the algorithms II, III turn out to converge in almost all cases, even when the frame bound ratio is close to 0. However, divergence of the algorithms IV, V occurs much more frequently. In Section 5.8 we present examples, using the Zak transform, of frames (g,a,b) such that algorithm II and IV diverges.

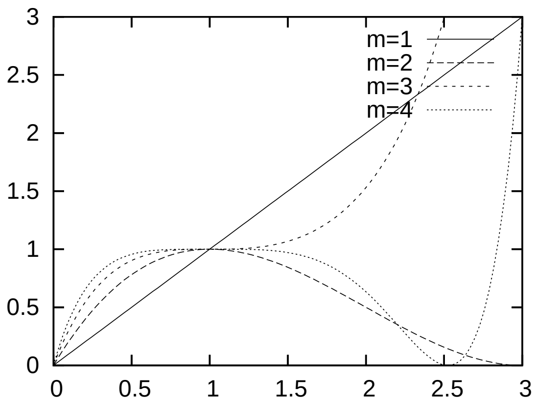
It would be desirable to have versions of the algorithms that are guaranteed to converge, no matter how small the frame bound ratio of the initial frame is (as long as it is positive). In the following, we present the initial scaling versions of the algorithms that converge at the envisaged convergence order m, possibly with suboptimal convergence constants. Since we can freely switch scaling strategy, a possible strategy is to initially scale such that convergence is guaranteed, and to continue until one is confident that the relevant condition number exceeds the specific lower bound so that the norm scaling mode can be applied from that point onwards.

The introduction in [56] of the notion of "safe modes" was prompted by an observation by M. Hampejs who prescaled the window g (and the frame operator) and deleted all normalization operations in the recursion step of algorithms II, IV. In the present paper, the prescaling is done in such a way that the scaled S has its spectrum exclusively in the attraction region of the function φ describing the simplified recursion. More specifically, we consider the iteration steps as given in Subsections 5.5.1, 5.5.2, with all α's and β's equal to 1. The φ's thus obtained are independent of k and are given by

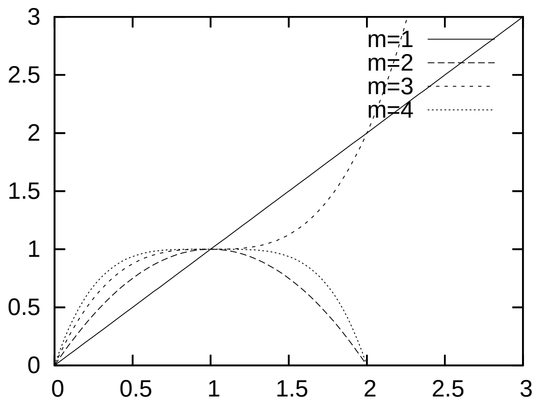
$$\varphi_m^t(s) \;:=\; \sum_{l=0}^{m-1}(-1)^l\begin{pmatrix}-1/2\\ l\end{pmatrix}(1-s)^l = \sum_{j=0}^{m-1} a_{mj}s^j, \quad s>0, \tag{5.39}$$

and

$$\varphi_m^d(z) \;:=\; \sum_{l=0}^{m-1}(1-z)^l = \sum_{j=0}^{m-1} b_{mj}z^j, \quad z>0, \tag{5.40}$$



(a) $s > 0 \mapsto s\left(\varphi_m^t(s)\right)^2$.



(b) $z > 0 \mapsto z\varphi_m^d(z)$.

Figure 5.1: The figure shows the two set of functions governing the convergence of the tight and dual iterations using initial scaling for order $m = 1, 2, 3, 4$ with φ_m^t and φ_m^d defined by (5.39) and (5.40). For (a), the attraction point 1 has attraction regions $m = 2 :\ (0, 3)$, $m = 3 :\ (0, 7/3)$, $m = 4 :\ (0, 2.525847988)$. For (b), the attraction point 1 has attraction region $(0, 2)$ for $m = 2, 3, 4$.

respectively. The relevant spectra transform by the spectral mapping theorem according to

$$\sigma(S_k) \quad \to \quad \left\{ s\left(\varphi_m^t(s)\right)^2 \,\middle|\, s \in \sigma(S_k) \right\} = \sigma(S_{k+1}), \tag{5.41}$$

and

$$\sigma(Z_k) \quad \to \quad \left\{ z\varphi_m^d(z) \,\middle|\, z \in \sigma(Z_k) \right\} = \sigma(Z_{k+1}), \tag{5.42}$$

respectively.

The functions φ_m^t and φ_m^d are $(m-1)^{\text{th}}$ order Taylor approximations of $s^{-1/2}$ and z^{-1} around $s = 1$ and $z = 1$, respectively. Hence $s\left(\varphi_m^t(s)\right)^2$ and $z\varphi_m^d(z)$ approximate 1 around $s = 1$ and $z = 1$, respectively. In Fig. 5.1 we have shown plots of the mappings

$$s > 0 \mapsto s\left(\varphi_m^t(s)\right)^2 \quad , \quad z > 0 \mapsto z\varphi_m^d(z) \tag{5.43}$$

for $m = 1, 2, 3, 4$, respectively. Fig. 5.1(a) also appears in [11]. In all cases, the point $s = 1 = z$ is an attractor for the region $(0, 2)$. Consequently, when $S_0 = S$, $Z_0 = S$ have spectrum in $(0, 2)$, the spectra $\sigma(S_k)$, $\sigma(Z_k)$ converge to 1 as $k \to \infty$, and the convergence is of order m in the sense that the ratio of minimum and maximum of the spectra converge to 1 at order m. Thus we should replace g by $g/\left(\hat{B}\right)^{1/2}$ and S by $S/\hat{B}$ where $\hat{B}$ is such that $\sigma\left(S/\hat{B}\right) \subset (0, 2)$ to obtain iterations having m^{th} order convergence to g^t and to $\left(\hat{B}\right)^{1/2} g^d$, respectively.

To guarantee convergence an estimate of $\max \sigma(S)$ is needed. In [5] a number of upper bounds of $\max \sigma(S)$ are developed for discrete-time, periodic Gabor systems. A

Table 5.1:

Method.	$\hat{B}$ or $\hat{F}$
I.	$\sqrt{AB}$
II.	$\frac{1}{3}\left(A+\sqrt{AB}+B\right)$
III.	$\frac{3}{10}(B+A)+\frac{2}{5}\sqrt{\frac{1}{2}(B^2+A^2)+\frac{1}{16}(B-A)^2}$
IV.	$\frac{1}{2}(E+F)$
V.	$\frac{1}{3}(F+E)+\frac{1}{3}\sqrt{\frac{1}{2}(F^2+E^2)+\frac{1}{2}(F-E)^2}$

The table shows the optimal scaling constant, $\hat{B}$ or $\hat{F}$, for doing initial scaling of the five iteration types.

convenient upper bound for our purposes follows from the dual lattice representation of the frame operator S, see Section 5.7 for more details, as

$$\max\sigma(S) \leq \frac{1}{ab}\sum_{j,l}\left|\left(g, g_{j/b,l/a}\right)\right|. \tag{5.44}$$

We make some comments for scaling optimally in the first iteration step. We shall refer to this method as *initial optimal scaling.* Assume that $\sigma(S)$ consists of the entire interval $[A,B]$. Consider the tight iterations as described in this Section, and assume that we replace S by $S/\hat{B}$. Then

$$A_1 = \min\left\{s\left(\varphi_m^t(s)\right)^2 \middle| s\in\left[A/\hat{B}, B/\hat{B}\right]\right\}, \tag{5.45}$$

$$B_1 = \max\left\{s\left(\varphi_m^t(s)\right)^2 \middle| s\in\left[A/\hat{B}, B/\hat{B}\right]\right\}. \tag{5.46}$$

Initial optimal scaling occurs for that value of $\hat{B}$ for which the ratio A_1/B_1 is maximal. The optimal value of the scaling parameter $\hat{F}$ for the dual iterations is defined in a similar way. For the five iteration types, the optimal $\hat{B}$ or $\hat{F}$ is shown in Table 5.1. All these numbers are close to the center of the interval $[A,B]$ or $[E,F]$. It is not hard to show that all optimally scaled operators S and Z have their spectra in the attraction regions given in Fig. 5.1 for the algorithms II-V.

If we scale optimally in each iteration step, and not just the first, we get the best possible convergence constants. However, this method is not practically feasible because of the repeated calculations of frame bounds, and we shall use it only as a reference method. We refer to it as *constant optimal scaling*, see Fig. 5.4.

5.7 Considerations in the adjoint orbit space

We consider the closed linear span $\mathcal{L}_g$ of the adjoint orbit $\left\{g_{j/b,l/a} \,\middle|\, j,l\in\mathbb{Z}\right\}$. According to the duality principle of Gabor analysis we have that the adjoint orbit is a Riesz basis for $\mathcal{L}_g$ (for this matter we refer to [23, Secs. 3.6 and 9.2], and [41, Ch. 7]). Furthermore, when $f \in L^2(\mathbb{R})$, the orthogonal projection of f onto $\mathcal{L}_g$ is given by

$$\begin{aligned} P_g f &= \frac{1}{ab} \sum_{j,l} \left(f, \left(g^d\right)_{j/b,l/a}\right) g_{j/b,l/a} \\ &= \frac{1}{ab} \sum_{j,l} \left(f, \left(g^t\right)_{j/b,l/a}\right) \left(g^t\right)_{j/b,l/a}. \end{aligned} \tag{5.47}$$

As a consequence of $\gamma_k^t = g^t$ in all our algorithms, we see that $\gamma_k \in \mathcal{L}_g$. There is also the Wexler-Raz biorthogonality relation,

$$\left(g, \left(g^d\right)_{j/b,l/a}\right) = \left(g^t, \left(g^t\right)_{j/b,l/a}\right) = ab\delta_{j0}\delta_{l0}, \tag{5.48}$$

where δ is Kronecker's delta. Finally, there is the following fundamental identity of Gabor analysis. Assume that $f, \xi, \gamma, h \in L^2(\mathbb{R})$ and that the three Gabor systems (f, a, b), (ξ, a, b), (γ, a, b) have finite upper frame bounds. Then we have

$$\sum_{m,n} \left(f, \gamma_{na,mb}\right) \left(\xi_{na,mb}, h\right) = \frac{1}{ab} \sum_{j,l} \left(\xi, \gamma_{j/b,l/a}\right) \left(f_{j/b,l/a}, h\right) \tag{5.49}$$

with absolute convergence at either side. We can regard (5.49) as a representation result for the frame-type operator

$$S_{\gamma,\xi} \ : \ f \to \sum_{n,m} \left(f, \gamma_{na,mb}\right) \xi_{na,mb}, \tag{5.50}$$

viz. as

$$S_{\gamma,\xi} = \frac{1}{ab} \sum_{j,l} \left(\xi, \gamma_{j/b,l/a}\right) U_{j,l}, \tag{5.51}$$

where $U_{j,l}$ is the unitary operator

$$U_{j,l} \ : \ f \to f_{j/b,l/a}. \tag{5.52}$$

This is the dual lattice representation (also known as the Janssen representation, see [23, Sec. 7.2] and [41, Corr. 9.3.7]) of the frame operator. In order for (5.51) to be well-defined, we assume that γ, ξ satisfies the so-called *Condition A'*:

A':

$$\sum_{j,l} \left|\left(\xi, \gamma_{j/b,l/a}\right)\right| < \infty, \tag{5.53}$$

see [41, Def. 7.2.1]. If $\xi = \gamma$ then this is the Condition A introduced by Tolimieri and Orr in [98]. We refer to Appendix 5.12 where an instance, relevant in the present context, of a pair ξ, γ satisfying condition A' is given.

5.7.1 Estimate for upper frame bound

If g satisfies Condition A, the frame operator S of (g,a,b) has the representation

$$S \;=\; \frac{1}{ab}\sum_{j,l}\left(g, g_{j/b,l/a}\right)U_{j,l}, \tag{5.54}$$

with absolute convergence in the operator norm. Therefore, there is the upper bound

$$\hat{B} \;=\; \frac{1}{ab}\sum_{j,l}\left|\left(g, g_{j/b,l/a}\right)\right| \tag{5.55}$$

for the best upper frame bound $\max\sigma(S)$ of (g,a,b).

5.7.2 Error measure

We measure convergence of γ_k to g^t and g^d by inspecting L^2-distances of the normed windows. This quantity is bounded in terms of the numbers Q_k and R_k in (5.16) and (5.20) that measure how close the operators S_k and Z_k are to being a multiple of the identity operator. In the converse direction, it would be useful to have a measure on the windows that translates directly to the distance of S_k and Z_k to (a multiple of) the identity operator. Such a measure can indeed be found. As to g^t we note that when g satisfies Condition A then S_k has the representation

$$S_k \;=\; \frac{1}{ab}\sum_{j,l}\left(\gamma_k, \left(\gamma_k\right)_{j/b,l/a}\right)U_{j,l}, \tag{5.56}$$

whence

$$\left\|S_k - \frac{1}{ab}\left\|\gamma_k\right\|^2 I\right\| \;\le\; \frac{1}{ab}\sum_{j,l\neq 0,0}\left|\left(\gamma_k, \left(\gamma_k\right)_{j/b,l/a}\right)\right|. \tag{5.57}$$

As to g^d we note that $Z_k = S\varphi_k(S)$ and with $\gamma_k = \varphi_k(S)\,g$ there holds by frame operator calculus

$$S\varphi_k(S)\,f \;=\; \sum_{m,n}\left(f, \left(\gamma_k\right)_{na,mb}\right)g_{na,mb}. \tag{5.58}$$

Hence there is the representation

$$Z_k \;=\; S\varphi_k(S) = \frac{1}{ab}\sum_{j,l}\left(g, \left(\gamma_k\right)_{j/b,l/a}\right)U_{j,l}. \tag{5.59}$$

Therefore

$$\left\|Z_k - \frac{1}{ab}\left(g,\gamma_k\right)I\right\| \le \frac{1}{ab}\sum_{j,l\neq 0,0}\left|\left(g, \left(\gamma_k\right)_{j/b,l/a}\right)\right|. \tag{5.60}$$

Note that the quantities of the right-hand sides of (5.57) and (5.60) measure to what extent the Wexler-Raz condition (5.48) is violated. In Appendix 5.12 it is shown for $a \in \mathbb{N}$, $b^{-1} \in \mathbb{N}$ and g satisfying condition A that the γ_k occurring in (5.57) and the g, γ_k occurring in (5.60) satisfy condition A and A', respectively. We shall refer to the right hand sides of (5.57) and (5.60) as the *dual lattice norm*.

5.7.3 Influence of out-of-space components

In this subsection, we give some heuristic observations that may serve to explain the difference in after-convergence behaviour between the algorithms to compute tight windows and those to compute dual windows.

We have seen that all iterands γ_k of the algorithms are in $\mathcal{L}_g$. We briefly comment on the impact on the algorithms of γ_k having non-zero components orthogonal to $\mathcal{L}_g$ (one can think here of round-off errors generating these components). To that end we consider the algorithms II and IV (assuming appropriate scaling has been carried out), and we assume that they have converged to the extent that the operators S_k and Z_k agree within machine precision with the identity operator.

As for algorithm II we thus have that

$$\gamma_{k+1} \;=\; \frac{3}{2}\gamma_k - \frac{1}{2}S_k\gamma_k = \gamma_k \tag{5.61}$$

within machine precision. Hence, possible out-of-space components in γ_k are reproduced within machine precision. As a consequence, we should expect that the error stays at its converged level when the iteration is continued beyond the point where machine precision is reached.

Next we consider algorithm IV using initial scaling so that

$$\gamma_{k+1} \;=\; 2\gamma_k - S_k g. \tag{5.62}$$

The term $S_k g$ has the representation

$$S_k g \;=\; \frac{1}{ab}\sum_{j,l}\left(\gamma_k, (\gamma_k)_{j/b,l/a}\right) g_{j/b,l/a} \in \mathcal{L}_g. \tag{5.63}$$

Furthermore,

$$P_g\gamma_k \;=\; \frac{1}{ab}\sum_{j,l}\left(\gamma_k, (g^d)_{j/b,l/a}\right) g_{j/b,l/a}. \tag{5.64}$$

Hence, $S_k g = P_g\gamma_k \in \mathcal{L}g$ to machine precision, and, to machine precision,

$$P_g\gamma_{k+1} \;=\; \frac{1}{ab}\sum_{j,l}\left(\gamma_k, (\gamma_k)_{j/b,l/a}\right) g_{j/b,l/a} = P_g\gamma_k. \tag{5.65}$$

On the other hand the orthogonal component $\gamma_k - P_g\gamma_k$ is per (5.62) multiplied by 2, i.e., to machine precision,

$$\gamma_{k+1} - P_g\gamma_{k+1} \;=\; 2\left(\gamma_k - P_g\gamma_k\right). \tag{5.66}$$

As a consequence, the algorithm starts to diverge beyond the point where machine precision is reached.

The observations just made continue to hold for the more general algorithms in Subsections 5.5.1 and 5.5.2. Thus no substantial after-convergence error build-up occurs for the algorithms of Subsection 5.5.1. For the algorithms of Subsection 5.5.2, with basic recursion step

$$\gamma_{k+1} \;=\; \sum_{j=0}^{m-1} b_{mj} Z_k^j \gamma_k \tag{5.67}$$

the terms with odd j all lie in $\mathcal{L}_g$, and those with even j are given within machine precision by $b_{mj}\gamma_k$. Since $\sum_{j \text{ even}} b_{mj} = 2^{m-1}$, the out-of-space component in γ_k gets multiplied by 2^{m-1} in each iteration step.

5.8 Zak domain considerations

We consider the case that $ab = \frac{p}{q}$ with integer $p, q > 0$ such that $\gcd(p,q) = 1$, and we define the Zak transform Z as (the extension to $L^2(\mathbb{R})$ of) the mapping

$$h \;\rightarrow\; (Zh)(t,\nu) = b^{-1/2} \sum_{k=-\infty}^{\infty} h\left(\frac{t-k}{b}\right) e^{2\pi i k \nu}, \quad t, \nu \in \mathbb{R}. \tag{5.68}$$

We refer to [111] and to [54, Sec. 1.5], for more details on the Zak transform and its role in Gabor analysis.

For $f, h \in L^2(\mathbb{R})$ we set (when $t, \nu \in \mathbb{R}$)

$$\Phi^f(t,\nu) \;=\; p^{-1/2} \left((Zf)\left(t - l\frac{p}{q}, \nu + \frac{k}{p}\right) \right)_{k=0,\dots,p-1,\, l=0,\dots,q-1}, \tag{5.69}$$

and

$$A^{f,h}(t,\nu) \;=\; \left(A_{k,r}^{f,h}(t,\nu) \right)_{k,r=0,\dots,p-1} = \Phi^f(t,\nu) \left(\Phi^h(t,\nu) \right)^*, \tag{5.70}$$

where the * denotes conjugate transpose. Now (g, a, b) is a Gabor frame, with frame bounds $A > 0$, $B < \infty$ if and only if we have $AI_{p\times p} \leq A^{gg}(t,\nu) \leq BI_{p\times p}$ for almost all $t, \nu \in \mathbb{R}$, with A and B the largest and smallest positive real number for which the respective inequalities hold. The frame operator S of (g, a, b) is "represented" by A^{gg} through the formula

$$\Phi^{Sf} \;=\; A^{gg} \Phi^f, \quad f \in L^2(\mathbb{R}), \tag{5.71}$$

with matrix multiplication at each point $(t, \nu) \in \mathbb{R}$ on the right-hand side of (5.71). This formula extends as follows. Assume that φ is continuous and positive on $[A, B]$. Then

$$\Phi^{\varphi(S)f} \;=\; \varphi(A^{gg}) \Phi^f, \quad f \in L^2(\mathbb{R}), \tag{5.72}$$

which is the basic formula for functional calculus in the Zak transform domain.

We consider in this section the critical case $a = b = 1$ (in Sections 5.10 and 5.11 more general rational ab will be dealt with). Then considerable simplifications occur since all the matrices Φ, A reduce to scalars. The formula (5.72) then becomes

$$(Z(\varphi(S)f))(t,\nu) \;=\; \varphi\left(|(Zg)(t,\nu)|^2 \right) (Zf)(t,\nu), \quad t, \nu \in \mathbb{R}, \tag{5.73}$$

for $f \in L^2(\mathbb{R})$. In particular we have

$$\begin{aligned}\left(Zg^t\right)(t,\nu) &= \left(Z\left(S^{-1/2}g\right)\right)(t,\nu) = \frac{(Zg)(t,\nu)}{|(Zg)(t,\nu)|}, \quad t,\nu \in \mathbb{R}, && (5.74)\\ \left(Zg^d\right)(t,\nu) &= \left(Z\left(S^{-1}g\right)\right)(t,\nu) = \frac{(Zg)(t,\nu)}{|(Zg)(t,\nu)|^2} \\ &= \frac{1}{(Zg)^*(t,\nu)}, \quad t,\nu \in \mathbb{R}. && (5.75)\end{aligned}$$

To illustrate the relevance for the algorithms, we consider algorithms II and IV for all scaling strategies. As to initial scaling, we assume that g and S are scaled such that $(g, a=1, b=1)$ has best upper frame bound $B < 2$, which means that $|Zg|^2 < 2$ everywhere. We let

$$G = Zg \quad , \quad \Gamma_k = Z\gamma_k. \tag{5.76}$$

Then by functional calculus in the Zak transform domain, the algorithms II and IV (initial scaling) assume the form

$$\Gamma_0 = G \quad ; \quad \Gamma_{k+1} = \frac{3}{2}\Gamma_k - \frac{1}{2}|\Gamma_k|^2 \Gamma_k, \quad k = 0,1,\ldots, \tag{5.77}$$

and

$$\Gamma_0 = G \quad ; \quad \Gamma_{k+1} = 2\Gamma_k - |\Gamma_k|^2 G, \quad k = 0,1,\ldots, \tag{5.78}$$

respectively, where the relations in (5.77) and (5.78) are to be considered at each point $(t,\nu) \in \mathbb{R}$. These recursions are then quite easily analyzed by elementary means. For instance, one sees that the assumption $B < 3$ is necessary and sufficient for (5.77) to converge to $\exp(i \arg(G))$ everywhere, while the assumption $B < 2$ is necessary and sufficient for (5.78) to converge to $1/G^*$ everywhere. Unbounded recursions result when we would have allowed B to be larger than 5 and 2, respectively.

Next we consider algorithms II, IV using norm scaling so that (5.77) and (5.78) are to be replaced by

$$\Gamma_0 = G \quad ; \quad \Gamma_{k+1} = \frac{3}{2}\frac{\Gamma_k}{\|\Gamma_k\|} - \frac{1}{2}\frac{|\Gamma_k|^2\Gamma_k}{\|\Gamma_k^3\|}, \quad k = 0,1,\ldots, \tag{5.79}$$

and

$$\Gamma_0 = G \quad ; \quad \Gamma_{k+1} = 2\frac{\Gamma_k}{\|\Gamma_k\|} - \frac{|\Gamma_k|^2 G}{\|\Gamma_k^2 G\|}, \quad k = 0,1,\ldots. \tag{5.80}$$

The norms used here are $L^2\left([0,1)^2\right)$-norms. We consider the case that

$$Zg = 1 \text{ on } N \quad , \quad Zg = x > 0 \text{ on } M, \tag{5.81}$$

where N, M are two measurable sets $\subset [0,1)^2$ such that $N \cap M = \emptyset$, $N \cup M = [0,1)^2$. Then $(g, a=1, b=1)$ is a Gabor frame with best frame bounds $A = \min(1, x^2)$, $B = \max(1, x^2)$. Furthermore,

$$Zg^t = 1 \text{ on } [0,1)^2 \quad ; \quad Zg^d = \frac{1}{x} \text{ on } M. \tag{5.82}$$

We have for both algorithms II and IV that

$$\Gamma_k = c_k \text{ on } N \quad , \quad \Gamma_k = d_k \text{ on } M, \tag{5.83}$$

where c_k, d_k follow recursions that can be made completely explicit (using that, for instance, $\|\Gamma_k\| = \left((1-\varepsilon)\, c_k^2 + \varepsilon d_k^2\right)^{1/2}$, where $\varepsilon = \mu\,(M)$). Due to the norming operations in the recursion steps, either recursion stays bounded.

We consider the case that $\varepsilon = \mu\,(M) \ll 1$. Then an elementary analysis shows the following: there is a $\delta > 0$ such that for recursion (5.79), (5.83) there holds

- $x \in \left(0, \sqrt{3} - \delta\right) \Rightarrow c_k, d_k \to 1,$
- $x \in \left(\sqrt{3} + \delta, \sqrt{5} - \delta\right) \Rightarrow c_k \to 1, d_k \to -1$
- $x \in \left(\sqrt{5} + \delta, \infty\right) \Rightarrow$ chaotic behaviour.

Also, there is a $\delta > 0$ such that for recursion (5.80), (5.83) there holds

- $x \in \left(0, \sqrt{2} - \delta\right) \Rightarrow c_k \to 1, d_k \to \frac{1}{x},$
- $x \in \left(\sqrt{2} + \delta, \infty\right) \Rightarrow d_k < 0.$

5.9 Sampling and periodization of Gabor frames

The algorithms considered in this paper and in [58, 55, 56] have been formulated for time-continuous Gabor frames while the tests we perform must take place in a finite setting. The transition from continuous to discrete/finite Gabor frames by sampling and periodization has been discussed in [53] and later in [62, 92], see also [24, Subsec. 8.4] and [23, Secs. 10.2 and 10.3]. Let a, M be positive integers, and assume that $(g, a, 1/M)$ is a Gabor frame with frame bounds $A > 0$, $B < \infty$. Furthermore, assume that g satisfies the aforementioned condition A and the so-called condition R:

R:

$$\lim_{\varepsilon \to 0} \sum_{j=-\infty}^{\infty} \frac{1}{\varepsilon} \int_{-\varepsilon/2}^{\varepsilon/2} |g\,(j+u) - g\,(j)|^2\, du \quad = \quad 0. \tag{5.84}$$

The conditions R and A are not very restrictive; they are, for instance, satisfied by all members g of Feichtinger's algebra S_0, see [24, comment after Thm. 8.4.2]. Then the system

$$g^D_{na,m/M} \quad := \quad \left(g_{na,m/M}\,(r)\right)_{r \in \mathbb{Z}}, \quad n \in \mathbb{Z}, m = 0, \ldots, M-1 \tag{5.85}$$

is a discrete Gabor frame with frame bounds A, B, the dual window g^d of the frame $(g, a, 1/M)$ satisfies conditions R and A, and the dual window $\left(g^D\right)^d$ corresponding to the discrete Gabor system in (5.85) is obtained by sampling g^d:

$$\left(g^D\right)^d (r) \quad = \quad \left(g^d\right)^D (r), \quad r \in \mathbb{Z}. \tag{5.86}$$

The transition from discrete Gabor frames to discrete, periodic Gabor frames is just as convenient. Assume that we have a $g \in l^1(\mathbb{Z})$ such that the discrete Gabor system $\left(g_{na,m/M}\right)_{n\in\mathbb{Z},m=0,\ldots,M-1}$ is a discrete Gabor frame with frame bounds $A > 0$ $B < \infty$. Let $L = Na = Mb$ for some positive integers N, b, and define

$$g^P(r) \;=\; \sum_{j=-\infty}^{\infty} g(r - jL), \quad r \in \mathbb{Z}. \tag{5.87}$$

Then the system

$$\left(g^P_{an,m/M}(r)\right)_{r\in\mathbb{Z}}, \quad n = 0, \ldots, N-1, m = 0, \ldots, M-1, \tag{5.88}$$

is a discrete, periodic Gabor system with frame bounds A, B, the dual window g^d of the discrete Gabor system is in $l^1(\mathbb{Z})$, and the dual window $\left(g^P\right)^d$ corresponding to the discrete periodic Gabor system in (5.88) is obtained by periodizing g^d:

$$\left(g^P\right)^d(r) \;=\; \left(g^d\right)^P(r) = \sum_{j=-\infty}^{\infty} g^d(r - jL), \quad r \in \mathbb{Z}. \tag{5.89}$$

An important extension of these results is given in [24, Subsec. 8.4]. Assume that φ is analytic in an open neighbourhood containing $[A, B]$, where $A > 0$, $B < \infty$ are frame bounds of the Gabor frame $(g, a, 1/M)$ with g satisfying condition R and A. Then $\varphi(S)g$ satisfies R and A as well, and

$$(\varphi(S)g)^D \;=\; \varphi\left(S^D\right)g^D, \tag{5.90}$$

where S^D is the frame operator corresponding to the system in (5.85). The approach in [24, Subsec. 8.4] (which uses the Dunford representation of operators as well as theorems of the Wiener $1/f$-type) can be mimicked so as to generalize the transition result from discrete Gabor systems as above with $g \in l^1(\mathbb{Z})$ to discrete, periodic Gabor systems. Thus, with φ as above and $\left(g_{na,m/M}\right)_{n\in\mathbb{Z},m=0,\ldots,M-1}$ a discrete Gabor system with $g \in l^1(\mathbb{Z})$, frame bounds $A > 0$, $B < \infty$ and frame operator S, we have $\varphi(S)g \in l^1(\mathbb{Z})$ and

$$(\varphi(S)g)^P \;=\; \varphi\left(S^P\right)g^P, \tag{5.91}$$

where S^P is the frame operator of the system in (5.88). In particular, we see that the sampling-and-periodization approach is valid for the tight window g^t in which case we should consider $\varphi(s) = s^{-1/2}$.

It follows from the above results that the algorithms can be considered for discrete and for discrete, periodic Gabor frames. The findings for these systems are of direct relevance to the algorithms we have considered for the time-continuous case.

5.10 Implementational aspects

All implementations are done in the finite, discrete setting of Gabor frames. We denote for $g \in \mathbb{C}^L$ and $a, b \in \mathbb{N}$ by (g, a, b) the collection of time-frequency shifted windows

$$g_{na,mb}, \quad n \in \mathbb{Z}, m \in \mathbb{Z}, \tag{5.92}$$

where for $j, k \in \mathbb{Z}$ we denote

$$g_{j,k} = e^{2\pi ikl/L} g(l-j), \quad l = 0, \ldots L-1. \tag{5.93}$$

Note that it must hold that $L = Na = Mb$ for some $M, N \in \mathbb{N}$. Additionally, we define $c, d, p, q \in \mathbb{N}$ by

$$c = \gcd(a, M), \quad d = \gcd(b, N), \quad p = \frac{a}{c} = \frac{b}{d}, \quad q = \frac{M}{c} = \frac{N}{d}. \tag{5.94}$$

With these numbers, the density of the Gabor system can be written as $(ab)/L = p/q$, where p/q is an irreducible fraction. It holds that $L = cdpq$.

5.10.1 Matrix representation and the SVD

Let $O_g \in \mathbb{C}^{L \times MN}$ be the matrix representation of the synthesis operator of a Gabor frame so that

$$(O_g)_{l,m+nM} = g_{ma,nb}(l), \quad l = 0, \ldots, L-1, \tag{5.95}$$

for $m = 0, \ldots, M-1$, $n = 0, \ldots, N-1$. Hence O_g has the column vectors $g_{ma,nb}$. The matrix representation of the frame operator corresponding to (g, a, b) is then given as $O_g O_g^*$. Since (g, a, b) is a frame we have that O_g has full rank $L \leq MN$.

Assume that φ is continuous and positive on $\sigma(S)$. From

$$(\varphi(S)\, g)_{na,mb} = \varphi(S)\, g_{na,mb}, \tag{5.96}$$

we have that

$$O_{\varphi(S)g} = \varphi(S)\, O_g = \varphi\left(O_g O_g^*\right) O_g. \tag{5.97}$$

Furthermore, note that for the Frobenius norm $\|O_g\|_{\text{fro}}$ we have $\|O_g\|_{\text{fro}}^2 = MN\|g\|^2$, since all columns of O_g have norm $\|g\|$.

The iterations can be written in terms of the synthesis operator matrices as follows. Denote the synthesis operator matrix corresponding to the Gabor frame (γ_k, a, b) by Ω_k. Then we can write the iteration step for algorithm II with norm scaling as

$$\Omega_0 = O_g;\ \Omega_{k+1} = \frac{3}{2}\frac{\Omega_k}{\|\Omega_k\|_{\text{fro}}} - \frac{1}{2}\frac{(\Omega_k \Omega^*)\,\Omega_k}{\|(\Omega_k \Omega_k^*)\,\Omega_k\|_{\text{fro}}}, \quad k = 0, 1, \ldots. \tag{5.98}$$

We shall consider the *thin* SVD of the synthesis operator matrices. Thus we let $O_g = U\Sigma V^*$, where $U \in \mathbb{C}^{L\times L}$ is unitary (O_g has full rank), $\Sigma \in \mathbb{R}^{L\times L}$ is a diagonal matrix with positive diagonal elements and $V \in \mathbb{C}^{MN\times L}$ has orthonormal columns. With φ as above, we compute the thin SVD of $O_{\varphi(S)g}$ as

$$\begin{aligned} O_{\varphi(S)g} &= \varphi\left(O_g O_g^*\right) O_g = \varphi\left(U\Sigma^2 U^*\right) U\Sigma V^* \\ &= U\varphi\left(\Sigma^2\right) U^* U\Sigma V^* = U\Sigma\varphi\left(\Sigma^2\right) V^*. \end{aligned} \tag{5.99}$$

Here we have used that $\varphi\left(U\Sigma^2U^*\right) = U\varphi\left(\Sigma^2\right)U^*$, a basic fact in the functional calculus of matrices. The equation (5.99) shows that $O_{\varphi(S)g}$ has the same right and left singular vectors as O_g, and the singular values transform according to $\varphi \rightarrow \sigma\varphi^2(\sigma)$. As a consequence we have,

$$O_{g^t} = UV^* \quad , \quad O_{g^d} = U\Sigma^{-1}V^*, \tag{5.100}$$

for the synthesis operators corresponding to (g^t, a, b) and $\left(g^d, a, b\right)$, respectively, for which we should take $\varphi(s) = s^{-1/2}$ and $\varphi(s) = s^{-1}$ in (5.99). We thus see that we have obtained the matrices occurring in the polar decomposition of O_g and the Moore-Penrose pseudo-inverse of O_g.

A further observation is that $\|O_g\|_{\text{fro}}^2 = \sum_{j=1}^{j=L}\sigma_j^2$, where σ_j, $j = 0, \ldots, L-1$, are the singular values of O_g. Letting $\varphi_{k,j}$ be the singular values of Ω_k and using that $\Omega_k = U\Sigma_kV^*$, we can write the iteration step in (5.98) on the level of singular values as

$$\begin{aligned} \sigma_{k+1,j} &= \frac{3}{2}\alpha_k\sigma_{k,j} - \frac{1}{2}\beta_k\sigma_{k,j}^3, \\ \alpha_k &= \frac{1}{\sqrt{\sum_{j=1}^{j=L}\sigma_{k,j}^2}}, \ \beta_k = \frac{1}{\sqrt{\sum_{j=1}^{j=L}\sigma_{k,j}^6}}, \end{aligned} \tag{5.101}$$

where $j = 0, \ldots, L-1$, and $k = 0, 1, \ldots$.

5.10.2 Factorization of finite, discrete Gabor systems

Similar to the Zibulski-Zeevi representation of the Gabor frame operator in the continuous case, see [111, 112, 54], it is possible to compute the actions of the finite, discrete Gabor frame operator (and also the analysis and synthesis operators) very efficiently. Several equivalent methods exists using almost the same number of operations, but differing in the order. In [7] a finite, discrete version of the Zibulski-Zeevi representation is developed. Another method was developed in [83] and [96]. Unfortunately, [96] contains some errors, which have been corrected in [3]. In the following we shall present the Zak-transform method from [7].

For $h \in \mathbb{C}^L$ and $K \in \{0, \ldots, L-1\}$ such that $\frac{L}{K} \in \mathbb{N}$, we define the finite, discrete Zak transform Z_Kh by

$$\left(Z_Kh\right)(r,s) = \sqrt{\frac{K}{L}}\sum_{l=0}^{L/K-1} h(r - lK)\,e^{2\pi islK/L}, \quad r, s \in \mathbb{Z}. \tag{5.102}$$

The finite, discrete Zak transform is quasi-periodic in its first variable and periodic in the second,

$$\left(Z_Kh\right)\left(r + kK, s + l\frac{K}{L}\right) = e^{2\pi iksK/L}\left(Z_Kh\right)(r,s), \tag{5.103}$$

see [48] for more details. The values $\left(Z_Kh\right)(r,s)$ of a finite, discrete Zak-transform on the fundamental domain $r = 0, \ldots, K-1$, $s = 0, \ldots, K/L-1$ can be calculated efficiently by K FFT's of length K/L. To obtain values outside the fundamental domain, the quasi-periodicity relation (5.103) can be used.

We define the cd matrices $\Phi^f_{r,s}$ of size $p \times q$ and the $p \times p$ matrices $A^f_{r,s}$ by

$$\Phi^f_{r,s} = \sqrt{cdq}\left(\left(Z_a f\right)(r + kM, s + ld)\right)_{k=0,\dots,p-1;\, l=0,\dots,q-1} \quad , \tag{5.104}$$

where $r = 0, \dots, c-1$, $s = 0, \dots, d-1$ and

$$A^{f,h}_{r,s} = \left(A^{f,h}_{r,s}\right)_{k,l=0,\dots,p-1} = \Phi^f_{r,s}\left(\Phi^h_{r,s}\right)^*. \tag{5.105}$$

With these definitions is holds that the frame operator S of (g, a, b) is "represented" by A^{gg} through the formula

$$\Phi^{Sf} = A^{gg}\Phi^f, \quad f \in \mathbb{C}^L, \tag{5.106}$$

see [7].

With this efficient representation of the frame operator of a finite, discrete Gabor system, we may express the iterations schemes in the finite, discrete Zak domain. We let

$$G = \Phi^g \quad , \quad \Gamma_k = \Phi^{\gamma_k},\ A_k = A^{\gamma_k,\gamma_k} = \Gamma_k\left(\Gamma_k\right)^*. \tag{5.107}$$

By functional calculus, algorithm II in the finite, discrete Zak transform takes the form

$$\Gamma_0 = G \quad ; \quad \Gamma_{k+1} = \frac{3}{2}\Gamma_k - \frac{1}{2}A_k\Gamma_k, \quad k = 0, 1, \dots. \tag{5.108}$$

The expressions for the other iterations types are similar.

5.10.3 Other methods

We have considered two other methods of computing the canonical tight window utilizing the factorization (5.104). To calculate the factorization of the canonical tight window g^t,Φ^{g^t}, we use an eigenvalue decomposition of the factorization of the frame operator of (g, a, b): for each $r = 0, \dots, c-1$, $s = 0, \dots, d-1$ compute $U_{r,s}$, $D_{r,s}$ such that $A^{gg}_{r,s} = U_{r,s}D_{r,s}U^*_{r,s}$, where $U_{r,s}$ is unitary and $D_{r,s}$ is diagonal and set

$$\Phi^{g^t}_{r,s} = U_{r,s}D^{-1/2}_{r,s}U^*_{r,s}\Phi^g_{r,s}. \tag{5.109}$$

We shall refer to this method as the EIG method. The other method uses (5.100) applied to the matrices of the factorization: for each $r = 0, \dots, c-1$, $s = 0, \dots, d-1$ compute $U_{r,s}$, $D_{r,s}$, $V_{r,s}$ such that $\Phi^g_{r,s} = U_{r,s}D_{r,s}V^*_{r,s}$, where $U_{r,s}$ is unitary, $D_{r,s}$ is diagonal and $V_{r,s}$ has orthonormal columns. Then it follows from functional calculus in the Zak transform domain (pretty much as in (5.99); also see (5.72)) that

$$\Phi^{g^t}_{r,s} = U_{r,s}V^*_{r,s}. \tag{5.110}$$

We shall refer to this method as the SVD method.

For computing the canonical dual window we have considered simply inverting the matrices of the factorization of the frame operator:

$$\Phi^{g^d}_{r,s} = \left(A^{gg}_{r,s}\right)^{-1}\Phi^g_{r,s}.$$

We shall refer to this as the INV method.

Table 5.2:

Method:	Flop count per iteration:
I.	$16Lp + \frac{4}{3}cdp^3$.
II.	$16Lp$.
III.	$24Lp$.
IV.	$16Lp$.
V.	$24Lp + 8cdp^3$.
	Total flop count:
INV.	$16Lp + \frac{4}{3}cdp^3$.
EIG.	$24Lp + 14cdp^3$.
SVD.	$64Lp + 32cdp^3$.

This table shows the flop count of each of the considered methods. The flop count does not include the cost of the pre- and post-factorization. The application of an inverse matrix needed for the algorithms I and INV is done using a Cholesky factorization followed by two substitutions. An iteration step of V takes more flops to compute than an iteration step of III, because we need to compute the two terms $S_k g$ and $S_k \gamma_k$. The flop counts for EIG and SVD methods are only approximations, because eigenvalues and singular values can be calculated by many different methods with different flop counts, and because the process usually involves an iterative step, see [40].The term cdp^3 is less than or equal to Lp for a Gabor frame.

5.10.4 Implementational costs

The computation of Φ^g needs to be done before the iteration step. It can be computed using $5L \log_2 N$ flops. This transforms the initial window g into the finite, discrete Zak domain. All computations in this domain are then done by multiplication of $p \times q$ and $p \times p$ matrices. The transform Φ is unitary from $\mathbb{C}^L$ with Euclidean norm into $\mathbb{C}^{c \times d \times p \times q}$, also with Euclidean norm. This gives an easy way to calculate the norms needed for the norm scaling.

We count the number of real floating point operations needed, and assume that everything is done using complex arithmetics. The flop count for a single iteration step in the transform domain for each of the 5 algorithms can be seen in Table 5.2.

A quick comparison show that the iterative methods for computing the tight window are comparable in number of flops to the EIG and SVD methods, if the number of necessary iterations is not to big. For the inverse iterations, the situation is different: computing the inverse of the block matrices by a direct approach requires only slightly more flops than a single iteration step of algorithm IV, so an iterative method will always use more flops than the direct approach. However, there might be situations were it is not desirable to compute the inverse. For instance, if the initial window g has small support then the iteration steps can be performed by multiple passes through a filter bank.

5.10.5 Stopping criterion

Because of the guaranteed quadratic/cubic convergence of the algorithms, it is possible to devise a simple yet powerful stopping criterion: we consider the difference

$$\frac{\|\gamma_{k+1}-\gamma_k\|}{\|\gamma_{k+1}\|}. \tag{5.111}$$

When this difference is close to the machine precision *eps*, the considered algorithm has converged. This is a standard stopping criterion, but using it this way means that we have done exactly one iteration step too much. Therefore, we stop when (5.111) is less than $\sqrt{eps}$ and $\sqrt[3]{eps}$ for the algorithms having quadratic and cubic convergence, respectively.

5.10.6 Window functions

As the basic window functions we shall use the Gaussian $\varphi_w \in L^2(\mathbb{R})$ and the hyperbolic secant $\psi_w \in L^2(\mathbb{R})$ given by

$$\varphi_w(t) = \left(\frac{w}{2}\right)^{-1/4} e^{-\pi t^2/w}, \quad t \in \mathbb{R}, \tag{5.112}$$

$$\psi_w(t) = \sqrt{\frac{\pi}{2}} w^{-1/4} \operatorname{sech}\left(t\frac{\pi}{\sqrt{w}}\right), \quad t \in \mathbb{R}. \tag{5.113}$$

It holds that the Fourier transform of φ_w is $\varphi_{1/w}$ and similarly for ψ_w. As window functions for the testing of the iterative algorithms we shall use finite, discrete versions of these, obtained by the sampling-and-periodization process described in Sec. 5.9:

$$\varphi_w^D(l) = \left(\frac{wL}{2}\right)^{-1/4} \sum_{k\in\mathbb{Z}} e^{-\pi\left(l/\sqrt{L}-k\sqrt{L}\right)^2/w}, \tag{5.114}$$

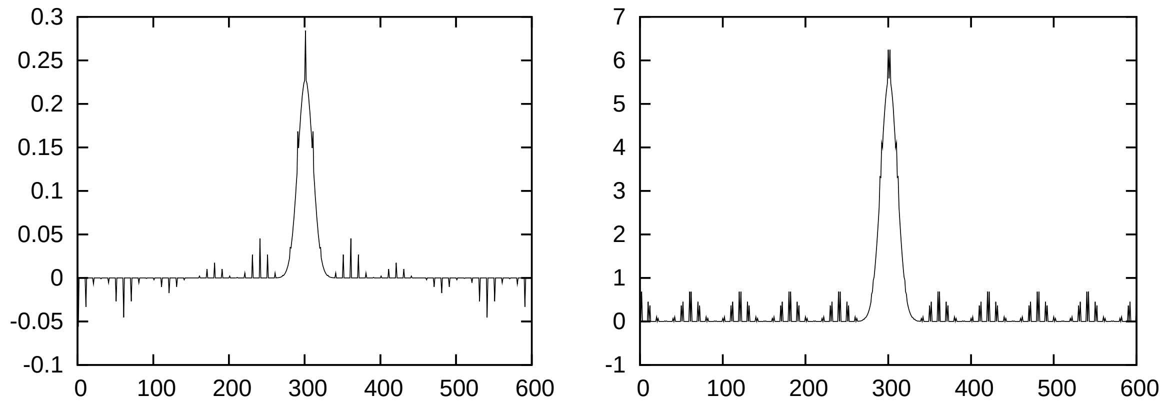
$$\psi_w^D(l) = \sqrt{\frac{\pi}{2}} (wL)^{-1/4} \sum_{k\in\mathbb{Z}} \operatorname{sech}\left(\left(\frac{l}{\sqrt{L}}-k\sqrt{L}\right)\frac{\pi}{\sqrt{w}}\right), \tag{5.115}$$

for $l = 0, \ldots, L-1$. The properties from the continuous setting carry over: the functions have almost unit norm, and the Discrete Fourier Transform of φ_w^D is $\varphi_{1/w}^D$ and similarly for ψ_w^D. For more details on the hyperbolic secant as a Gabor window, see [59].

To produce examples for which the norm scaling methods diverges, we have constructed a function (MONSTER) which is a Gaussian function modified in such a way that the first singular value, σ_{real}, of the matrix representation of the Gabor synthesis operator that corresponds to a real and symmetric singular vector, is given a large value. The function is shown on Fig. 5.3. This function is a generalization to the case of rational oversampling of the counterexample given in Sec. 5.8 and exploits the fact that the iterations can be considered as scalar iterations of the singular values of the Gabor synthesis operator, (5.101).

5.11 Experiments

This section contains the results from the experiments we have done in order to test the algorithms thoroughly and to demonstrate the various aspects of the algorithms that have



(a) MONSTER (b) DFT of MONSTER

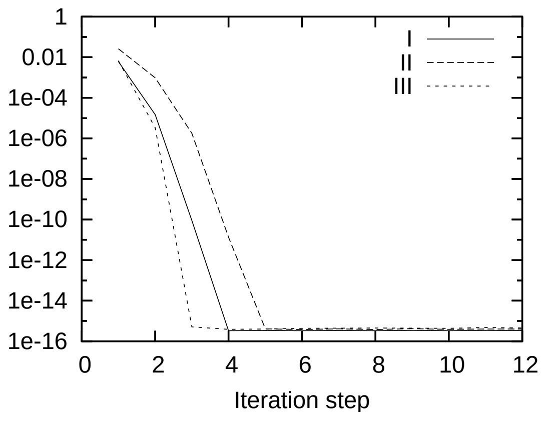
Figure 5.2: Fig. (a) shows the function MONSTER of length 600 with a single singular value set to $\sigma_{\text{real}} = 6$. Fig. (b) shows the Discrete Fourier Transform of the function.

been shown analytically. The computations have been done in Matlab and Octave, and the full source code is available for download from `http://www2.mat.dtu.dk/software/iteralg/`. We will show figures demonstrating the important aspects, but since we cannot include all material, the reader is encouraged to download the software and do experiments with it.

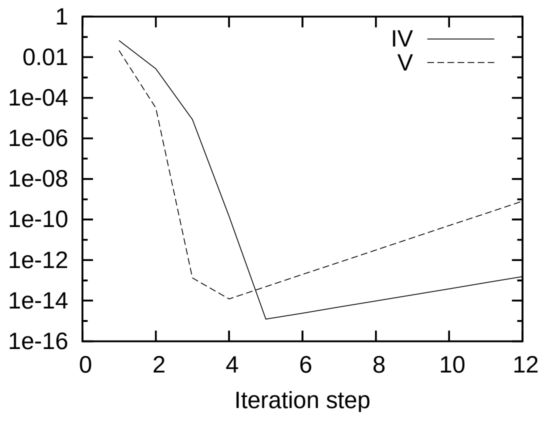
5.11.1 Convergence and divergence of norm scaling

Figure 5.3 shows the convergence behaviour for a well-conditioned problem. The figure shows that algorithm I,II and IV exhibit quadratic convergence, while III and V exhibit cubic convergence as proved in Subsections 5.5.1 and 5.5.2. Furthermore, the algorithms for computing the tight window stay converged close to the machine precision, while the algorithms for computing the dual window diverge. Algorithm V also diverges faster than IV. This is as proved in Subsection 5.7.3; the slopes of the two line segments beyond the 5th iteration in Fig. 5.3(b) corresponds to divergence factors 2 (for IV) and 4 (for V). A visible numerical aspect is that iteration V is not able to reach full precision, because the iterand is quickly affected by the buildup of numerical errors. The convergence behaviour of the algorithms for the initial window being a hyperbolic secant is almost the same.

Two examples of using different scaling strategies are shown in Fig. 5.4(a) and 5.4(b). The figures show that initially scaling by the best scaling constant is as good as using norm scaling, and using initial scaling by an easily computable scaling constant results in only 1-2 more iterations than using norm scaling. Comparing these methods to the method using optimal scaling, we see that for a well-conditioned problem the norm-scaling and optimal initial scaling are close to matching optimal convergence. For a worse conditioned problem (Fig. 5.4(b)), optimal scaling clearly outperforms the other methods. However, this observation has little practical relevance, because the computed canonical windows g^d and g^t will have a bad time-frequency localization. Higham [44] uses a scaling strategy for algorithm I that approximates the optimal scaling. This requires an estimate for the smallest eigenvalue of the matrix, but this is easy to obtain since the matrix is inverted

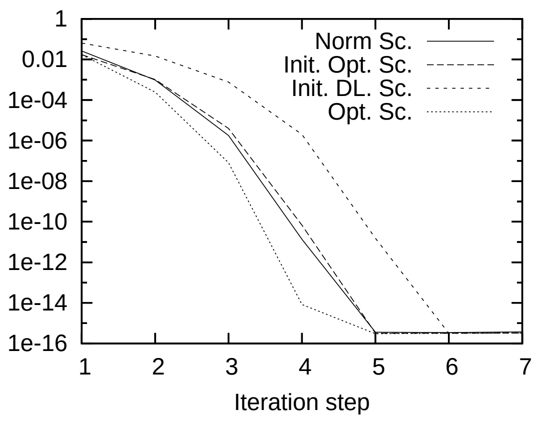


(a) Tight iterations.

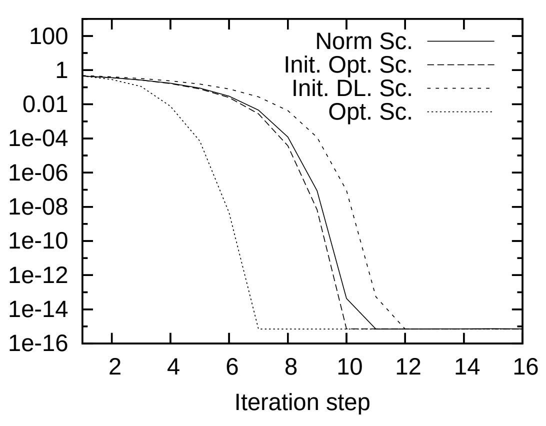


(b) Dual iterations.

Figure 5.3: The figure shows the behaviour of the 5 iteration types for the first 12 iterations of each. The y-axis shows the l^2-norm of the difference between the iteration step and a precomputed, normalized solution. The system considered in the Gabor frame for $\mathbb{C}^{432}$, $\left(\varphi_1^D, 18, 18\right)$. The Gabor frame has a frame bound ratio of $B/A = 2.03$.

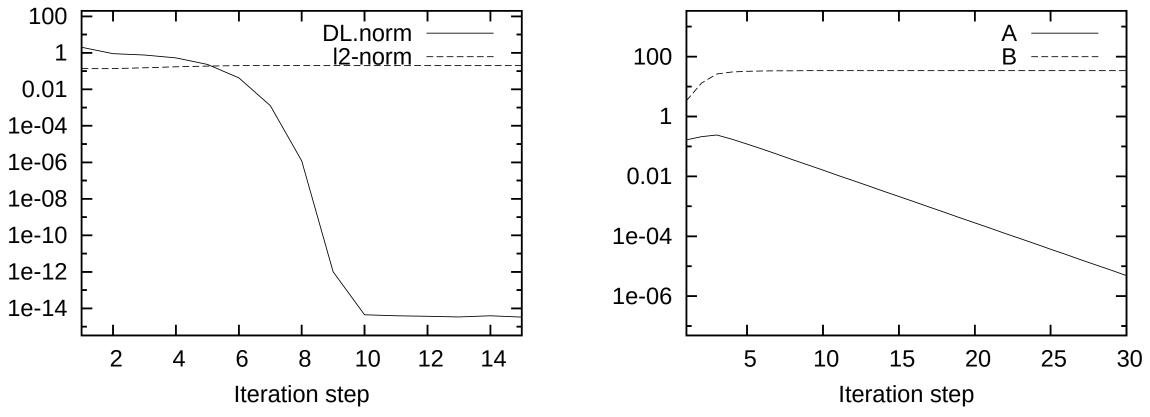


(a) Gaussian



(b) Badly scaled Gaussian

Figure 5.4: Fig. (a) shows the convergence behaviour for algorithm II using four different scaling strategies: norm scaling, initial scaling by the optimal constant, initial scaling by the dual lattice norm and constant optimal scaling. The system considered in the Gabor frame for $\mathbb{C}^{432}$, $\left(\varphi_1^D, 18, 18\right)$. Fig (b) shows the same, but instead using the window $\varphi_{1/5}^D$. This is a very narrow window, and the generated Gabor frame has a frame bound ratio of $B/A = 180.8$.



(a) II (b) IV

Figure 5.5: Fig. (a) shows the behaviour of the dual lattice norm and the l^2-norm of the difference between the iteration step and the normalized solution for a run of algorithm II using norm scaling. Fig. (b) shows the behaviour of the best upper and lower frame bound of Z_k in each iteration step for a run of algorithm IV using norm scaling. The system considered is in both cases the Gabor frame for $\mathbb{C}^{600}$ with $a = b = 20$ using the MONSTER function.

as part of the iteration step. For algorithms II-V we cannot use inversions, and so an estimate for the smallest eigenvalue (or lower frame bound) is difficult to obtain. We have therefore not pursued such a method for algorithms II-V.

The iterations for computing the tight window are very robust when using norm scaling. It is easy to create examples of Gabor systems with frame bound ratios $B/A > 10^{12}$ for which the iterations converge, by using badly dilated Gaussians or by using a constant function with a small amount of noise added. However, by using the *MONSTER* function, it is possible to create an example for which the norm scaling iterations diverge. The behaviour of the dual lattice norm and $\|g - \gamma_k\|_2$ in each iteration step for a run of algorithm II is shown in Fig. 5.5(a). It can be seen that the iteration converges to the wrong tight window. Another typical behaviour is that the iteration oscillates between two different functions with the same dual lattice norm. The behaviour of algorithm IV on the same examples is shown in Fig. 5.5(b), the figure displays the optimal frame bounds of Z_k for each iteration step. Here we see exponential convergence of the lower frame bound of Z_k to zero.

5.11.2 Comparison with other methods

Figure 5.6 shows a comparison of the numerical precision of the algorithms for computing the canonical tight window compared to the numerical precision of other standard methods. The stability of the tight iterations proved in Subsection 5.7.3 is clearly visible. The method based on computing eigenvalues deteriorates quickly as the frame bound ratio increases, while the SVD behaves much better. The eigenvalue method should not be used if the frame bound ratio of the problem is unknown. An explanation for this is that in the SVD method, the singular values are never considered, they are simply set equal to

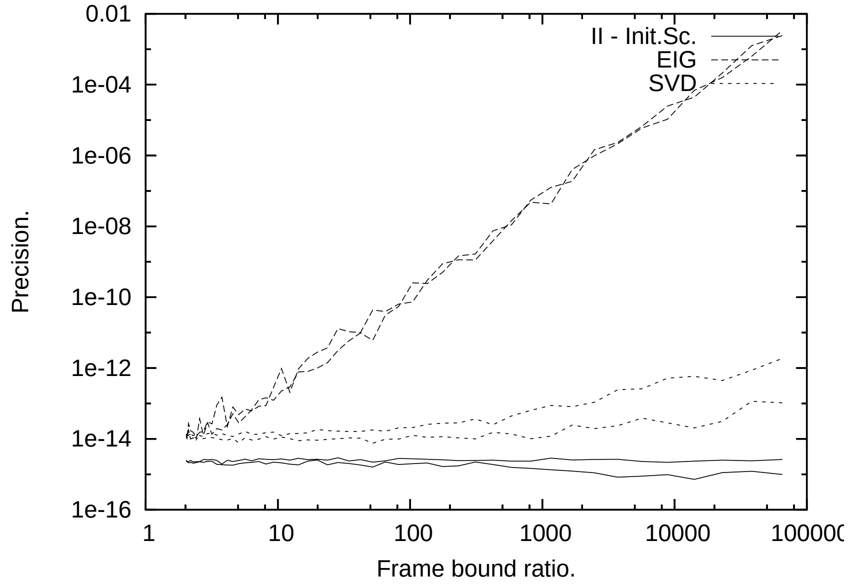
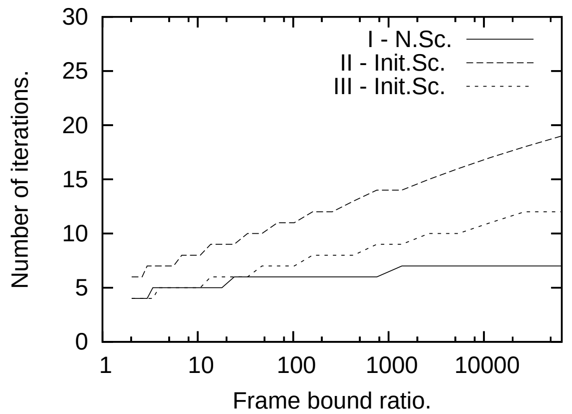


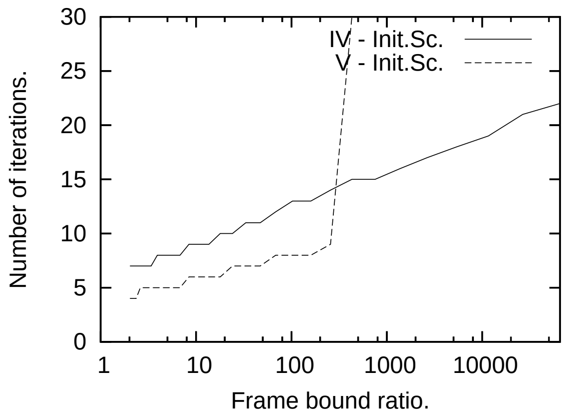
Figure 5.6: The figure shows the numerical accuracy of three different methods to compute the canonical tight window. The plot is parametric in w. For each method, one line corresponds to a narrow window, $w < 1$, the other corresponds to a wide window, $w > 1$. Almost overlapping points on different lines correspond to values w_1, w_2 such that $w_1 = 1/w_2$. The error measure used is the dual lattice norm. The Gabor frames used are the Gabor frames for $\mathbb{C}^{432}$, $\left(\varphi_w^D, 18, 18\right)$.

1. Therefore, roundoff errors on the small singular values do not affect the computation, in contrast to the EIG method, where round-off errors on the smallest eigenvalues are magnified because of the inversion of eigenvalues.[1]

The actual running time of the methods is determined by the flop count for each method (see the previous section for details) and of how fast the floating point operations can be executed by a computer. We will not give exact timings of the iterative algorithms, because we have not created optimal implementations of the algorithms, so timing them makes little sense. We note, however, that the key ingredients in the algorithms are FFTs of small length and matrix multiplications of small size matrices. Fast implementations exists for both algorithms, see [108, 39]. This makes it possible to create efficient implementations of the iterative algorithms. To present an idea of the speed of the algorithms, we have timed the computation of the canonical tight window of a Gaussian of length $L = 10800$ with $a = 120$ and $M = 160$ using algorithm II with norm scaling. On a normal (at the time of writing) laptop PC using Matlab this computation takes about 1 second.



(a) Tight iterations.



(b) Dual iterations.

Figure 5.7: The figure shows the number of iterations the algorithms need in order to reach machine precision. The three algorithms compared are I using norm scaling, and II-V using initial scaling by (5.55). The Gabor frames used are the Gabor frames for $\mathbb{C}^{432}$, $\left(\varphi_w^D, 18, 18\right)$.

5.11.3 Number of iterations

To study how the number of necessary iterations depends on the frame bound ratio of the initial Gabor frame, we have plotted the number of iterations for the algorithms to converge, as a function of the frame bound ratio of the Gabor frame. Figure 5.7 shows such a plot, using Gaussians to generate Gabor frames with varying frame bound ratios. The jumps in the curves occur when, according to the stopping criterion, an additional iteration step is necessary. Even though algorithm III has cubic convergence, it is almost never able to compete with algorithm I. The jump in the graph for algorithm V is due to the magnification of round-off errors dominating the convergence, and causing divergence. The same happens for algorithm IV, but for considerable worse frame bound ratios (not visible in the graph). The graphs for the hyperbolic secant look similar.

The number of necessary iterations might also depend on the size of the matrix blocks appearing in the factorization. This issue is slightly problematic to address, since creating a test problem involving bigger matrices also means altering the frame bound ratio. To minimize this effect, we have considered p, q running through the Fibonacci numbers, $2, 3, 5, 8, \ldots$, such that $p/q \rightarrow \left(\sqrt{5}-1\right)/2$. This creates a series of irreducible fractions p/q while keeping p/q close to a certain number away from 1. The result of the test is that the number of necessary iterations seems to be completely independent of the size of the matrix blocks! We have omitted the graphs, as they are simply horizontal lines. For algorithm I, it is proved in [63] that this is indeed the case.

[1]The slope of the graph for the EIG method in Fig. 5.6 is highly problem dependent. For a Gaussian window then $err \approx b(B/A)^a$ with $a \approx 2.67$ where err can be the dual lattice norm or the l^2-norm of the distance to a reference solution. For a sech window then $a \approx 1.1$. For more details on the stability of computing eigenvalues and singular values, see [1].

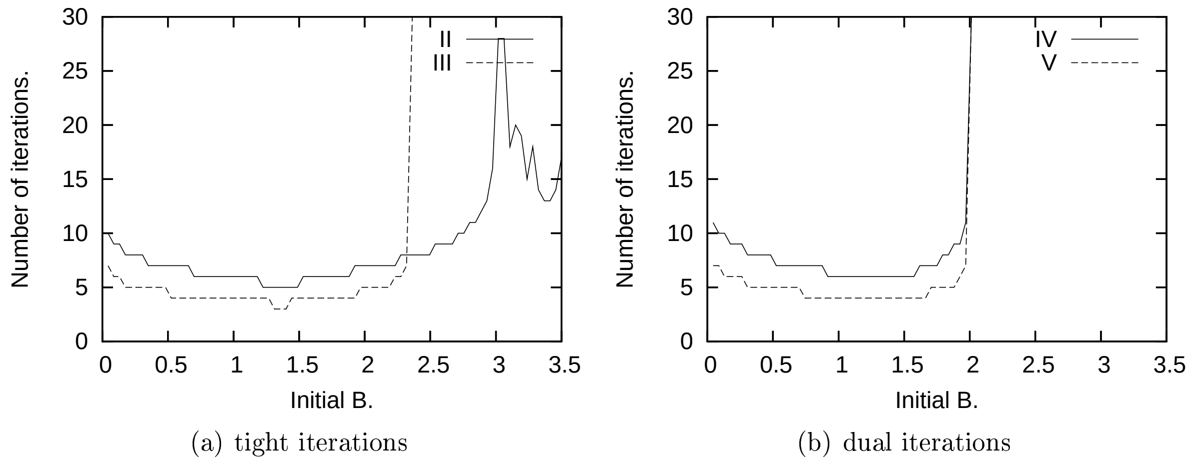


Figure 5.8: The figure shows the number of necessary iterations to find the tight/dual window of a Gabor frame for $\mathbb{C}^{432}$, $\left(\varphi_1^D, 18, 18\right)$, as a function of the best upper frame bounds of the initial window.

5.11.4 Choosing an initial scaling

Figure 5.1 shows that for each iteration type there is a range of values of the upper frame bound of the scaled window, B_{scaled}, that will guarantee convergence. Figure 5.8 shows an example of the effect of prescaling the input window to obtain specific values of B_{scaled}. As shown in Fig. 5.1, algorithm III diverges if B_{scaled} is larger than 7/3. The dual iterations IV and V diverge if $B_{scaled} > 2$ and II has a chaotic behaviour if $3 < B_{scaled} < 5$ and diverges if $B_{scaled} > 5$ (not shown in the plot).

The choice of $\hat{B}$ that minimizes the number of iterations is to choose $\hat{B}$ according to Table 5.1. An estimate for this is difficult to calculate, as it involves an estimate for the lower frame bound. Fortunately, as can be seen on Fig. 5.8 there is a large region around the optimal scaling point, where only 1 or 2 extra iterations are needed.

5.11.5 Summary from the numerical experiments

We sum up the main points of the numerical experiments.

1. The algorithms are simple to implement, because they do not require implementation of matrix inversion or the SVD.
2. The algorithms with quadratic convergence are usually faster than the ones with cubic convergence, because of the lower computational complexity per iteration step.
3. The iterative algorithms for computing canonical dual windows should only be used in under special circumstances, because they slowly diverge if too many iterations are done, and because they are usually slower than doing a direct inversion. However, they are the best choice if a direct inversion is not possible.
4. The iterative algorithms for computing canonical tight windows are extremely precise and faster than a direct method for well behaved cases.

5. Initial scaling by an easily computable bound provides methods that are both fast and guaranteed to converge.

5.12 A result on Condition A'

Proposition 5.12.1. *Assume that $(g, a, b = 1/M)$ is a Gabor frame that satisfies condition A, where $a, M \in \mathbb{N}$. Also assume that φ is analytic around $[A, B]$ and positive on $[A, B]$, where $A > 0$, $B < \infty$ are lower, upper frame bounds of (g, a, b). Finally, let $\gamma = \varphi(S)\, g$ where S is the frame operator corresponding to (g, a, b). Then g, γ satisfies the Condition A', i.e.,*

$$\sum_{j,l} \left| \left(g, \gamma_{j/b, l/a}\right) \right| < \infty. \tag{5.116}$$

Proof. We have for $f, h \in L^2(\mathbb{R})$ that

$$\begin{aligned} \sum_{n,m} (f, \gamma_{na,mb})\,(g_{na,mb}, h) &= \sum_{n,m} \left(f, (\varphi(S)\, g)_{na.mb}\right)(g_{na,mb}, h) \\ &= \sum_{n,m} (\varphi(S)\, f, g_{na,mb})\,(g_{na,mb}, h) \\ &= (S\varphi(S)\, f, h)\,. \end{aligned} \tag{5.117}$$

We know that (γ, a, b) is a Gabor frame. Now take $f, h \in L^2(\mathbb{R})$ such that (f, a, b) and (h, a, b) have finite upper frame bounds. Then by the fundamental identity of Gabor analysis, see [54, Subsecs. 1.4.1 and 1.4.2],

$$\sum_{n,m} (f, \gamma_{na,mb})\,(g_{na,mb}, h) = \frac{1}{ab} \sum_{j,l} \left(g, \gamma_{j/b,l/a}\right)\left(f_{j/b,l/a}, h\right), \tag{5.118}$$

with absolute convergence on either side of (5.118) . Thus $S\varphi(S)$ has the dual lattice representation

$$S\varphi(S) = \frac{1}{ab} \sum_{j,l} \left(g, \gamma_{j/lb,l/a}\right) U_{j,l}. \tag{5.119}$$

Now let $\psi(s) = s\varphi(s)$. This ψ is analytic around $[A, B]$ and positive on $[A, B]$. By functional calculus of frame operators in the time-frequency domain, see [24, Sec. 8.3], there holds that $S\varphi(S)$ has also the dual lattice representation

$$S\varphi(S) = \sum_{j,l} \left(\psi\left(\frac{1}{ab} HH^*\right)\right)_{0,0;j,l} U_{j,l}, \tag{5.120}$$

where H is the analysis operator with respect to the dual lattice, defined for $f \in L^2(\mathbb{R})$, by

$$Hf = \left(\left(f, g_{j/b,l/a}\right)\right)_{j,l \in \mathbb{Z}}. \tag{5.121}$$

It follows from the proof of [24, Thm. 4.3], in particular from uniform boundedness of (8.4.14) (with ψ instead of φ), that

$$\sum_{j,l} \left| \left(\psi \left(\frac{1}{ab} HH^* \right) \right)_{0,0;j,l} \right| \ < \ \infty. \tag{5.122}$$

By uniqueness of the coefficients in the dual lattice representation (just consider a well-behaved h such that $(h_{na,mb})_{n,m\in\mathbb{Z}}$ is a tight frame, i.e. such that $U_{j,l}h$, $j,l \in \mathbb{Z}$, is an orthogonal set of functions), it follows that $\sum_{j,l} \left|(g, \gamma_{j/b,l/a})\right| < \infty$, as required. $\square$

Note 5.12.2. It is implicit in the statement and proof of [24, Thm 4.3] that the γ of the above result is such that condition A is satisfied by (γ, a, b).

Acknowledgements

The cooperation leading to this paper began in Vienna, 2005, during the "Special Semester on Modern Methods of Time-Frequency Analysis", organized by H.G. Feichtinger and K. Gröchenig under the auspices of the Erwin Schrödinger Institute for Mathematical Physics (ESI). The authors express their thanks to the organizers and ESI.

Chapter 6

Algorithm XXX: The Linear Time Frequency Toolbox

This chapter is an early version of a paper written together with Bruno Torresani and Peter Balazs. It will be submitted to Transactions on Mathematical Software.

6.1 Abstract

The Linear Time Frequency Analysis Toolbox is a Matlab/Octave toolbox for doing time-frequency analysis. It is intended both as an educational and computational tool. The toolbox provides the basic Gabor, Wilson and MDCT transform along with routines for constructing windows (filter prototypes) and routines for manipulating coefficients.

6.2 Introduction

Time-frequency stands as one of the major recent developments in the field of mathematical signal processing. Besides the very natural (and old) idea of providing a localized version of the Fourier transformation, that led long ago to the development of short time Fourier transform (STFT), the theory of Gabor transforms and generalizations has emerged as a new scientific domain, that has found applications in many different areas.

Fourier analysis (see for example the first chapter of [41] for a review) provides expansions for signals as linear combinations of sines and cosines. In the continuous time setting, one writes

$$x(t) = \int_{-\infty}^{\infty} \hat{x}(\nu) e^{2i\pi\nu t}\, d\nu\ ,$$

where $i = \sqrt{-1}$, t is the time variable, ν is the frequency variable, and the Fourier transform $\hat{f}$ of the signal f reads

$$\hat{x}(\nu) = \int_{-\infty}^{\infty} x(t) e^{-2i\pi\nu t}\, dt\ .$$

Similar expansions may be written in the infinite and finite discrete time settings (see below for more details), as well as higher dimensional situations.

The short time Fourier transform of a signal is essentially a family of Fourier transforms of localized versions of the signal, obtained by multiplying it with shifted copies of a window function. Even though the STFT is a commonly used tool in signal analysis (see e.g. [19, 38] for a number of these), it suffers from a number of shortcomings in a number of applications. Among these, one may mention the fact that the STFT generally represents a vastly overcomplete representations for signals, which is often not suitable (for example for signal coding, or simply in terms of computational load or memory requirements). Another consequence of overcompleteness is the non-uniqueness of the signal expansion as linear combination of localized sine waves; this may be seen as a richness for the STFT, but also introduces extra difficulties (which one should one use ?).

Gabor analysis encompasses short time Fourier analysis, by introducing a suitable mathematical framework, within which the role of the window, as well as the sampling of the time-frequency domain are controlled. Gabor analysis provides expansions for signals as (discrete) linear combinations of **Gabor atoms** g_{mn}, defined as shifted and modulated copies of a reference window γ, termed the *synthesis window*, as

$$\gamma_{mn}(t) = e^{2i\pi mbt}\gamma(t - na) \ , \tag{6.1}$$

with t a (discrete or continuous) time variable, and $a, b > 0$ two time and frequency sampling constants. Gabor analysis then provides expansions of the type

$$x = \sum_{m,n} c(m,n)\gamma_{mn} \ , \tag{6.2}$$

and algorithms for the computation of the (generally non-unique) set of coefficients $c(m, n)$ of the expansion, provided that the window γ and the constants a, b have been suitably chosen.

In a dual point of view, Gabor analysis may also be formulated in terms of the Gabor transform, which associates with a signal x the family of its inner products

$$c(m,n) = \langle x, g_{mn}\rangle \tag{6.3}$$

with Gabor atoms g_{mn}, constructed from a single *analysis window* g. In this language, Gabor analysis allows one to invert the Gabor transform, i.e. express the signal x in terms of the coefficients $c(m, n)$, again provided that the window g and the constants a, b have been suitably chosen. The (again generally non-unique) inversion formula then assumes the form (6.2), for some synthesis window γ.

In both situations, the Gabor coefficients $c(m, n)$ may be used for various purposes: they can be analyzed, to provide information on the signal. They can also be modified, which induces a transformation on the signal. Besides analysis and synthesis, signal transformations via coefficients modifications represents the third main aspect of Gabor analysis. The simplest way of modifying Gabor coefficients is to multiply them pointwise with a given mask, or time-frequency transfer function. The transfer function takes the form of a doubly labeled sequence $\mathbf{m}(m, n)$, and generates the transformation

$$\mathbb{M}_{\mathbf{m}}x = \sum_{m,n} \mathbf{m}(m,n)c(m,n)\gamma_{m,n} \ . \tag{6.4}$$

It may be shown that the transformation $\mathbb{M}_{\mathbf{m}}$, termed Gabor multiplier, is a linear operator. By analogy with the classical linear filters commonly used signal processing, which are defined as multiplications with a transfer function in the Fourier domain, Gabor multipliers stand as non-stationary, or time-varying linear filters.

It was proved (see [41] for a review) that classical Gabor analysis may also suffer from some drawbacks, which motivated several authors to propose modified versions of Gabor analysis, or generalizations. One of these drawbacks is the impossibility of finding smooth windows g and sampling constants a, b such that the corresponding family of Gabor atoms would generate an orthonormal basis of the space of signals (the reader not familiar with the elements of linear algebra relevant for signal processing may want to consult the first chapter of [104] for an outline). Mainly two constructions were proposed for time-frequency like orthonormal bases: Wilson bases, and MDCT (modified discrete cosine transform) bases. Both are based upon a subtle modification of the construction rule of Gabor functions, which overcome the obstruction that prevents Gabor atoms from forming orthonormal bases. MDCT bases have become extremely popular in practical applications, as they are commonly used in audio coders such as the standards mp3, ogg vorbis and mpeg4 aac.

Such linear time-frequency decomposition methods actually offer a wide variety of expansion methods, which are adapted to various situations. In the signal processing domain, redundant representations (for example Gabor expansions with small values of the product ab) are generally preferred for signal analysis purposes, as they often allow the user to “read” relevant information from the transform, and proceed to further tasks such as detection, parameter estimation,... Redundant representations turn out to be also extremely useful for building time-varying signal filters, as mentioned above. On the contrary, applications such as signal coding and compression prefer to avoid redundancy, and Gabor systems with low redundancy and Wilson or MDCT bases are then preferred. Such is also the case for signal denoising, even though very little is known regarding denoising in redundant systems.

The LTFAT toolbox features a number of linear time-frequency transformation tools, pure frequency transforms, and some signal processing tools including examples and test signals.

All signals, windows and transforms are considered periodic. This gives the fastest algorithms and the simplest mathematics, at the expense of the sometimes unnatural periodic boundary condition.

Section 6.3 introduces the basic tools from discrete Fourier analysis and discrete time/frequency analysis. These methods are mostly intended for teaching and general exploration of the field.

Section 6.4 introduces the three time/frequency transforms: The Gabor transform, the Wilson transform and the MDCT. These are the main computational methods in the toolbox.

Section 6.5 introduces a collection of tools that can be used for the construction of windows (filter prototypes) for the transforms.

Section 6.6 introduces the plotting routines and the time-varying filter method.

Section 6.7 introduces a collection of signal processing tools and examples that complements the time-frequency methods. These tools and examples demonstrates how time-frequency analysis is useful for a range of signal processing tasks.

Section 6.8 discuss the algorithms and implementation of the toolbox.

6.3 Basic Fourier and time-frequency analysis

The toolbox contains some basic Fourier analysis tools intended mostly for teaching. This includes `dft`, a unitary discrete Fourier transform (DFT), its inverse `idft`, periodic convolution `pconv` and involution `involute`.

The DFT $c \in \mathbb{C}^L$ of a signal $f \in \mathbb{C}^L$ is given by

$$c\left(k\right)=\frac{1}{\sqrt{L}}\sum_{l=0}^{L-1}f(l)e^{-2\pi ikl/L},\quad k=0,\ldots,L-1. \tag{6.5}$$

The inverse transform reads

$$f(l)=\frac{1}{\sqrt{L}}\sum_{k=0}^{L-1}c(k)e^{2\pi ikl/L},\quad l=0,\ldots,L-1. \tag{6.6}$$

The invertibility of the DFT originates from the fact that the family of vectors $\epsilon^k \in \mathbb{C}^L$ defined by their components

$$\epsilon_l^k=\frac{1}{\sqrt{L}}e^{2\pi ikl/L}\ ,\quad l=0\ldots L-1 \tag{6.7}$$

form an orthonormal basis of $\mathbb{C}^L$.

The standard DFT implementation `fft` that is included in Matlab and Octave is normalized differently. The periodic convolution $h \in \mathbb{C}^L$ of two signals $f, g \in \mathbb{C}^L$ is given by

$$h\left(l\right)=\sum_{k=0}^{L-1}f(k)\overline{g(l-k)},\quad l=0,\ldots,L-1. \tag{6.8}$$

The involution $h \in \mathbb{C}^L$ of a signal $f \in \mathbb{C}^L$ is the signal reversed except for its first element:

$$h\left(l\right)=f\left(-l\right)\quad l=0,\ldots,L-1. \tag{6.9}$$

For more information on the relationship between Fourier transformation, convolution and involution, see [41, Ch. 1].

The Zak transform is an important tool in relation to non-redundant time/frequency analysis and critically sampled Filter banks. It is also known as the polyphase representation. The finite discrete Zak transform $Z_a f$ of a signal $f \in \mathbb{C}^L$ is given by

$$\left(Z_a h\right)\left(r,s\right)\ =\ \sqrt{\frac{a}{L}}\sum_{l=0}^{L/K-1}h\left(r-la\right)e^{2\pi isla/L}.\quad r,s\in\mathbb{Z}. \tag{6.10}$$

For $r = 0, \ldots, a-1$ and $s = 0, \ldots, \frac{L}{a}-1$ the Zak transform can be computed efficiently by the routine `zak`. Values outside this domain can be computed by a simple multiplication by an exponential function, see [49, 17].

In addition to the Discrete Fourier transform, the toolbox also contains the classical discrete cosine and sine transforms type I-IV. These transforms are real valued (real valued input gives real valued output) as opposed to the DFT, and they are therefore sometimes better suited for practical applications on real valued signals.

The toolbox contains a list of functions with special behavior in time and frequency: A periodic chirp `pchirp`, that grows linearly in frequency, two families of Hermite functions which are invariant with respect to the DFT and the Shah distribution `shah` (also known as a pulse train) which is the least concentrated function in time and frequency.

6.4 Time/Frequency transforms

The prototype of time-frequency transform is the *Short Time Fourier Transform* (STFT). The STFT of a signal $f \in \mathbb{C}^L$ is obtained by Fourier transforming copies of f that are localized by pointwise multiplication with a sliding window $l \to g(l-n)$, i.e. by computing

$$\mathrm{stft}(m,n) = \sum_{l=0}^{L-1} f(l)\overline{g}(l-n)e^{-2i\pi ml/L} \ , \quad m,n = 0,\dots L \ , \tag{6.11}$$

the bar denoting complex conjugation, and being purely conventional here (for coherence with the next sections). The simplest choice for the window g, namely a rectangular window turns out to be quite inappropriate in practice, and smoother window functions are generally preferred.

The STFT is invertible: a direct calculation yields

$$f(l) = \frac{1}{\|g\|^2} \sum_{m,n=0}^{L-1} \mathrm{stft}(m,n)g(l-n)e^{2i\pi ml/L} \ , \tag{6.12}$$

where $\|g\|$ is the Euclidean norm of the vector g. However, it is worth mentioning that the STFT of a signal represents highly redundant information, since a signal $f \in \mathbb{C}^L$ is represented by L^2 coefficients $\mathrm{stft}(m,n)$. Such a redundancy may be useful in a number of applications. However, for large L, the large size of full STFT is often not suitable. One has to subsample the STFT, which lead to Gabor transforms.

6.4.1 The discrete Gabor transform

The STFT inversion formula (6.12) may be interpreted as an expansion of the signal f with respect to a (redundant) system of elementary waveforms

$$l \longrightarrow g(l-n)e^{2i\pi ml/L} \ , \qquad m,n = 0,\dots L-1.$$

Subsampling this redundant system leads to the so-called Gabor systems, also called Weyl-Heisenberg systems, or Fourier modulated filter banks [18]:

$$l \longrightarrow g(l-an)e^{2i\pi ml/M} \ , \qquad m = 0,\dots M-1, n = 0,\dots N-1. \tag{6.13}$$

The following parameters describes the size of a discrete Gabor transform:

a	:	Time separation.
M	:	Number of frequency bands.
L	:	Length of signal.

It must hold that $L = Mb = Na$. With these parameters, the coefficients $c \in \mathbb{C}^{M\times N}$ computed by the DGT of the signal f are given by:

$$c\,(m,n) = \sum_{l=0}^{L-1} f(k)e^{-2\pi iml/M}\overline{g\,(l-an)}, \tag{6.14}$$

These coefficients are samples of the discrete short-time Fourier transform of the signal. They are complex, even if both window and input signal are real.

Regarding inversion from a discrete Gabor transform, it is convenient to introduce the following notion.

Definition 6.4.1. A family of vectors e_j, $j = 0, ..., J-1$ of length L is called a *frame* if constants $0 < A \leq B$ exist such that

$$A \|f\|^2 \leq \sum_{j=0}^{J-1} |\langle f, e_j \rangle|^2 \leq B \|f\|^2 , \quad \forall f \in \mathbb{C}^L. \tag{6.15}$$

The constants A and B are called lower and upper frame bounds. If $A = B$, the frame is called *tight*. If $J > L$, the frame is redundant (oversampled).

The redundancy of the frame is the fraction $\frac{J}{L}$. Finite- and infinite dimensional frames are described in [23].

In the finite dimensional setting we are concerned with, a frame is essentially a family of vectors which is complete in the considered signal space. The introduction of frame bounds is useful, since it allows one to control the convergence rate of the inversion algorithm. A frame may be a basis, but interesting frames are generally redundant.

A nice and useful feature of frames is the existence of (generally infinitely many) *dual frame(s)*, from which signal expansions may be obtained. More precisely, given a frame $\{e_j, \, j = 0, ..., J-1\}$, there exist a family $\{\tilde{e}_j, \, j = 0, ..., J-1\}$ such that for all $f \in \mathbb{C}^L$,

$$f = \sum_{j=0}^{J-1} \langle f, e_j \rangle \tilde{e}_j \tag{6.16}$$

$$= \sum_{j=0}^{J-1} \langle f, \tilde{e}_j \rangle e_j \tag{6.17}$$

Among the dual frames, the so-called *canonical dual frame* has a specific status, in the sense that it is the closest to the original frame. The canonical dual frame is constructed as follows.

Consider the linear transformation $S : \mathbb{C}^L \to \mathbb{C}^L$, defined by

$$(Sf)(l) = \sum_{j=0}^{J-1} \langle f, e_j \rangle e_j(l) \ , \qquad l = 0, \dots L-1 \ . \tag{6.18}$$

It may be shown that S is invertible. The canonical dual frame is then defined by

$$\tilde{e}_j^o = S^{-1} e_j \ , \qquad j = 0, \dots J-1 \ . \tag{6.19}$$

In the particular case of Gabor frames, the canonical dual frame of a Gabor frame is still a Gabor frame, and is therefore generated using the *canonical dual window*

$$g^d = S^{-1} g \ . \tag{6.20}$$

Another window of special importance is the *canonical tight window*

$$g^t = S^{-1/2} g. \tag{6.21}$$

This window generates a tight Gabor frame and gives perfect reconstruction if it is used for both analysis and synthesis, much like the situation of an orthonormal basis. Given an analysis window g, and lattice constants a and b that generate a Gabor frame, a vector h is an admissible dual window if for all $f \in \mathbb{C}^L$,

$$f = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \langle f, g_{mn} \rangle h_{mn} \ . \tag{6.22}$$

To obtain perfect reconstruction with well-behaved windows, it is necessary that the Gabor system be redundant, so the Gabor system is a frame, and not a basis. A Gabor frame is redundant if $ab < L$. For detailed information about Gabor systems, see the books [41, 35, 36].

6.4.2 The discrete Wilson transform

Wilson bases were proposed as substitutes for Gabor frames, because of the impossibility of constructing Gabor systems that would be simultaneously generated using well behaved windows, and bases of the considered signal spaces.

A Wilson basis is formed by taking linear combinations of appropriate basis functions from a Gabor frame with redundancy 2, [15]. This remarkable constructions turns a tight Gabor frame into an real, orthonormal basis, or turns a non-tight Gabor frame into a Riesz basis (corresponding to a biorthogonal filter bank). In [67] this system is described as a “linear phase cosine modulated maximally decimated filter bank with perfect reconstruction”.

The coefficients $w \in \mathbb{C}^{2M \times \frac{L}{2M}}$ computed by the Discrete Wilson Transform, DWILT, of the signal $f \in \mathbb{C}^L$ are given by

If $m = 0$:

$$w\left(0, n\right) \quad = \quad \sum_{l=0}^{L-1} f(l) g\left(l - 2na\right) \tag{6.23}$$

If m is odd and less than M:

$$w\left(m, n\right) \quad = \quad \sqrt{2} \sum_{l=0}^{L-1} f(l) \sin(\pi \frac{m}{M} l) g(l - 2na) \tag{6.24}$$

$$w\left(m + M, n\right) \quad = \quad \sqrt{2} \sum_{l=0}^{L-1} f(l) \cos(\pi \frac{m}{M} l) g\left(l - \left(2n + 1\right) a\right) \tag{6.25}$$

If m is even and less than M:

$$w\left(m, n\right) \quad = \quad \sqrt{2} \sum_{l=0}^{L-1} f(l) \cos(\pi \frac{m}{M} l) g(l - 2na) \tag{6.26}$$

$$w\left(m + M, n\right) \quad = \quad \sqrt{2} \sum_{l=0}^{L-1} f(l) \sin(\pi \frac{m}{M} l) g\left(l - \left(2n + 1\right) a\right) \tag{6.27}$$

If $m = M$ and M is even:

$$w(M,n) = \sum_{l=0}^{L-1} f(l)(-1)^l g(l-2na) \tag{6.28}$$

else if $m = M$ and M is odd:

$$w(M,n) = \sum_{k=0}^{L-1} f(l)(-1)^l g\left(l-(2n+1)a\right) \tag{6.29}$$

Some notes on this: $w_{0,n}$ and $w_{M,n}$ only have half the bandwidth of the other filters. This implies that an M-channel Wilson filter bank will split a signal into $M+1$ frequency bands, with the highest and lowest frequency band having half the bandwidth.

If a Wilson basis is considered as an $2M$-channel filter bank, then roughly half of the filters ($M-1$ if M is even, otherwise $M+1$) are time-shifted by M samples.

6.4.3 The modified discrete cosine transform (MDCT)

The MDCT (modified discrete cosine transform) is another substitute for the non-existent Gabor bases, that has become extremely popular recently for its numerous applications, in audio coding for instance [81, 82, 72]. The main idea is again to provide a local version for trigonometric bases, using smooth windows rather than rectangular ones.

The coefficients $c \in \mathbb{C}^{M\times N}$ computed by the MDCT of $f \in \mathbb{C}^L$ are given by:

For $m+n$ even:

$$w(m,n) = \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)l+\frac{\pi}{4}\right)g(l-na). \tag{6.30}$$

For $m+n$ odd:

$$w(m,n) = \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)l+\frac{\pi}{4}\right)g(l-na). \tag{6.31}$$

Notice that this definition of the MDCT is not the most common one: the common definition of the MDCT require the window to have the HPE symmetry (see Sec. 6.5), whereas this definition uses WPE windows just as the Gabor and Wilson transforms.

6.5 Window design

The following two types of symmetries are important in discrete time/frequency analysis:

Definition 6.5.1. Let $g \in \mathbb{C}^L$. We say that g is *whole point even* (WPE) if

$$g(l) = \overline{g(-l)} = \overline{g(L-l)} \tag{6.32}$$

for $l = 0,\ldots,L-1$. This implies that $g(0)$ must always be real, and so must $g\left(\frac{L}{2}+1\right)$ if L is even.

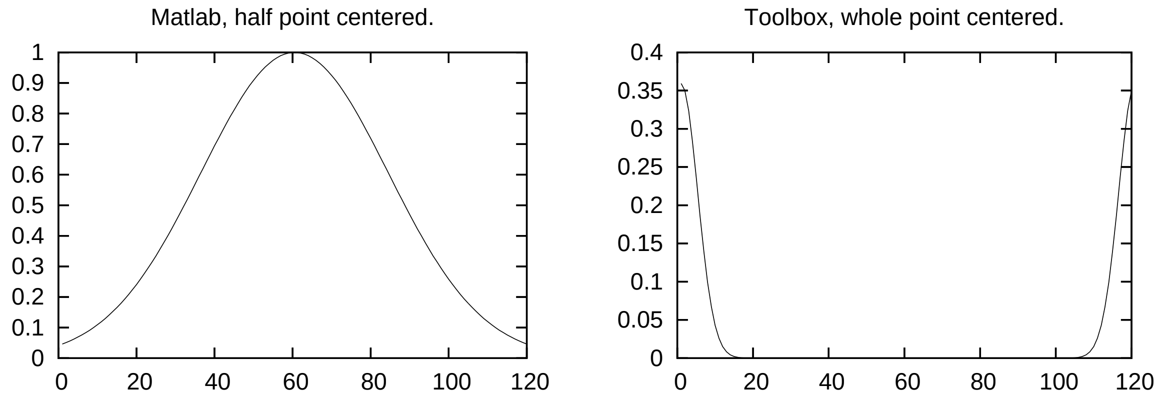


Figure 6.1: The figure to the left shows the output of the Matlab command `gausswin(120)` while the figure to the right show the output of the LTFAT command `pgauss(120)`.

Definition 6.5.2. Let $g \in \mathbb{C}^L$. We say that g is *half point even* (HPE) if

$$g(l) = \overline{g(L-1-l)} \tag{6.33}$$

for $l = 0, \ldots, L-1$. This implies that $g\left(\frac{L-1}{2}\right)$ must be real if L is odd.

A signal g is WPE if and only if the DFT of g is real valued. The HPE symmetry is typically used in signal processing, as an example all windows generated by the signal processing toolbox in Matlab are HPE.

The toolbox can use both FIR (finite impulse response) and IIR (infinite impulse response) windows. FIR windows are shorter than the signal to be analyzed, while IIR windows will have the same length.

The intent of the toolbox is to stay close to the underlying mathematics, and we have therefore adopted a different layout of windows than what it usually done in Matlab/Octave. This difference is visible on Figure 6.1, which shows a Gaussian window computed by the Matlab/Octave `gausswin` and the LTFAT command `pgauss`. The command `gausswin` produces a FIR window made from a truncated Gaussian function that is centered in between the two middle points in the vector (HPE). The command `pgauss` produces an IIR window which is the Gaussian window centered around the first element of the vector (WPE).

Centering the window around the first element make it look unnatural when plotting, however it has the benefit that the Gaussian window is invariant with respect to the DFT, and that reconstructions are produced with zero delay.

All the other window functions in the toolbox work similarly to `pgauss`. They return WPE windows centered around the first element of the vector. A routine `middlepad` is included to cut or extend these kind of windows.

6.5.1 FIR windows

Finite impulse response (FIR) can be generated by the routines `firwin` and `firkaiser`. The routine `firwin` generates the classical Hanning, Hamming and Blackman windows defined in [78]. In addition to these classical windows, also the square root of Hanning and Hamming windows may be generated. The advantage of taking the square root of the Hanning and Hamming window is, that these new windows generates tight Gabor frames or orthonormal Wilson/MDCT bases.

6.5.2 IIR windows

The toolbox contains routines to generate Gaussian (`pgauss`), Sech (`psech`) and Gauss-Hermite (`pherm`) windows. These windows are all sampled and periodized version of their continuous counterparts. They are invariant with respect to a discrete Fourier transform, just as their continuous counterparts are invariant with respect to the Fourier transform.

The finite, discrete Gaussian $\varphi_w \in \mathbb{C}^L$ computed by `pgauss` is given by:

$$\varphi_w(l) = \left(\frac{wL}{2}\right)^{-1/4} \sum_{k\in\mathbb{Z}} e^{-\pi\left((l+c_t)/\sqrt{L}-k\sqrt{L}\right)^2/w}, \tag{6.34}$$

where the parameter $w > 0$ controls the ratio of the time-support and the frequency support of the Gaussian. For a Gabor system (φ_w, a, M) for $\mathbb{C}^L$, the value of w that gives the lowest frame bounds is the choice $w = \frac{a}{b}$, where $b = \frac{L}{M}$. The routines `psech` and `pherm` work entirely similar to `pgauss` taking the same parameter w as input.

The routine `pbspline` can generate several different classes of discrete, fractional splines. These splines are described in detail in [94]. It may also simply be used to generate sampling of the classical B-splines.

6.5.3 Symmetries

All the window routines in the toolbox may be used to generate both WPE and HPE windows. They all take a parameter `centering` as their last input parameter. Setting `centering=0` will result in the output having the WPE symmetry while setting `centering=0.5` will result in the output having the HPE symmetry.

The Wilson and MDCT transform defined in Sec. both require WPE windows if reconstruction is desired. It is easy to define variants of Wilson and MDCT transforms that instead work with HPE windows. These transforms are available through the object-oriented interface.

6.5.4 Dual / tight windows

The canonical dual window (6.20) of a Gabor frame may me easily calculated by `candual`, and the canonical tight window (6.21) by `cantight`. For Wilson and MDCT bases the Riesz dual window is computed by `wildual` and an orthonormal window may be generated by `wilorth`. Wilson and MDCT bases use the same windows.

To judge the quality of a given Gabor system, two methods are supplied. One possibility is to consider the frame bounds (A and B from (6.15)) of a Gabor frame, these can be calculated by `gfbounds`. The ratio $\frac{B}{A}$ of the frame bounds play the exact same role as the condition number of a matrix.

Another method is to consider the reconstruction quality of a pair of windows g, γ. The function `gfdualnorm` will return the maximal reconstruction error of any signal when using the pair g, γ as windows for analysis and synthesis, [4, 57].

6.5.5 Non-canonical dual window

The toolbox contains two routines for fast construction of noncanonical dual windows. The routine `projdual` will project any window onto the space of admissible dual windows. This is very fast and works well if your input window is already close to being a dual window. Another method is to compute the `mixdual` of two windows. It works by mixing the properties of two Gabor systems: The resulting window will be a dual of the first Gabor system, but resemble a mixture of the canonical dual windows of the two Gabor system. The example `examp_mixdual` describes how to use this function to get dual windows that are more or less concentrated than the canonical dual.

To aid in the construction of new windows, a reconstruction measure, `gfdualnorm`, is included. Given an analysis and a synthesis window, the function returns an upper bound for the relative error of analyzing and synthesizing a function f using the two windows. The estimate depends only on the windows, and not on the signal f.

6.6 Other tools

6.6.1 Time variant filtering

Time-invariant filter are systems, where the frequency spectrum is multiplied by a fixed function, called the *transfer function*, [78]. Using the Fourier transform to calculate the spectrum, such an operator can also be called a *Fourier multiplier.* This technique have been used for many years and finds a wide range of applications, e.g. in computer music [28].

A generalization of this technique is the so called *time-variant filtering*, which recently has grown more and more important [110]. If the STFT is used in its sampled version, the Gabor transform, one possibility to construct a time variant filter is the usage of *Gabor multipliers* [34], (6.4). These operators have been already used for quite some time implicitly in engineering applications and recently have been used in signal-processing applications as time-variant filters called *Gabor filters* [74]. Recent applications can be found for example in the field of system identification [69] or psychoacoustical modeling [6]. The toolbox contain a routine `gabmul` for easy application of Gabor multipliers.

6.6.2 Plotting routines

The toolbox includes a few plotting routines:

- A spectrogram plot. This is an easy-to-use function, that displays a spectrogram of the input function. It is based on a suitable Gabor frame. The spectrogram displays the squared amplitude of the Gabor transform, i.e. $|c(m,n)|^2$.
- A phase plot. In the spectrogram the information about the complex angle of the coefficients $c(m,n)$, i.e. the *phase*, gets lost. The phase may sometimes reveal features not visible on the spectrogram, [19].

6.7 Signal processing tasks

The toolbox includes special support for some simple signal processing tasks. This has been done to make the toolbox more self contained, so that is it possible to teach and demonstrate aspects of time/frequency signal processing using only the tools available in the toolbox.

6.7.1 Compression

Signal compression aims at representing a signal with maximal fidelity, using minimal storage. Transform coding is one of the most popular signal compression techniques. Transform coding starts by expanding the signal on a given, suitably chosen basis. Then only the most significant coefficients of the expansion are retained, which provide an approximation of the signal. Finally, the retained coefficients are quantized, i.e. represented using a finite (small) number of bits.

MDCT bases form the main building block for transform coding of audio signals. After an mdct expansion, the significant coefficient selection is generally done by keeping the mdct coefficients corresponding to lower frequencies, i.e. coefficients $w\left(m,n\right)$ with values of m lower than some reference value M_1. Those retained coefficients are then quantized, after suitable weighting. Such schemes are called linear transform coding schemes.

An alternative, called non-linear transform coding, follows a different rule for the selection of significant coefficients. Instead of retaining low frequency coefficients, the coefficients with largest magnitude (after frequency dependent weighting if needed) are retained. The approximation error in such approaches may be proved to be always lower than the approximation error of the corresponding linear transform coding scheme (for the same number of retained coefficients). However, in non-linear coding schemes, reconstruction from retained coefficients is only possible if their “addresses (i.e. corresponding values of (m,n)), termed significance maps, are stored as well, which represents a significant amount of side information. Efficient strategies for representing significance maps are then necessary.

6.7.2 Denoising

Denoising is one of the classical tasks in signal processing. The emergence of time-frequency tools has made it possible to develop efficient methods for performing denoising using thresholding strategies. The rationale is the following. If a signal of interest is represented by a sparse expansion with respect to a given basis, noise is generally not. Therefore, the signal's energy is concentrated in a small number of coefficients, while the noise's energy is spread on all coefficients. Thresholding the coefficients of the noisy signal allows one to get rid of most of the noise contribution. TO BE FINISHED

6.7.3 Feature extraction using formants

Formants [29] in phonetics are considered as the resonant frequencies of the human vocal tract and this information is for example used for speaker recognition [89]. In the file `examp_formants` we give a simple example by finding peaks in the spectrogram and ordering them by frequency. More elaborate methods include using model-based analysis (e.g. LPC) [73] or homomorphic deconvolution methods [77].

6.7.4 OFDM transmission

Orthogonal Frequency Division Multiplexing is a widely use technique for transmitting digital signals. The idea is to generate a series of pulses that are mutually orthogonal. The pulses carry the information, and the orthogonality ensures that the method is robust again noise, delay, dispersion etc.

Gabor frame theory is very useful in this context because for each Gabor frame (6.13) with lattice constants a, b there exist a Gabor Riesz sequence on the dual lattice $M = \frac{L}{b}$, $N = \frac{L}{a}$, [88]. This Gabor Riesz sequence can be used for OFDM, with the benefit that all the methods used for construction of windows for Gabor frames can be reused for OFDM. The toolbox includes methods for QAM modulation and an example `examp_ofdm` that demonstrates the various aspects of OFDM transmission using Gabor frames. For more information on using Gabor and Wilson systems for OFDM, see [97, 14].

6.8 Implementation

The toolbox uses two different algorithms for computing a DGT, depending on the length of the window as compared to the length of the signal.

The short window algorithms correspond to a FIR window filter bank approach. The algorithms are based on an simple application of the FFT in each timestep. The gives a computational complexity of $\mathcal{O}\left(NM \log M + Nd\right)$, where M is the number of channels, N is the number of time shifts, and d is the length of the window.

The long window variants are used when the window has the same length as the signal. The algorithm of the full-window DGT is described in [93] and is based on previous work on matrix factorizations of Gabor operators done in [83, 96]. They long window algorithm require all data to be available before computing the transform. This makes it unsuitable for streaming data through a DSP, but it poses no problem in Matlab. The full long window algorithms have complexity $\mathcal{O}\left(NM \log M + Lq\right)$, where L is the length of the signal, MN is the total number of coefficients and q appears in the redundancy $\frac{q}{p}$ written as an irreducible fraction (i.e. for a Gabor frame with redundancy 1.25, $q = 5$. For a Wilson basis $q = 2$ always).

The DGT algorithms (for either short or full length windows) are also used to calculate the canonical dual and tight windows, frame bounds and the dual norm measure. Wilson and MDCT bases are also handled by the DGT algorithm by the use of pre/post processing stages. This is very convenient, as it make it possible to speed up the whole toolbox by optimizing a single algorithm. For this reason, a C-implementation of the DGT algorithm is included in the toolbox, as well as interfaces for Octave and Matlab (Mex) to use this library. The library links to the efficient FFT implementation (FFTW, see [39]) included in both Octave and Matlab.

The discrete sine and cosine transforms are computed by the classic algorithms published in [109].

Chapter 7

Symmetric Gabor and Wilson systems

In this section we discuss DGTs and Wilson transforms with different symmetry properties. Everything will be done strictly in the finite, discrete setting.

7.1 Symmetries of discrete Fourier analysis

A well known property of the Fourier transform is that the Fourier transform of a real valued function is an even function and vice versa. The same property holds for the finite, discrete transform, but the symmetry property is slightly more complicated. In the continuous case, an even function is symmetric around $x = 0$. The point $x = 0$ on which the mirror axis is placed has measure 0 and therefore the behaviour of a function on the mirror axis is not important in $L^2(\mathbb{R})$ sense.

In the finite, discrete case $\mathbb{C}^L$, there are no sets of measure zero and even functions have special behaviour on the point $l = 0$ where the mirror axis is placed.

This has led to the development of two different approaches: Placing the mirror axis on $l = 0$ or placing the mirror axis on $l = -\frac{1}{2}$ where all indices are considered modulo L. This leads to the following four types of symmetries.

Definition 7.1.1. Let $g \in \mathbb{C}^L$. We say that g is *Whole-point even (WPE)* if it holds that $g(l) = \overline{g(-l)} = \overline{g(L-l)}$ for $l = 0, \ldots, L-1$. This implies that $g(0)$ must always be real and so must $g\left(\frac{L}{2}+1\right)$ if L is even.

Definition 7.1.2. Let $g \in \mathbb{C}^L$. We say that g is *Half-point even (HPE)* if it holds that $g(l) = \overline{g(L-1-l)}$ for $l = 0, \ldots, L-1$. This implies that $g\left(\frac{L-1}{2}\right)$ must be real if L is odd.

It can be shown that if g is HPE then the DFT of g is $\hat{g}(k) = h(k)\, e^{\pi i k/L}$, where h is some real-valued signal: $h \in \mathbb{R}^L$.

Definition 7.1.3. Let $g \in \mathbb{C}^L$. We say that g is *Whole-point odd (WPO)* if it holds that $g(l) = -\overline{g(L-l)}$ for $l = 0, \ldots, L-1$. This implies that $g(0)$ must always be real and $g\left(\frac{L}{2}+1\right)$ must be purely imaginary if L is even.

Definition 7.1.4. Let $g \in \mathbb{C}^L$. We say that g is *Half-point odd (HPO)* if it holds that $g(l) = -\overline{g(L-1-l)}$ for $l = 0, \ldots, L-1$. This implies that $g\left(\frac{L-1}{2}\right)$ must be purely imaginary if L is odd.

In addition we say that a g is *WPE in frequency* if the DFT of g is WPE and similarly for HPE, WPO and HPO. If a signal is WPE in frequency, it implies that it is real-valued. This is a classical property of the DFT.

7.2 The symmetric DGT transforms

If a DGT with a real-valued window is applied to a real-valued signal then the output coefficients will be complex. However, they posses the WPE symmetry in the frequency direction. This means that the upper half plane of the coefficients is just a complex conjugate, mirror image of the lower half plane. This an example of a symmetry of the DGT and in the following we will create a system of DGTs with different symmetries.

We define the General DGT $c = C_g^G f$ by

$$\begin{aligned} c\left(m,n\right) &= \sum_{l=0}^{L-1} f(l) e^{-2\pi i (l+m_t)\left(mb+w_f\right)/L} \overline{g(l-na-w_t)} && (7.1) \\ &= e^{-2\pi i m_t \left(mb+w_f\right)/L} \sum_{l=0}^{L-1} f(l) e^{-2\pi i l \left(mb+w_f\right)/L} \overline{g(l-na-w_t)}, && (7.2) \end{aligned}$$

where $m_t = \left\{0, \frac{1}{2}\right\}$, $w_t = \left\{0, \left\lfloor \frac{a}{2} \right\rfloor\right\}$ and $w_f = \left\{0, \frac{b}{2}\right\}$. Using $w_f = \frac{b}{2}$ is only allowed when b is even.

So a GDGT can be computed from a DGT in the following steps:

1. Compute $\tilde{f}\left(l\right) = f(l) e^{-2\pi i l w_f / L}$. This is a simple modulation of the input signal.
2. Compute a DGT of $\tilde{f}$ using a shifted window: $\tilde{c} = \mathcal{C}_{T_{w_t} g} \tilde{f}$.
3. Correct the phase of the obtained coefficients: $c\left(m,n\right) = e^{-2\pi i m_t \left(mb+w_f\right)/L} \tilde{c}\left(m,n\right)$.

This algorithm show that the GDGT in itself is not particular interesting, as it is so simple to compute from a regular DGT.

The GDGT will serve as a helping construction to define 8 different DGTs indexed by roman numerals: DGTI-DGTVIII depending on the choice of variables m_t, w_t and w_f. Table A.1 shows an overview of the 8 transforms and the value of m_t, w_t and w_f associated to each transform. We shall denote the coefficient operator of DGTI-DGTVIII by C^I - C^{VIII}.

7.3 The symmetric Wilson transforms

In this section we use the DGTI-DGTIV transform to construct different Wilson transforms.

The motivation is the fact that the finite, discrete Wilson transform as defined in [15] require the input window to have the WPE symmetry. However, in the signal processing litterature as well as in Matlab and Octave, the most common symmetry is the HPE. Therefore, we wish to construct a Wilson basis working with HPE windows.

Table 7.1: Symmetries of the CDGT.

Name	m_t	w_f	w_t	Real	WPE	HPE
DGTI	0	0	0	WPE	WPE	-
DGTII	$\frac{1}{2}$	0	0	WPO	-	WPE
DGTIII	0	$\frac{b}{2}$	0	HPE	WPE	-
DGTIV	$\frac{1}{2}$	$\frac{b}{2}$	0	HPO	-	WPE
DGTV	0	0	$\left\lfloor\frac{a}{2}\right\rfloor$	WPE	HPE	-
DGTVI	$\frac{1}{2}$	0	$\left\lfloor\frac{a}{2}\right\rfloor$	WPO	-	HPE
DGTVII	0	$\frac{b}{2}$	$\left\lfloor\frac{a}{2}\right\rfloor$	HPE	HPE	-
DGTVIII	$\frac{1}{2}$	$\frac{b}{2}$	$\left\lfloor\frac{a}{2}\right\rfloor$	HPO	-	HPE

The fifth to seventh column display the symmetry of the output coefficients of the DGT listed in the first column. Fifth column show the symmetry in frequency when both the signal f and the window g are real-valued. The sixth and seventh column show the symmetry in time when both f and g are WPE and HPE, respectively. A "-" indicate no particular symmetry in the output coefficients.

The other motivation is the fact that a Wilson basis with M channels as defined in Section A.3.1 splits the signal into $M+1$ different frequency bands, where the first and last band only have half the bandwidth of the others. We would like to construct a Wilson transform where all subbands have the same bandwidth. This leads to the construction of the Modified Discrete Cosine Transform, originally defined in [81, 82].

A requirement for all the Wilson transform is that L/M must be even, where $g \in \mathbb{C}^L$ and M denoted the number of channels. The requirement that L/M be even is due to the fact that Wilson bases are constructed differently depending on whether the number of the timeshift is even or odd.

7.3.1 DWILT

This is the original finite, discrete Wilson transform as defined in [15].

Let $c = C^{I}_{\left(g,a,\frac{b}{2}\right)} f$. We define $C^{W}_{(g,M)} f \in \mathbb{C}^{2M\times\frac{N}{2}}$ by

If $m = 0$:

$$\begin{aligned} w\left(0,n\right) &= \sum_{l=0}^{L-1} f(l) g\left(l-2na\right) && (7.3) \\ &= c\left(0,2n\right) && (7.4) \end{aligned}$$

If m is odd and less than M:

$$\begin{aligned} w\left(m,n\right) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin(\pi\frac{m}{M}l)g(l-2na) && (7.5)\\ &= \frac{i}{\sqrt{2}}\left(c\left(m,2n\right)-c\left(2M-m,2n\right)\right) && (7.6)\\ w\left(m+M,n\right) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos(\pi\frac{m}{M}l)g\left(l-\left(2n+1\right)a\right) && (7.7)\\ &= \frac{1}{\sqrt{2}}\left(c\left(m,2n+1\right)+c\left(2M-m,2n+1\right)\right) && (7.8)\end{aligned}$$

If m is even and less than M:

$$\begin{aligned} w\left(m,n\right) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos(\pi\frac{m}{M}l)g(l-2na) && (7.9)\\ &= \frac{1}{\sqrt{2}}\left(c\left(m,2n\right)+c\left(2M-m,2n\right)\right) && (7.10)\\ w\left(m+M,n\right) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin(\pi\frac{m}{M}l)g\left(l-\left(2n+1\right)a\right) && (7.11)\\ &= \frac{i}{\sqrt{2}}\left(c\left(m,2n+1\right)+c\left(2M-m,2n+1\right)\right) && (7.12)\end{aligned}$$

If $m=M$ and M is even:

$$\begin{aligned} w\left(M,n\right) &= \sum_{l=0}^{L-1} f(l)(-1)^l g(l-2na) && (7.13)\\ &= c\left(M,2n\right) && (7.14)\end{aligned}$$

else if $m=M$ and M is odd:

$$\begin{aligned} w\left(M,n\right) &= \sum_{k=0}^{L-1} f(l)(-1)^l g\left(l-\left(2n+1\right)a\right) && (7.15)\\ &= c\left(M,2n+1\right) && (7.16)\end{aligned}$$

Note that when M is even then the transform is composed of two filterbanks with windows $T_{2na}g$ and $T_{2na+1}g$, respectively. The first filterbanks has both a DC and Nyquest channel, while the second has neither. When M is odd then the first filterbank has a DC channel while the second has a Nyquest channel.

7.3.2 DWILTII

This is a modification of the DWILT using HPE windows instead.

Let $c = C^{II}_{\left(g,a,\frac{b}{2}\right)}f$. We define $w = C^{W-II}_{(g,M)}f \in \mathbb{C}^{2M\times\frac{N}{2}}$ by

If $m = 0$:

$$\begin{aligned} w(m,n) &= \sum_{l=0}^{L-1} f(l) g(l - 2na) && (7.17) \\ &= c(0, 2n) && (7.18) \end{aligned}$$

If m is odd and less than M:

$$\begin{aligned} w(m,n) &= \sqrt{2} \sum_{l=0}^{L-1} f(l) \sin\left(\pi \frac{m}{M}\left(l + \frac{1}{2}\right)\right) g(l - 2na) && (7.19) \\ &= \frac{i}{\sqrt{2}} \left(c(m, 2n) + c(2M - m, 2n)\right) && (7.20) \\ w(m+M, n) &= \sqrt{2} \sum_{l=0}^{L-1} f(l) \cos\left(\pi \frac{m}{M}\left(l + \frac{1}{2}\right)\right) g(l - (2n+1)a) && (7.21) \\ &= \frac{1}{\sqrt{2}} \left(c(m, 2n+1) - c(2M - m, 2n+1)\right) && (7.22) \end{aligned}$$

If m is even and less than M:

$$\begin{aligned} w(m,n) &= \sqrt{2} \sum_{l=0}^{L-1} f(l) \cos\left(\pi \frac{m}{M}\left(l + \frac{1}{2}\right)\right) g(l - 2na) \\ &= \frac{1}{\sqrt{2}} \left(c(m, 2n) - c(2M - m, 2n)\right) \\ w(m+M, n) &= \sqrt{2} \sum_{l=0}^{L-1} f(l) \sin\left(\pi \frac{m}{M}\left(l + \frac{1}{2}\right)\right) g(l - (2n+1)a) \\ &= \frac{i}{\sqrt{2}} \left(c(m, 2n+1) + c(2M - m, 2n+1)\right) \end{aligned}$$

If $m = M$ and M is even:

$$\begin{aligned} w(m+M, n) &= \sum_{l=0}^{L-1} f(l) (-1)^l g(l - (2n+1)a) && (7.23) \\ &= ic(M, 2n+1) && (7.24) \end{aligned}$$

else if $m = M$ and M is odd:

$$\begin{aligned} w(m+M, n) &= \sum_{l=0}^{L-1} f(l) (-1)^l g(l - 2na) && (7.25) \\ &= ic(M, 2n) && (7.26) \end{aligned}$$

7.3.3 DWILTIII

This is the MDCT using Windows.

Let $c = C^{III}_{\left(g,a,\frac{b}{2}\right)} f$. We define $w = C^{W-III}_{(g,M)} f \in \mathbb{C}^{M\times N}$ by

For $m+n$ even:

$$\begin{aligned} w\left(m,n\right) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)l+\frac{\pi}{4}\right)g(l-na) && (7.27)\\ &= \frac{1}{\sqrt{2}}\left(e^{-i\pi/4}c\left(m,n\right)+e^{i\pi/4}c\left(M-1-m,n\right)\right) && (7.28)\end{aligned}$$

For $m+n$ odd:

$$\begin{aligned} w\left(m,n\right) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)l+\frac{\pi}{4}\right)g(l-na) && (7.29)\\ &= \frac{1}{\sqrt{2}}\left(e^{i\pi/4}c\left(m,n\right)+e^{-i\pi/4}c\left(M-1-m,n\right)\right) && (7.30)\end{aligned}$$

7.3.4 DWILTIV

This is the MDCT using HPE windows. It is the definition most commonly referred to as the MDCT.

Let $c = C^{IV}_{\left(g,a,\frac{b}{2}\right)} f$. We define $w = C^{W-IV}_{(g,M)} f \in \mathbb{C}^{M\times N}$ by

For $m+n$ even:

$$\begin{aligned} w\left(m,n\right) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)\left(l+\frac{1}{2}\right)+\frac{\pi}{4}\right)g(l-na) && (7.31)\\ &= \frac{1}{\sqrt{2}}\left(e^{-i\pi/4}c\left(m,n\right)+e^{-i\pi 3/4}c\left(M-1-m,n\right)\right) && (7.32)\end{aligned}$$

For $m+n$ odd:

$$\begin{aligned} w\left(m,n\right) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)\left(l+\frac{1}{2}\right)+\frac{\pi}{4}\right)g(l-na) && (7.33)\\ &= \frac{1}{\sqrt{2}}\left(e^{i\pi/4}c\left(m,n\right)+e^{i\pi 3/4}c\left(M-1-m,n\right)\right) && (7.34)\end{aligned}$$

Appendix A

Symmetric Gabor and Wilson systems

In this section we discuss DGTs and Wilson transforms with different symmetry properties. Everything will be done strictly in the finite, discrete setting.

A.1 Symmetries of discrete Fourier analysis

A well known property of the Fourier transform is that the Fourier transform of a real valued function is an even function and vice versa. The same property holds for the finite, discrete transform, but the symmetry property is slightly more complicated. In the continuous case, an even function is symmetric around $x = 0$. The point $x = 0$ on which the mirror axis is placed has measure 0 and therefore the behaviour of a function on the mirror axis is not important in $L^2(\mathbb{R})$ sense.

In the finite, discrete case $\mathbb{C}^L$, there are no sets of measure zero and even functions have special behaviour on the point $l = 0$ where the mirror axis is placed.

This has led to the development of two different approaches: Placing the mirror axis on $l = 0$ or placing the mirror axis on $l = -\frac{1}{2}$ where all indices are considered modulo L. This leads to the following four types of symmetries.

Definition A.1.1. Let $g \in \mathbb{C}^L$. We say that g is *Whole-point even (WPE)* if it holds that $g(l) = \overline{g(-l)} = \overline{g(L-l)}$ for $l = 0, \ldots, L-1$. This implies that $g(0)$ must always be real and so must $g\left(\frac{L}{2}+1\right)$ if L is even.

Definition A.1.2. Let $g \in \mathbb{C}^L$. We say that g is *Half-point even (HPE)* if it holds that $g(l) = \overline{g(L-1-l)}$ for $l = 0, \ldots, L-1$. This implies that $g\left(\frac{L-1}{2}\right)$ must be real if L is odd.

It can be shown that if g is HPE then the DFT of g is $\hat{g}(k) = h(k)\, e^{\pi i k/L}$, where h is some real-valued signal: $h \in \mathbb{R}^L$.

Definition A.1.3. Let $g \in \mathbb{C}^L$. We say that g is *Whole-point odd (WPO)* if it holds that $g(l) = -\overline{g(L-l)}$ for $l = 0, \ldots, L-1$. This implies that $g(0)$ must always be real and $g\left(\frac{L}{2}+1\right)$ must be purely imaginary if L is even.

Definition A.1.4. Let $g \in \mathbb{C}^L$. We say that g is *Half-point odd (HPO)* if it holds that $g(l) = -\overline{g(L-1-l)}$ for $l = 0, \ldots, L-1$. This implies that $g\left(\frac{L-1}{2}\right)$ must be purely imaginary if L is odd.

In addition we say that a g is *WPE in frequency* if the DFT of g is WPE and similarly for HPE, WPO and HPO. If a signal is WPE in frequency, it implies that it is real-valued. This is a classical property of the DFT.

A.2 The symmetric DGT transforms

If a DGT with a real-valued window is applied to a real-valued signal then the output coefficients will be complex. However, they posses the WPE symmetry in the frequency direction. This means that the upper half plane of the coefficients is just a complex conjugate, mirror image of the lower half plane. This an example of a symmetry of the DGT and in the following we will create a system of DGTs with different symmetries.

We define the General DGT $c = C_g^G f$ by

$$\begin{aligned} c\left(m,n\right) &= \sum_{l=0}^{L-1} f(l) e^{-2\pi i (l+m_t)\left(mb+w_f\right)/L} \overline{g(l-na-w_t)} & \text{(A.1)} \\ &= e^{-2\pi i m_t \left(mb+w_f\right)/L} \sum_{l=0}^{L-1} f(l) e^{-2\pi i l \left(mb+w_f\right)/L} \overline{g(l-na-w_t)}, & \text{(A.2)} \end{aligned}$$

where $m_t = \left\{0, \frac{1}{2}\right\}$, $w_t = \left\{0, \left\lfloor \frac{a}{2} \right\rfloor\right\}$ and $w_f = \left\{0, \frac{b}{2}\right\}$. Using $w_f = \frac{b}{2}$ is only allowed when b is even.

So a GDGT can be computed from a DGT in the following steps:

1. Compute $\tilde{f}\left(l\right) = f(l) e^{-2\pi i l w_f / L}$. This is a simple modulation of the input signal.
2. Compute a DGT of $\tilde{f}$ using a shifted window: $\tilde{c} = \mathcal{C}_{T_{w_t} g} \tilde{f}$.
3. Correct the phase of the obtained coefficients: $c\left(m,n\right) = e^{-2\pi i m_t \left(mb+w_f\right)/L} \tilde{c}\left(m,n\right)$.

This algorithm show that the GDGT in itself is not particular interesting, as it is so simple to compute from a regular DGT.

The GDGT will serve as a helping construction to define 8 different DGTs indexed by roman numerals: DGTI-DGTVIII depending on the choice of variables m_t, w_t and w_f. Table A.1 shows an overview of the 8 transforms and the value of m_t, w_t and w_f associated to each transform. We shall denote the coefficient operator of DGTI-DGTVIII by C^I - C^{VIII}.

A.3 The symmetric Wilson transforms

In this section we use the DGTI-DGTIV transform to construct different Wilson transforms.

The motivation is the fact that the finite, discrete Wilson transform as defined in [15] require the input window to have the WPE symmetry. However, in the signal processing litterature as well as in Matlab and Octave, the most common symmetry is the HPE. Therefore, we wish to construct a Wilson basis working with HPE windows.

Table A.1: Symmetries of the CDGT.

Name	m_t	w_f	w_t	Real	WPE	HPE
DGTI	0	0	0	WPE	WPE	-
DGTII	$\frac{1}{2}$	0	0	WPO	-	WPE
DGTIII	0	$\frac{b}{2}$	0	HPE	WPE	-
DGTIV	$\frac{1}{2}$	$\frac{b}{2}$	0	HPO	-	WPE
DGTV	0	0	$\left\lfloor\frac{a}{2}\right\rfloor$	WPE	HPE	-
DGTVI	$\frac{1}{2}$	0	$\left\lfloor\frac{a}{2}\right\rfloor$	WPO	-	HPE
DGTVII	0	$\frac{b}{2}$	$\left\lfloor\frac{a}{2}\right\rfloor$	HPE	HPE	-
DGTVIII	$\frac{1}{2}$	$\frac{b}{2}$	$\left\lfloor\frac{a}{2}\right\rfloor$	HPO	-	HPE

The fifth to seventh column display the symmetry of the output coefficients of the DGT listed in the first column. Fifth column show the symmetry in frequency when both the signal f and the window g are real-valued. The sixth and seventh column show the symmetry in time when both f and g are WPE and HPE, respectively. A "-" indicate no particular symmetry in the output coefficients.

The other motivation is the fact that a Wilson basis with M channels as defined in Section A.3.1 splits the signal into $M+1$ different frequency bands, where the first and last band only have half the bandwidth of the others. We would like to construct a Wilson transform where all subbands have the same bandwidth. This leads to the construction of the Modified Discrete Cosine Transform, originally defined in [81, 82].

A requirement for all the Wilson transform is that L/M must be even, where $g \in \mathbb{C}^L$ and M denoted the number of channels. The requirement that L/M be even is due to the fact that Wilson bases are constructed differently depending on whether the number of the timeshift is even or odd.

A.3.1 DWILT

This is the original finite, discrete Wilson transform as defined in [15].

Let $c = C^I_{\left(g,a,\frac{b}{2}\right)} f$. We define $C^W_{(g,M)} f \in \mathbb{C}^{2M \times \frac{N}{2}}$ by

If $m = 0$:

$$w(0,n) = \sum_{l=0}^{L-1} f(l) g(l - 2na) \tag{A.3}$$

$$= c(0, 2n) \tag{A.4}$$

If m is odd and less than M:

$$w(m,n) = \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin(\pi\frac{m}{M}l)g(l-2na) \tag{A.5}$$
$$= \frac{i}{\sqrt{2}}\left(c(m,2n) - c(2M-m,2n)\right) \tag{A.6}$$
$$w(m+M,n) = \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos(\pi\frac{m}{M}l)g(l-(2n+1)a) \tag{A.7}$$
$$= \frac{1}{\sqrt{2}}\left(c(m,2n+1) + c(2M-m,2n+1)\right) \tag{A.8}$$

If m is even and less than M:

$$w(m,n) = \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos(\pi\frac{m}{M}l)g(l-2na) \tag{A.9}$$
$$= \frac{1}{\sqrt{2}}\left(c(m,2n) + c(2M-m,2n)\right) \tag{A.10}$$
$$w(m+M,n) = \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin(\pi\frac{m}{M}l)g(l-(2n+1)a) \tag{A.11}$$
$$= \frac{i}{\sqrt{2}}\left(c(m,2n+1) + c(2M-m,2n+1)\right) \tag{A.12}$$

If $m = M$ and M is even:

$$w(M,n) = \sum_{l=0}^{L-1} f(l)(-1)^l g(l-2na) \tag{A.13}$$
$$= c(M,2n) \tag{A.14}$$

else if $m = M$ and M is odd:

$$w(M,n) = \sum_{k=0}^{L-1} f(l)(-1)^l g(l-(2n+1)a) \tag{A.15}$$
$$= c(M,2n+1) \tag{A.16}$$

Note that when M is even then the transform is composed of two filterbanks with windows $T_{2na}g$ and $T_{2na+1}g$, respectively. The first filterbanks has both a DC and Nyquest channel, while the second has neither. When M is odd then the first filterbank has a DC channel while the second has a Nyquest channel.

A.3.2 DWILTII

This is a modification of the DWILT using HPE windows instead.

Let $c = C^{II}_{\left(g,a,\frac{b}{2}\right)}f$. We define $w = C^{W-II}_{(g,M)}f \in \mathbb{C}^{2M\times\frac{N}{2}}$ by

If $m = 0$:

$$\begin{aligned} w(m,n) &= \sum_{l=0}^{L-1} f(l) g(l-2na) && \text{(A.17)}\\ &= c(0,2n) && \text{(A.18)} \end{aligned}$$

If m is odd and less than M:

$$\begin{aligned} w(m,n) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin\left(\pi\frac{m}{M}\left(l+\frac{1}{2}\right)\right) g(l-2na) && \text{(A.19)}\\ &= \frac{i}{\sqrt{2}}\left(c(m,2n)+c(2M-m,2n)\right) && \text{(A.20)}\\ w(m+M,n) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos\left(\pi\frac{m}{M}\left(l+\frac{1}{2}\right)\right) g(l-(2n+1)a) && \text{(A.21)}\\ &= \frac{1}{\sqrt{2}}\left(c(m,2n+1)-c(2M-m,2n+1)\right) && \text{(A.22)} \end{aligned}$$

If m is even and less than M:

$$\begin{aligned} w(m,n) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos\left(\pi\frac{m}{M}\left(l+\frac{1}{2}\right)\right) g(l-2na)\\ &= \frac{1}{\sqrt{2}}\left(c(m,2n)-c(2M-m,2n)\right)\\ w(m+M,n) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin\left(\pi\frac{m}{M}\left(l+\frac{1}{2}\right)\right) g(l-(2n+1)a)\\ &= \frac{i}{\sqrt{2}}\left(c(m,2n+1)+c(2M-m,2n+1)\right) \end{aligned}$$

If $m = M$ and M is even:

$$\begin{aligned} w(m+M,n) &= \sum_{l=0}^{L-1} f(l)(-1)^l g(l-(2n+1)a) && \text{(A.23)}\\ &= ic(M,2n+1) && \text{(A.24)} \end{aligned}$$

else if $m = M$ and M is odd:

$$\begin{aligned} w(m+M,n) &= \sum_{l=0}^{L-1} f(l)(-1)^l g(l-2na) && \text{(A.25)}\\ &= ic(M,2n) && \text{(A.26)} \end{aligned}$$

A.3.3 DWILTIII

This is the MDCT using Windows.

Let $c = C^{III}_{\left(g,a,\frac{b}{2}\right)} f$. We define $w = C^{W-III}_{(g,M)} f \in \mathbb{C}^{M\times N}$ by

For $m+n$ even:

$$\begin{aligned} w\left(m,n\right) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)l+\frac{\pi}{4}\right)g(l-na) && \text{(A.27)}\\ &= \frac{1}{\sqrt{2}}\left(e^{-i\pi/4}c\left(m,n\right)+e^{i\pi/4}c\left(M-1-m,n\right)\right) && \text{(A.28)}\end{aligned}$$

For $m+n$ odd:

$$\begin{aligned} w\left(m,n\right) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)l+\frac{\pi}{4}\right)g(l-na) && \text{(A.29)}\\ &= \frac{1}{\sqrt{2}}\left(e^{i\pi/4}c\left(m,n\right)+e^{-i\pi/4}c\left(M-1-m,n\right)\right) && \text{(A.30)}\end{aligned}$$

A.3.4 DWILTIV

This is the MDCT using HPE windows. It is the definition most commonly referred to as the MDCT.

Let $c = C^{IV}_{\left(g,a,\frac{b}{2}\right)} f$. We define $w = C^{W-IV}_{(g,M)} f \in \mathbb{C}^{M\times N}$ by

For $m+n$ even:

$$\begin{aligned} w\left(m,n\right) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)\left(l+\frac{1}{2}\right)+\frac{\pi}{4}\right)g(l-na) && \text{(A.31)}\\ &= \frac{1}{\sqrt{2}}\left(e^{-i\pi/4}c\left(m,n\right)+e^{-i\pi3/4}c\left(M-1-m,n\right)\right) && \text{(A.32)}\end{aligned}$$

For $m+n$ odd:

$$\begin{aligned} w\left(m,n\right) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)\left(l+\frac{1}{2}\right)+\frac{\pi}{4}\right)g(l-na) && \text{(A.33)}\\ &= \frac{1}{\sqrt{2}}\left(e^{i\pi/4}c\left(m,n\right)+e^{i\pi3/4}c\left(M-1-m,n\right)\right) && \text{(A.34)}\end{aligned}$$

Appendix B

Reference manual

B.1 The Linear Time-Frequency Analysis Toolbox.

B.1.1 Basic Fourier/TF analysis

B.1.1.1 DFT

Normalized Discrete Fourier Transform

Usage:

- `f = dft(f);`
- `f = dft(f,N,dim);`

This function computes a normalized discrete Fourier transform. This is nothing but a scaled version of the output from `FFT`. The function take exactly the same arguments as `FFT`. See the help on `FFT` for a throurough description.

B.1.1.2 IDFT

Inverse DFT

Usage:

- `f = idft(f);`
- `f = idft(f,N,dim);`

This function computes a normalized inverse discrete Fourier transform. This is nothing but a scaled version of the output from `IFFT`. The function takes exactly the same arguments as `IFFT`. See the help on `IFFT` for a throurough description.

B.1.1.3 INVOLUTE

Involution

Usage:

- `finv = involute(f);`
- `finv = involute(f,dim);`

`INVOLUTE(f)` will return the involution of `f`.

`INVOLUTE(f,dim)` will return the involution of `f` along dimension `dim`. This can for instance be used to calculate the 2D involution:

```
f=involute(f,1);
f=involute(f,2);
```

The involution `finv` of `f` is given by

```
    finv(l+1)=conj(f(mod(-l,L)+1));
```

for $l = 0, ..., L-1$.
The relation between conjugation, Fourier transformation and involution is expressed by

```
   conj(dft(f)) == dft(involute(f))
```

for all signals `f`. The inverse discrete Fourier transform can be expressed by

```
   idft(f) == conj(involute(dft(f)));
```

B.1.1.4 PCONV

Periodic convolution
Usage:

- `h = pconv(f,g)`
- `h = pconv(ptype,f,g);`

`PCONV(f,g)` computes the periodic convolution of `f` and `g`. The convolution is given by

$$h(l+1) = \sum_{k=0}^{L-1} f(k+1)\, g(l-k+1)$$

`PCONV('r',f,g)` computes the alternative where `g` is reversed given by

$$h(l+1) = \sum_{k=0}^{L-1} f(k+1)\, \overline{g(k-l+1)}$$

`PCONV('rr',f,g)` computes the alternative where both `f` and `g` are reversed given by

$$h(l+1) = \sum_{k=0}^{L-1} f(-k+1)\, g(l-k+1)$$

In the above formulas, $l-k$, $k-l$ and $-k$ are computed modulo L.

B.1.1.5 HERMBASIS

Orthonormal basis of discrete Hermite functions.
Usage:

- `V = hermbasis(L);`

`HERMBASIS(L)` will compute an orthonormal basis of discrete Hermite functions of length `L`. The vectors are returned as columns in the output.

All the vectors in the output are eigenvectors of the discrete Fourier transform, and resemble samplings of the continuous Hermite functions to some degree.

References: [80]

B.1.1.6 DSFT

Discrete Symplectic Fourier Transform

Usage:

- `C = dsft(F);`

`DSFT(F)` computes the discrete symplectic Fourier transform of `F`. `F` must be a matrix or a 3D array. If `F` is a 3D array, the transformation is applied along the first two dimensions.

Let `F` be a `K` x `L` matrix. Then the `DSFT` of `F` is given by

$$C(m+1,n+1) = \frac{1}{\sqrt{KL}} \sum_{l=0}^{L-1} \sum_{k=0}^{K-1} F(k+1,l+1)\, e^{2\pi i(kn/K - lm/L)}$$

for $m = 0, ..., L-1$ and $n = 0, ..., K-1$.

The `DSFT` is its own inverse.

B.1.1.7 SHAH

Discrete Shah-distribution

Usage:

- `f = shah(L,a);`

`SHAH(L,a)` computes the discrete, normalized Shah-distribution of length `L` with a distance of `a` between the spikes. The Shah distribution is defined by `f(n*a+1)=1/sqrt(L/a)` for integer `n`, otherwise `f` is zero.

This is also known as a pulse train or as the comb function, because the shape of the function resembles a comb. It is the sum of unit impulses ('diracs') with the distance a.

The `DFT` of `SHAH(L,a)` is `SHAH(L,L/a)`.

The Shah function has an extremely bad time-frequency localization. It does not generate a Gabor frame for any `L` and a.

B.1.1.8 PCHIRP

Periodic chirp

Usage:

- `g = pchirp(L,n);`

`PCHIRP(L,n)` returns a normalized, periodic, discrete chirp of length `L` that revolves `n` times around the frequency plane in frequency. `n` must be a whole number.

To get a chirp that revolves around the frequency plane in time, use

```
    dft(pchirp(L,N));
```

The chirp is computed by:

$$g(l+1) = \frac{1}{\sqrt{L}} e^{\pi i n l^2/L}, \quad l = 0, \ldots, L-1$$

References: [32]

B.1.1.9 ISEVEN

True if function is even

Usage:

- `t = iseven(f);`
- `t = iseven(f,centering);`
- `t = iseven(f,centering,tol);`

`ISEVEN(f)` returns 1 if `f` is whole point even. Otherwise it returns 0.

`ISEVEN(f,centering)` is true if `f` is even according to the value of centering: 0 indicates a whole-point even function and 0.5 indicates a half-point even function.

`ISEVEN(f,centering,tol)` does the same, using the tolerance `tol` to measure how large the error between the two parts of the vector can be.

B.1.1.10 MIDDLEPAD

Symmetrically zero-extends or cuts a function.

Usage:

- `h = middlepad(f,L);`

`MIDDLEPAD(f,L)` cuts or zero-extends `f` to length `L` by inserting zeros in the middle of the vector, or by cutting in the middle of the vector.

If `f` is whole-point even, `MIDDLEPAD(f,L)` will also be whole-point even.

If `f` has even length, then `f` will not be purely zero-extended, but the last element will be repeated once and multiplied by 1/2! That is, the support of `f` will increase by one!

`MIDDLEPAD` can be used to Fourier interpolate a vector. The following code will return an interpolation of `f` to length L:

```
    f=idft(middlepad(dft(f),L));
```

B.1.1.11 ZAK

Zak transform

Usage:

- `c = zak(f,a);`

`ZAK(f,a)` computes the Zak transform of `f` with parameter `a`. The coefficients are arranged in an `a` x `L/a` matrix, where `L` is the length of `f`.

If `f` is a matrix, then the transformation is applied to each column. This is then indexed by the third dimension of the output.

Assume that `c=ZAK(f,a)`, where `f` is a column vector of length L and `N=L/a`. Then the following holds for $m = 0, ..., a-1$ and $n = 0, ..., N-1$

$$c(m+1, n+1) \quad = \quad \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f(m-ka+1) e^{2\pi i n k/M}$$

References: [48], [17]

B.1.1.12 IZAK

Inverse Zak transform

Usage:

- `f = izak(c);`

`IZAK(c)` computes the inverse Zak transform of `c`. The parameter of the Zak transform is deduced from the size of `c`.

References: [48], [17]

B.1.2 Gabor systems.

B.1.2.1 DGT

Discrete Gabor transform.

Usage:

- `c = dgt(f,g,a,M);`
- `c = dgt(f,g,a,M,L);`
- `[c,Ls] = dgt(f,g,a,M);`
- `[c,Ls] = dgt(f,g,a,M,L);`

Input parameters:

- `f`: Input data
- `g`: Window function.
- `a`: Length of time shift.
- `M`: Number of modulations.
- `L`: Length of transform to do.

Output parameters:

- `c`: `M*N` array of coefficients.
- `Ls`: Length of input signal.

`DGT(f,g,a,M)` computes the Gabor coefficients of the input signal `f` with respect to the window `g` and parameters `a` and `M`. The output is a vector/matrix in a rectangular layout.

The length of the transform will be the smallest multiple of a and `M` that is larger than the signal. `f` will be zero-extended to the length of the transform. If `f` is a matrix, the transformation is applied to each column.

The length of the transform done can be obtained by `L=size(c,2)*a;`

`DGT(f,g,a,M,L)` computes the Gabor coefficients as above, but does a transform of length `L`. `f` will be cut or zero-extended to length `L` before the transform is done.

`[c,Ls]=DGT(f,g,a,M)` or `[c,Ls]=DGT(f,g,a,M,L)` additionally returns the length of the input signal `f`. This is handy for reconstruction:

```
[c,Ls]=dgt(f,g,a,M);
fr=idgt(c,gd,a,Ls);
```

will reconstruct the signal `f` no matter what the length of `f` is, provided that `gd` is a dual window of `g`.

The Discrete Gabor Transform is defined as follows: Consider a window `g` and a one-dimensional signal `f` of length `L` and define $N = L/a$. The output from `c=DGT(f,g,a,M)` is then given by

$$c\left(m+1,n+1\right)=\sum_{k=0}^{L-1} f(k)e^{-2\pi imk/M}\overline{g(k-an+1)}$$

where $m = 0, ..., M-1$ and $n = 0, ..., N-1$.

References: [35], [41]

B.1.2.2 IDGT

Inverse discrete Gabor transform.

Usage:

- `f = idgt(c,g,a);`
- `f = idgt(c,g,a,Ls);`

Input parameters:

- `c`: Array of coefficients.
- `g`: Window function.
- `a`: Length of time shift.
- `Ls`: length of signal.

Output parameters:

- `f`: Signal.

`IDGT(c,g,a)` computes the Gabor expansion of the input coefficients `c` with respect to the window `g` and time shift `a`. The number of channels is deduced from the size of the coefficients `c`.

`IDGT(c,g,a,Ls)` does as above but cuts or extends `f` to length `Ls`.

For perfect reconstruction, the window used must be a dual window of the one used to generate the coefficients.

Assume that `f=IDGT(c,g,a,L)` for an array `c` of size M x N. Then the following holds for `k=0,...,L-1`:

$$f(l+1) = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} c(m+1,n+1)e^{2\pi iml/M}g(l-an+1)$$

B.1.2.3 DGT2

2-D Discrete Gabor transform.

Usage:

- `c = dgt2(f,g,a,M);`
- `c = dgt2(g,g1,g2,[a1,a2],[M1,M2]);`
- `c = dgt2(g,g1,g2,[a1,a2],[M1,M2],[L1,L2]);`
- `[c,Ls] = dgt2(g,g1,g2,[a1,a2],[M1,M2]);`
- `[c,Ls] = dgt2(g,g1,g2,[a1,a2],[M1,M2],[L1,L2]);`

Input parameters:

- `f`: Input data, matrix.
- `g,g1,g2`: Window functions.
- `a,a1,a2`: Length of time shifts.
- `M,M1,M2`: Number of modulations.
- `L1,L2`: Length of transform to do

Output parameters:

- `c`: array of coefficients.
- `Ls`: Original size of input matrix.

`DGT2(f,g,a,M)` will calculate a separable two dimensional discrete Gabor transformation of the input signal `f` with respect to the window `g` and parameters `a` and `M`.

For each dimension, the length of the transform will be the smallest possible that is larger than the length of the signal along that dimension. `f` will be appropriately zero-extended.

`DGT2(f,g,a,M,L)` computes a Gabor transform as above, but does a transform of length L along each dimension. `f` will be cut or zero-extended to length L before the transform is done.

`[c,Ls]=DGT2(f,g,a,M)` or `[c,Ls]=DGT2(f,g,a,M,L)` additionally returns the length of the input signal `f`. This is handy for reconstruction:

```
[c,Ls]=dgt2(f,g,a,M);
fr=idgt2(c,gd,a,Ls);
```

will reconstruct the signal `f` no matter what the size of `f` is, provided that `gd` is a dual window of `g`.

`DGT2(f,g1,g2,a,M)` makes it possible to use a different window along the two dimensions.

The parameters `a, M`, L and `Ls` can also be vectors of length 2. In this case the first element will be used for the first dimension and the second element will be used for the second dimension. For perfect reconstruction,

The output `c` will be have 4 or 5 dimensions. The dimensions index the following properties:

1	Number of translation along 1st dimension of input.
2	Number of channel along 1st dimension of input
3	Number of translation along 2nd dimension of input.
4	Number of channel along 2nd dimension of input
5	Plane number, corresponds to 3rd dimension of input.

B.1.2.4 IDGT2

2D Inverse discrete Gabor transform.

Usage:

- `f = idgt2(c,g,a);`
- `f = idgt2(c,g1,g2,a);`
- `f = idgt2(c,g1,g2,[a1 a2]);`
- `f = idgt2(c,g,a,Ls);`
- `f = idgt2(c,g1,g2,a,Ls);`
- `f = idgt2(c,g1,g2,[a1 a2],Ls);`

Input parameters:

- `c`: Array of coefficients.
- `g,g1,g2`: Window function(s).
- `a,a1,a2`: Length(s) of time shift.
- `Ls`: Length(s) of reconstructed signal (optional).

Output parameters:

- `f`: Output data, matrix.

`IDGT2(c,g,a,M)` will calculate a separable two dimensional inverse discrete Gabor transformation of the input coefficients `c` using the window `g` and parameters `a`, along each dimension. The number of channels is deduced from the size of the coefficients `c`.

`IDGT2(c,g1,g2,a)` will do the same using the window `g1` along the first dimension, and window `g2` the second dimension.

`IDGT2(c,g,a,Ls)` or `IDGT2(c,g1,g2,a,Ls)` will cut the signal to size `Ls` after the transformation is done.

The parameters `a` and `Ls` can also be vectors of length 2. In this case the first element will be used for the first dimension and the second element will be used for the second dimension.

B.1.3 Window construction for Gabor frames.

B.1.3.1 CANDUAL

Canonical dual window.

Usage:

- `gamma = candual(g,a,M);`
- `gamma = candual(g,a,M,L);`

Input parameters:

- `g`: Gabor window.
- `a`: Length of time shift.
- `M`: Number of channels.
- `L`: Length of window. (optional)

Output parameters:

- `gamma`: Canonical dual window.

`CANDUAL(g,a,M)` computes the canonical dual window of the discrete Gabor frame with window `g` and parameters `a, M`.

If `L` is specified, then the window will be padded or truncated to length `L` before the canonical dual window is calculated. `gamma` will also have length `L`.

If `a>M` then the dual window of the Gabor Riesz sequence with window `g` and parameters `a` and `M` will be calculated.

B.1.3.2 CANTIGHT

Canonical tight window.

Usage:

- `gamma = cantight(a,M,L);`
- `gamma = cantight(g,a,M);`
- `gamma = cantight(g,a,M,L);`

Input parameters:

- `g`: Gabor window.
- `a`: Length of time shift.
- `M`: Number of modulations.
- `L`: Length of window. (optional)

Output parameters:

- `gamma`: Canonical tight window, column vector.

`CANTIGHT(a,M,L)` computes a nice tight window of length `L` for a lattice with parameters `a, M`.

`CANTIGHT(g,a,M)` computes the canonical tight window of the Gabor frame with window `g` and parameters `a, M`.

`CANTIGHT(g,a,M,L)` pads or truncates `g` to length `L` before calculating the tight window. `gamma` will also be of length `L`.

If `a>M` then an orthonormal window of the Gabor Riesz sequence with window `g` and parameters `a` and `M` will be calculated.

B.1.3.3 PROJDUAL

Dual window by projection.

Usage:

- `gd = projdual(gm,g,a,M)`
- `gd = projdual(gm,g,a,M,L)`

Input parameters:

- `gm`: Window to project.
- `g`: Window function.
- `a`: Length of time shift.
- `M`: Number of modulations.
- `L`: Total length of vectors (optional).

Output parameters:

- `gd`: Dual window.

`PROJDUAL(gm,g,a,M)` calculates the dual window of the Gabor frame given by `g`, `a` and `M` closest to `gm` measured in the l^2 norm.

`PROJDUAL(gm,g,a,M,L)` first symmetrically extends the windows `g` and `gm` to length `L`.

B.1.3.4 MIXDUAL

Computes the mixdual of g1

Usage:

- `gamma = mixdual(g1,g2,a,M)`

Input parameters:

- `g1`: Window 1
- `g2`: Window 2
- `a`: Length of time shift.
- `M`: Number of modulations.

Output parameters:

- `gammaf`: Mixdual of window 1.

`MIXDUAL(g1,g2,a,M)` computes a dual window of `g1` from a mix of the canonical dual windows of `g1` and `g2`.

References: [106]

B.1.4 Wilson bases and MDCT.

B.1.4.1 DWILT

Discrete Wilson transform.

Usage:

- `c = dwilt(f,g,M);`
- `c = dwilt(f,g,M,L);`
- `[c,Ls] = dwilt(f,g,M);`
- `[c,Ls] = dwilt(f,g,M,L);`

Input parameters:

- `f`: Input data
- `g`: Window function.
- `M`: Number of bands.
- `L`: Length of transform to do.

Output parameters:

- `c`: 2*M x N array of coefficients.
- `Ls`: Length of input signal.

DWILT(f,g,M) computes a discrete Wilson transform with M bands and window g.

The length of the transform will be the smallest possible that is larger than the signal. f will be zero-extended to the length of the transform. If f is a matrix, the transformation is applied to each column.

g must be whole-point even.

DWILT(f,g,M,L) computes the Wilson transform as above, but does a transform of length L. f will be cut or zero-extended to length L before the transform is done.

[c,Ls]=DWILT(f,g,M) or [c,Ls]=DWILT(f,g,M,L) additionally return the length of the input signal f. This is handy for reconstruction:

```
[c,Ls]=dwilt(f,g,M);
fr=idwilt(c,gd,M,Ls);
```

will reconstruct the signal f no matter what the length of f is, provided that gd is a dual Wilson window of g.

A Wilson transform is also known as a maximally decimated, even-stacked cosine modulated filter bank.

Assume that the following code has been executed for a column vector f:

```
c=dwilt(f,g,M);  % Compute a Wilson transform of f.
N=size(c,2)*2;   % Number of translation coefficients.
```

The following holds for $m = 0, ..., M-1$ and $n = 0, ..., N/2-1$:

If $m = 0$:

$$c(1, n+1) = \sum_{l=0}^{L-1} f(l+1) g(l - 2nM + 1)$$

If m is odd and less than M

$$\begin{aligned} c(m+1, n+1) &= \sqrt{2}\sum_{l=0}^{L-1} f(l+1)\sin(\pi\frac{m}{M}l)g(l-2nM+1) \\ c(m+M+1, n+1) &= \sqrt{2}\sum_{l=0}^{L-1} f(l+1)\cos(\pi\frac{m}{M}l)g(l-(2n+1)M+1) \end{aligned}$$

If m is even and less than M

$$\begin{aligned} c(m+1, n+1) &= \sqrt{2}\sum_{l=0}^{L-1} f(l+1)\cos(\pi\frac{m}{M}l)g(l-2nM+1) \\ c(m+M+1, n+1) &= \sqrt{2}\sum_{l=0}^{L-1} f(l+1)\sin(\pi\frac{m}{M}l)g(l-(2n+1)M+1) \end{aligned}$$

if $m = M$ and `M` is even:

$$c\,(M+1, n+1) \;=\; \sum_{l=0}^{L-1} f(l+1)(-1)^l g(l-2nM+1)$$

else if $m = M$ and `M` is odd

$$c\,(M+1, n+1) \;=\; \sum_{k=0}^{L-1} f(l+1)(-1)^l g\,(l-(2n+1)\,M+1)$$

References: [15], [67]

B.1.4.2 IDWILT

Inverse discrete Wilson transform.

Usage:

- `f = idwilt(c,g);`
- `f = idwilt(c,g,Ls);`

Input parameters:

- `c`: `M*N` array of coefficients.
- `g`: Window function.
- `Ls`: Final length of function (optional)

Output parameters:

- `f`: Input data

`IDWILT(c,g)` computes an inverse discrete Wilson transform with window `g`. The number of channels is deduced from the size of the coefficient array `c`.

`g` must be whole-point even.

`IDWILT(f,g,Ls)` does the same, but cuts of zero-extend the final result to length `Ls`.

References: [15], [67]

B.1.4.3 DWILT2

2-D Discrete Wilson transform.

Usage:

- `c = dwilt2(f,g,M);`

- `c = dwilt2(f,g1,g2,[M1,M2]);`
- `c = dwilt2(f,g1,g2,[M1,M2],[L1,L2]);`
- `[c,L] = dwilt2(f,g1,g2,[M1,M2],[L1,L2]);`

Input parameters:

- `f`: Input data, matrix.
- `g,g1,g2`: Window functions.
- `M,M1,M2`: Number of bands.
- `L1,L2`: Length of transform to do.

Output parameters:

- `c`: array of coefficients.
- `Ls`: Original size of input matrix.

`DWILT2(f,g,M)` calculates a two dimensional discrete Wilson transform of the input signal `f` using the window `g` and parameter `M` along each dimension.

For each dimension, the length of the transform will be the smallest possible that is larger than the length of the signal along that dimension. `f` will be appropriatly zero-extended.

All windows must be whole-point even.

`DWILT2(f,g,M,L)` computes a Wilson transform as above, but does a transform of length `L` along each dimension. `f` will be cut or zero-extended to length `L` before the transform is done.

`[c,Ls]=DWILT(f,g,M)` or `[c,Ls]=DWILT(f,g,M,L)` additionally returns the length of the input signal `f`. This is handy for reconstruction:

`c=DWILT2(f,g1,g2,M)` makes it possible to use a different window along the two dimensions.

The parameters `L`, `M` and `Ls` can also be vectors of length 2. In this case the first element will be used for the first dimension and the second element will be used for the second dimension.

The output `c` will be have 4 or 5 dimensions. The dimensions index the following properties:

1	Number of translation along 1st dimension of input.
2	Number of channel along 1st dimension of input
3	Number of translation along 2nd dimension of input.
4	Number of channel along 2nd dimension of input
5	Plane number, corresponds to 3rd dimension of input.

B.1.4.4 IDWILT2

2D Inverse Discrete Wilson transform.

Usage:

- `f = idwilt2(c,g);`
- `f = idwilt2(c,g1,g2);`
- `f = idwilt2(c,g1,g2,Ls);`

Input parameters:

- `c`: Array of coefficients.
- `g,g1,g2`: Window functions.
- `Ls`: Size of reconstructed signal.

Output parameters:

- `f`: Output data, matrix.

`IDWILT2(c,g)` will calculate a separable two dimensional inverse discrete Wilson transformation of the input coefficients `c` using the window `g`. The number of channels is deduced from the size of the coefficients `c`.

`IDWILT2(c,g1,g2)` will do the same using the window `g1` along the first dimension, and window `g2` the second dimension.

`IDWILT2(c,g1,g2,Ls)` will cut the signal to size `Ls` after the transformation is done.

B.1.4.5 MDCT

Modified Discrete Cosine Transform

Usage:

- `c = mdct(f,g,M);`
- `c = mdct(f,g,M,L);`
- `[c,Ls] = mdct(f,g,M);`
- `[c,Ls] = mdct(f,g,M,L);`

Input parameters:

- `f`: Input data
- `g`: Window function.
- `M`: Number of bands.

- `L`: Length of transform to do.

Output parameters:

- `c`: 2*M x N array of coefficients.
- `Ls`: Length of input signal.

`MDCT(f,g,M)` computes a Modified Discrete Cosine Transform with `M` bands and window `g`.

The length of the transform will be the smallest possible that is larger than the signal. `f` will be zero-extended to the length of the transform. If `f` is a matrix, the transformation is applied to each column. `g` must be whole-point even.

`MDCT(f,g,M,L)` computes the `MDCT` transform as above, but does a transform of length `L`. `f` will be cut or zero-extended to length `L` before the transform is done.

`[c,Ls]=MDCT(f,g,M)` or `[c,Ls]=MDCT(f,g,M,L)` additionally return the length of the input signal `f`. This is handy for reconstruction:

```
[c,Ls]=mdct(f,g,M);
fr=imdct(c,gd,M,Ls);
```

will reconstuct the signal `f` no matter what the length of `f` is, provided that `gd` is a dual Wilson window of `g`.

The `MDCT` is sometimes known as an odd-stacked cosine modulated filter bank. The `MDCT` defined by this routine is slightly different from the most common definition of the `MDCT`, in order to be able to use the same window functions as the Wilson transform.

Assume that the following code has been executed for a column vector `f` of length L:

```
c=mdct(f,g,M);  % Compute the MDCT of f.
N=size(c,2);    % Number of translation coefficients.
```

The following holds for $m = 0, ..., M-1$ and $n = 0, ..., N-1$:

If $m+n$ is even:

$$c\,(m+1,n+1) \quad = \quad \sqrt{2}\sum_{l=0}^{L-1} f(l+1)\cos\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)l+\frac{\pi}{4}\right)g(l-nM+1)$$

If $m+n$ is odd:

$$c\,(m+1,n+1) \quad = \quad \sqrt{2}\sum_{l=0}^{L-1} f(l+1)\sin\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)l+\frac{\pi}{4}\right)g(l-nM+1)$$

References: [81], [82], [72], [16]

B.1.4.6 IMDCT

Inverse MDCT

Usage:

- `f = imdct(c,g);`
- `f = imdct(c,g,Ls);`

Input parameters:

- `c`: `M*N` array of coefficients.
- `g`: Window function.
- `Ls`: Final length of function (optional)

Output parameters:

- `f`: Input data

`IMDCT(c,g)` computes an inverse `MDCT` with window `g`. The number of channels is deduced from the size of the coefficient array `c`.

`g` must be whole-point even.

`IMDCT(f,g,Ls)` does the same, but cuts or zero-extend the final result to length `Ls`.

References: [81], [82], [72], [16]

B.1.4.7 MDCT2

2-D Discrete Wilson transform.

Usage:

- `c = mdct2(f,g,M);`
- `c = mdct2(f,g1,g2,[M1,M2]);`
- `c = mdct2(f,g1,g2,[M1,M2],[L1,L2]);`
- `[c,L] = mdct2(f,g1,g2,[M1,M2],[L1,L2]);`

Input parameters:

- `f`: Input data, matrix.
- `g,g1,g2`: Window functions.
- `M,M1,M2`: Number of bands.
- `L1,L2`: Length of transform to do.

Output parameters:

- `c`: array of coefficients.
- `Ls`: Original size of input matrix.

`MDCT2(f,g,M)` calculates a two dimensional Modified Discrete Cosine transform of the input signal `f` using the window `g` and parameter `M` along each dimension.

For each dimension, the length of the transform will be the smallest possible that is larger than the length of the signal along that dimension. `f` will be appropriatly zero-extended.

All windows must be whole-point even.

`MDCT2(f,g,M,L)` computes a 2-D `MDCT` as above, but does a transform of length `L` along each dimension. `f` will be cut or zero-extended to length `L` before the transform is done.

`[c,Ls]=MDCT(f,g,M)` or `[c,Ls]=MDCT(f,g,M,L)` additionally returns the length of the input signal `f`. This is handy for reconstruction:

`c=MDCT2(f,g1,g2,M)` makes it possible to use a different window along the two dimensions.

The parameters `L`, `M` and `Ls` can also be vectors of length 2. In this case the first element will be used for the first dimension and the second element will be used for the second dimension.

The output `c` will be have 4 or 5 dimensions. The dimensions index the following properties:

1	Number of translation along 1st dimension of input.
2	Number of channel along 1st dimension of input
3	Number of translation along 2nd dimension of input.
4	Number of channel along 2nd dimension of input
5	Plane number, corresponds to 3rd dimension of input.

B.1.4.8 IMDCT2

2D Inverse Discrete Wilson transform.

Usage:

- `f = imdct2(c,g);`
- `f = imdct2(c,g1,g2);`
- `f = imdct2(c,g1,g2,Ls);`

Input parameters:

- `c`: Array of coefficients.
- `g,g1,g2`: Window functions.
- `Ls`: Size of reconstructed signal.

Output parameters:

- `f`: Output data, matrix.

`IMDCT2(c,g)` will calculate a separable two dimensional inverse `MDCT` transformation of the input coefficients `c` using the window `g`. The number of channels is deduced from the size of the coefficients `c`.

`IMDCT2(c,g1,g2)` will do the same using the window `g1` along the first dimension, and window `g2` the second dimension.

`IMDCT2(c,g1,g2,Ls)` will cut the signal to size `Ls` after the transformation is done.

B.1.4.9 WILORTH

Wilson orthonormal window.

Usage:

- `gamma = wilorth(M,L);`
- `gamma = wilorth(g,M);`
- `gamma = wilorth(g,M,L);`

Input parameters:

- `g`: Auxiliary window window function (optional).
- `M`: Number of modulations.
- `L`: Length of window (optional).

Output parameters:

- `gamma`: Window generating an orthonormal Wilson basis.

`gamma=WILORTH(M,L)` computes a nice window of length `L` generating a Wilson or `MDCT` orthonormal basis with `M` frequency bands for signal of length `L`.

`gamma=WILORTH(g,M)` creates an orthonormal window from the window `g`, `g` must be whole-point even.

`gamma=WILORTH(g,M,L)` pads or truncates `g` to length `L` before calculating the orthonormal window. `gamma` will also be of length `L`.

B.1.4.10 WILDUAL

Canonical dual window.

Usage:

- `gamma = wildual(g,M);`
- `gamma = wildual(g,M,L);`

Input parameters:

- `g`: Gabor window.
- `M`: Number of modulations.
- `L`: Length of window. (optional)

Output parameters:

- `gamma`: Canonical dual window.

`WILDUAL(g,M)` returns the dual window of the Wilson or `MDCT` basis with window `g`, parameter `M` and length equal to the length of the window `g`.

`WILDUAL(g,M,L)` does the same, but now `L` is used as the length of the Wilson basis. `g` will be cut or zero-extended to length `L`.

The input window `g` must be whole-point even.

B.1.5 Window functions.

B.1.5.1 PGAUSS

Sampled, periodized Gaussian.

Usage:

- `g = pgauss(L);`
- `g = pgauss(L,tfr);`
- `g = pgauss(L,tfr,cent);`

Input parameters:

- `L`: Length of vector.
- `tfr`: ratio between time and frequency support.
- `cent`: Centering.

Output parameters:

- `g`: The periodized Gaussian(s).

`PGAUSS(L,tfr)` computes samples of a periodized Gaussian. The function returns a regular sampling of the periodization of the function $e^{-\pi x^2}$.

The returned function has norm == 1.

The parameter `tfr` determines the ratio between the effective support of `g` and the effective support of the `DFT` of `g`. If `tfr>1` then `g` has a wider support than the `DFT` of `g`.

`PGAUSS(L)` does the same setting `tfr=1`.

The function is whole-point even. This implies that fft(pgauss(L,tfr)) is real for any `L` and `tfr`.

`PGAUSS(L,tft,cent)` will generate a differently centered Gaussian. Setting `cent=0` generates a whole point centered function (as above) and setting `cent=.5` generates a half point centered function, as most other Matlab filter routines.

If this function is used to generate a window for a Gabor frame, then the optimal window is generated by `PGAUSS(L,a*M/L);`

References: [71]

B.1.5.2 PSECH

Sampled, periodized hyperbolic secant.

Usage:

- `g = psech(L);`
- `g = psech(L,tfr);`

Input parameters:

- `L`: Length of vector.
- `tfr`: ratio between time and frequency support.

Output parameters:

- `g`: The periodized Gaussian(s).

`PSECH(L,tfr)` computes samples of a periodized hyperbolic secant. The function returns a regular sampling of the periodization of the function sech(pi*x)

The returned function has norm == 1.

The parameter `tfr` determines the ratio between the effective support of `g` and the effective support of the `DFT` of `g`. If `tfr>1` then `g` has a wider support than the `FFT` of `g`.

`PSECH(L)` does the same setting `tfr=1`.

If `tfr` is a vector, `PSECH` will return a matrix, where each column is a `PSECH` function with the corresponding value of `tfr`.

The function is whole-point even. This implies that fft(psech(L,tfr)) is real for any `L` and `tfr`.

If this function is used to generate a window for a Gabor frame with a rectangular lattice, then the optimal window is generated by `g=psech(L,a/b);`

References: [59]

B.1.5.3 PHERM

Periodized Hermite function.

Usage:

- `g = pherm(L,order);`
- `g = pherm(L,order,tfr);`

Input parameters:

- `L`: Length of vector.
- `order`: Order of Hermite function.
- `tfr`: ratio between time and frequency support.

Output parameters:

- `g`: The periodized Gaussian(s).

`PHERM(L,order,tfr)` computes samples of a periodized Hermite function of order `order`. order is counted from 0, so the zeroth order Hermite function is the Gaussian.

The returned functions are eigenvectors of the `DFT`. The first four Hermite functions are orthonormal, but in general they are not.

The parameter `tfr` determines the ratio between the effective support of `g` and the effective support of the `DFT` of `g`. If `tfr>1` then `g` has a wider support than the `DFT` of `g`.

`PHERM(L,order)` does the same setting `tfr=1`.

If `order` is a vector, `DHERM` will return a matrix, where each column is a Hermite function with the corresponding order.

If `tfr` is a vector, `DHERM` will return a matrix, where each column is a Hermite function with the corresponding `tfr`.

If both `order` and `tfr` are vectors, they must have the same length, and the values will be paired.

If this function is used to generate a window for a Gabor frame with a rectangular lattice with lattice constants a and b, then the optimally centered window is generated by `PSECH(L,order,a/b)`; Note that this window does not generate a frame unless `order` is a factor of 4. Use the functions for construction of multi-window Gabor frames.

Use this function only for lower orders. For orders above 66 the result can not be trusted due to numerical errors.

B.1.5.4 PBSPLINE

Periodized B-spline.

Usage:

- `g = pbspline(L,order,a);`
- `g = pbspline(stype,order,a);`
- `[g,nlen] = pbspline(L,order,a);`
- `[g,nlen] = pbspline(stype,L,order,a);`
- `[g,nlen] = pbspline(stype,L,order,a,cent);`

Input parameters:

- `stype`: Type of spline.
- `L`: Length of window.
- `order`: Order of B-spline.
- `a`: Time-shift parameter for partition of unity.
- `cent`: Centering (0 or 0.5)

Output parameters:

- `g`: Fractional B-spline.
- `nlen`: Number of non-zero elements in out.

`PBSPLINE(L,order,a)` computes a (slightly modified) B-spline of order `order` of total length `L`.

If shifted by the distance `a`, the returned function will form a partition of unity. The result is normalized such that the functions sum to 1/sqrt(a).

`PBSPLINE(stype,L,order,a)` will compute a spline of type `stype`. `stype` can be one of:

'+d'	Asymmetric discrete fractional spline.
'ed'	Even discrete fractional spline.
'xd'	Symmetric 'flat' discrete fractional spline.
'*d'	Symmetric 'pointy' discrete fractional spline
'+c'	Asymmetric fractional spline by sampling.
'ec'	Even fractional spline by sampling.
'xc'	Symmetric 'flat' fractional spline by sampling.
'*c'	Symmetric 'pointy' fractional spline by sampling.

The different types are accurately described in the referenced paper. Generally, the 'd' types of splines are very fast to compute, while the 'c' types are samplings of the continuous splines. The 'e' types coincides with the regular B-splines for integer orders. The 'x' types do not coincide, but generate Gabor frames with favorable frame bounds. The default type is 'ed' to guarantee fast computation are a familiar shape of the splines.

`PBSPLINE(stype,L,order,a,cent)` will generate differently centered splines. Setting `cent=0` generates a whole point centered function and setting `cent=.5` generates a half point centered function, as most other Matlab filter routines. Default is `cent=0`.

`[out,nlen]=PBSPLINE(stype,L,order,a)` will additionally compute the number of non-zero elements in out.

If `nlen > L`, the function returned will be a periodization of a B-spline.

If `nlen < L`, you can choose to remove the additional zeros by calling `MIDDLEPAD(g,nlen)`

References: [94]

B.1.5.5 FIRWIN

FIR window

Usage:

- `g = firwin(name,M);`
- `g = firwin(name,M,centering);`

`FIRWIN(name,M)` will compute a two overlapping window of length `M` of type `name`.

`FIRWIN(name,M,centering)` will create a window centered as specified by `centering`. The default `(centering=0)` is to return a whole-point centered window, while a value of .5 will produce a half-point even function (traditional signal processing style).

The windows are normalized, such that if used for Gabor systems with parameters `a=M/2` and `M`, they will generate Gabor frames with framebounds close to 1.

All windows are symmetric and can be used for Wilson bases, except when noted otherwise.

The windows available are:

hanning	Hanning window. Forms a PU.
sqrthan	Square root of a Hanning window. Normalized so it generates a normalized tight Gabor system with parameters `a=M/2` and `M`
hamming	Hamming window. Forms a PU. This window cannot be used for a Wilson basis.
sqrtham	Square root of a Hamming window. As sqrthan.
blackman	Blackman window
vorbis	Iterated sine window. This is a tight window.

References: [78]

B.1.5.6 FIRKAISER

Kaiser-Bessel window

Usage:

- `g = firkaiser(M,beta);`
- `g = firkaiser(M,beta,centering);`

`FIRKAISER(M,beta)` computes the `Kaiser-Bessel` window of length `M` with parameter `beta`.

`FIRKAISER(M,beta,'zero')` will set the smallest element of the window to zero. This makes it possible to use the window for a Wilson basis.

`FIRKAISER(M,beta,centering)` will create a window centered as specified by `centering`. The default `(centering=0)` is to return a whole-point centered window, while a value of .5 will produce a half-point even function (traditional signal processing style).

References: [78]

B.1.5.7 FIREXTEND

Extend FIR window to IIR

Usage:

- `g = firextend(g,L);`

`FIREXTEND(g,L)` will extend the `FIR` window `g` to a length `L` window by inserting zeros. Note that this is a slightly different behaviour than `MIDDLEPAD`.

B.1.6 Time Varying Filtering.

B.1.6.1 GABMUL

Gabor multiplier.

Usage:

- `h = gabmul(f,c,a);`
- `h = gabmul(f,c,g,a);`
- `h = gabmul(f,c,ga,gs,a);`

Input parameters:

- `f`: Input signal

- `c`: symbol of Gabor multiplier
- `g`: analysis/synthesis window
- `ga`: analysis window
- `gs`: synthesis window
- `a`: Length of time shift.

Output parameters:

- `h`: Output signal

`GABMUL(f,c,g,a)` will filter `f` by a Gabor multiplier determined by the symbol `c` over the rectangular time-frequency lattice determined by `a` and `M`, where `M` is deduced from the size of `c`. The window `g` will be used for both analysis and synthesis.

`GABMUL(f,c,a)` will do the same using an optimally concentrated, tight Gaussian as window function.

`GABMUL(f,c,gs,ga,a)` will do the same using the window `ga` for analysis and `gs` for synthesis.

B.1.7 Conditions numbers.

B.1.7.1 GFBOUNDS

Calculate frame bounds of Gabor frame.

Usage:

- `fcond = gfbounds(g,a,M);`
- `[A,B] = gfbounds(g,a,M);`
- `[A,B] = gfbounds(g,a,M,L);`

Input parameters:

- `g`: The window function.
- `a`: Length of time shift.
- `M`: Number of modulations.
- `L`: Length of transform to consider.

Output parameters:

- `fcond`: Frame condition number `(B/A)`
- `A,B`: Frame bounds.
-

`GFBOUNDS(g,a,M)` calculates the ratio `B/A` of the frame bounds of the Gabor frame with window `g`, and parameters `a, M`.

`[A,B]=GFBOUNDS(g,a,M)` calculates the frame bounds `A` and `B` of the Gabor frame with window `g`, and parameters `a, M`.

If the optional parameter `L` is specified, the window is cut or zero-extended to length `L`.

B.1.7.2 WILBOUNDS

Calculate frame bounds of Wilson basis.

Usage:

- `[AF,BF] = wilbounds(g,M)`
- `[AF,BF] = wilbounds(g,M,L)`

Input parameters:

- `g`: Window function.
- `M`: Number of channels.
- `L`: Length of transform to do (optional)

Output parameters:

- `A,B`: Frame bounds.
-

`WILBOUNDS(g,M)` calculates the frame bounds of the Wilson frame operator of the Wilson basis with window `g` and `M` channels.

If the optional parameter `L` is specified, the window is cut or zero-extended to length `L`.

B.1.7.3 GFDUALNORM

Measure of how close a window is to be a dual window.

Usage:

- `dn = gfdualnorm(g,gamma,a,M);`
- `dn = gfdualnorm(g,gamma,a,M,L);`
- `[scal,res] = gfdualnorm(g,gamma,a,M);`
- `[scal,res] = gfdualnorm(g,gamma,a,M,L);`

Input parameters:

- `gamma`: input window..
- `g`: window function.
- `a`: Length of time shift.
- `M`: Number of modulations.
- `L`: Length of transform to consider

Output parameters:

- `dn`: dual norm.
- `scal`: Scaling factor
- `res`: Residual

`GFDUALNORM(g,gamma,a,M)` calculates how close `gamma` is to be a dual window of the Gabor frame with window `g` and parameters `a` and `M`.

`GFDUALNORM(g,gamma,a,M,L)` does the same, but considers a transform length of `L`.

`[scal,res]=GFDUALNORM(g,gamma,a,M)` or `[scal,res]=GFDUALNORM(g,gamma,a,M,L)` will compute two entities: `scal` determines if the windows are scaled correctly, it must be 1 for the windows to be dual. `res` is close to zero if the windows (scaled correctly) are dual windows.

`GFDUALNORM` can be used to get the maximum relative reconstruction error when using the two specified windows. Consider the following code for some signal f, windows `g`, `gamma`, parameters `a` and `M` and transform-length `L` (See help on `DGT` for how to obtain L):

```
fr=idgt(dgt(f,g,a,M),gamma,a);
er=norm(f-fr)/norm(f);
eest=gfdualnorm(g,gamma,a,M,L);
```

Then `er < eest` for all possible input signals f.

To get a similar estimate for a tight window gt, simply use

```
eest=gfdualnorm(gt,gt,a,M,L);
```

B.1.8 Plots.

B.1.8.1 SGRAM

Spectrogram.

Usage:

- `sgram(f,op1,op2, ... );`

`SGRAM(f)` plots a spectrogram of `f` using a `DGT`.

Additional arguments can be supplied like this: `SGRAM(f,'nf','log')`. The arguments are character strings, and the available options (so far) are:

'nf'	Display negative frequencies, with the zero centered in the middle. For real signals, this will just mirror the upper half plane. This is standard for complex signals.
'tc'	Time centering. Move the beginning of the signal to the middle of the plot. This is useful for visualizing the window functions of the toolbox.
'db'	Apply 20*log10 to the coefficients. This makes it possible to see very weak phenomena, but it might show to much noise. A logarithmic scale is more adapted to perception of sound. This is the default of Matlab's `SPECGRAM`.
'vgg'	Use the signal itself as window. In LaTeX, this is the symbol V_gg This is a quadratic time-frequency distribution related to the `Wigner-Ville` distribution.
'contour'	Do a contour plot instead of an image.
'surf'	Do a surf plot instead of an image.

In Octave, the default colormap is greyscale. Change it to `colormap(jet)` for something prettier.

B.1.8.2 PHASEPLOT

Phase plot

Usage:

- `phaseplot(f,op1,op2, ... );`

`PHASEPLOT(f)` plots the phase of `f` using a `DGT`.

Additional arguments can be supplied like this: `PHASEPLOT(f,'nf','log')`. The arguments are character strings, and the available options (so far) are:

'nf'	Display negative frequencies, with the zero centered in the middle. For real signals, this will just mirror the upper half plane. This is standard for complex signals.
'tc'	Time centering. Move the beginning of the signal to the middle of the plot. This is usefull for visualizing the window functions of the toolbox.
'vgg'	Use the signal itself as window. In LaTeX, this is the symbol V_gg
'thr'	Use the amplitude values to threshold the phase values. For small amplitude values the phase values are. This is because an error, that is small in amplitude can still lead to arbitrary phase values in the vicinity of zero. Setting this flag will set the phase is to a constant value (0) for all pairs (m,n), where the amplitude is below a certain relative value, set to 0.001 of the maximum amplitude.
'phl'	Different to dgt.m the Gabor transform can be defined by shifting the order of translation and modulation. This is sometimes used in algorithms to be able to directly apply the `FFT` without circular shifts, as for every time point the `FFT` can start from zero. Obviously this choice has a big influence on the development of the phase of the coefficients over time.

In Octave, the default colormap is greyscale. Change it to `colormap(jet)` for something prettier.

Obviously in the spectrogram the information about the complex angle of the Gabor coefficients, i.e. the phase, gets lost. To plot the phase some issues have to be considered:

- phase unwrapping: The angle function is non-continuous on a half-axis in the complex plane. So there are continuous complex function, whose angle is not continuous, e.g. a periodic rotation around zero. A work-around is to add integer multiples of 2 pi to the result, the so called 'unwrapping'. For plots a continuous effect can just be reached by choosing a fitting color scheme.

References: [19]

B.1.8.3 MAGRESP

Magnitude response plot of TF-transform

Usage:

- `magresp(g);`
- `magresp(g,L);`
- `magresp(g,L,'nf');`

`MAGRESP(g)` will display the magnitude response of the window `g`. This is the `DFT` of `g` shown on a log scale normalized such that the peak is 0 db.

`MAGRESP(g,L)` does the same, but extends the window to length `L`. Always use this mode for `FIR` windows, and select an `L` somewhat longer than the window to make an accurate plot.

If the input window is real, only the positive frequencies will be shown. `MAGRESP(g,L,'nf')` will show the negative frequencies anyway.

B.1.9 Cosine and Sine transforms.

B.1.9.1 DCTI

Discrete Cosine Transform type I

Usage:

- `c = dcti(f);`
- `c = dcti(f,N);`
- `c = dcti(f,[],dim);`
- `c = dcti(f,N,dim);`

`DCTI(f)` computes the discrete cosine transform of type I of the input signal `f`. If `f` is a matrix, then the transformation is applied to each column. For `N-D` arrays, the transformation is applied to the first dimension.

`DCTI(f,N)` zero-pads or truncates `f` to length `N` before doing the transformation.

`DCTI(f,[],dim)` applies the transformation along dimension `dim`. `DCTI(f,N,dim)` does the same, but pads or truncates to length `N`.

The transform is real (output is real if input is real) and it is orthonormal.

This transform is its own inverse.

Let `f` be a signal of length `L`, let `c=DCTI(f)` and define the vector `w` of length `L` by

$$w\left(n\right)=\begin{cases}\frac{1}{\sqrt{2}} & \text{if } n=0 \text{ or } n=L-1\\ 1 & \text{otherwise}\end{cases}$$

Then

$$c\left(n+1\right)=\sqrt{\frac{2}{L-1}}\sum_{m=0}^{L-1}w\left(n\right)w\left(m\right)f\left(m+1\right)\cos\left(\frac{\pi nm}{L-1}\right)$$

The implementation of this functions uses a simple algorithm that require an `FFT` of length 2N-2, which might potentially be the product of a large prime number. This may cause the function to sometimes execute slowly. If guaranteed high speed is a concern, please consider using one of the other `DCT` transforms.

References: [87], [109]

B.1.9.2 DCTII

Discrete Consine Transform type II

Usage:

- `c = dctii(f);`
- `c = dctii(f,N);`
- `c = dctii(f,[],dim);`
- `c = dctii(f,N,dim);`

`DCTII(f)` computes the discrete consine transform of type `II` of the input signal `f`. If `f` is a matrix, then the transformation is applied to each column. For `N-D` arrays, the transformation is applied to the first dimension.

`DCTII(f,N)` zero-pads or truncates `f` to length `N` before doing the transformation.

`DCTII(f,[],dim)` applies the transformation along dimension `dim`. `DCTII(f,N,dim)` does the same, but pads or truncates to length `N`.

The transform is real (output is real if input is real) and it is orthonormal.

This is the inverse of `DCTIII.`

Let `f` be a signal of length `L,` let `c=DCTII(f)` and define the vector `w` of length `L` by

$$w\left(n\right)=\begin{cases}\frac{1}{\sqrt{2}} & \text{if } n=0\\ 1 & \text{otherwise}\end{cases}$$

Then

$$c\left(n+1\right)=\sqrt{\frac{2}{L}}\sum_{m=0}^{L-1}w\left(n\right)f\left(m+1\right)\cos\left(\frac{\pi}{L}n\left(m+\frac{1}{2}\right)\right)$$

References: [87], [109]

B.1.9.3 DCTIII

Discrete Consine Transform type III

Usage:

- `c = dctiii(f);`
- `c = dctiii(f,N);`
- `c = dctiii(f,[],dim);`
- `c = dctiii(f,N,dim);`

`DCTIII(f)` computes the discrete consine transform of type `III` of the input signal `f`. If `f` is a matrix, then the transformation is applied to each column. For `N-D` arrays, the transformation is applied to the first dimension.

`DCTIII(f,N)` zero-pads or truncates `f` to length `N` before doing the transformation.

`DCTIII(f,[],dim)` applies the transformation along dimension `dim`. `DCTIII(f,N,dim)` does the same, but pads or truncates to length `N`.

The transform is real (output is real if input is real) and it is orthonormal.

This is the inverse of `DCTII`

Let `f` be a signal of length `L`, let `c=DCTIII(f)` and define the vector `w` of length `L` by

$$w(n) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } n = 0 \\ 1 & \text{otherwise} \end{cases}$$

Then

$$c(n+1) = \sqrt{\frac{2}{L}} \sum_{m=0}^{L-1} w(n) f(m+1) \cos\left(\frac{\pi}{L}\left(n+\frac{1}{2}\right)m\right)$$

References: [87], [109]

B.1.9.4 DCTIV

Discrete Consine Transform type IV

Usage:

- `c = dctiv(f);`

`DCTIV(f)` computes the discrete consine transform of type `IV` of the input signal `f`. If `f` is a matrix, then the transformation is applied to each column.

`DCTIV(f,N)` zero-pads or truncates `f` to length N before doing the transformation.

`DCTIV(f,[],dim)` applies the transformation along dimension dim. `DCTIV(f,N,dim)` does the same, but pads or truncates to length N.

B.1.9.5 DSTI

Discrete Sine Transform type I

Usage:

- `c = dsti(f);`
- `c = dsti(f,N);`
- `c = dsti(f,[],dim);`
- `c = dsti(f,N,dim);`

`DSTI(f)` computes the discrete sine transform of type I of the input signal `f`. If `f` is a matrix, then the transformation is applied to each column. For `N-D` arrays, the transformation is applied to the first dimension.

`DSTI(f,N)` zero-pads or truncates `f` to length `N` before doing the transformation.

`DSTI(f,[],dim)` applies the transformation along dimension `dim`. `DSTI(f,N,dim)` does the same, but pads or truncates to length `N`.

The transform is real (output is real if input is real) and it is orthonormal.

This transform is its own inverse.

Let `f` be a signal of length `L` and let `c=DSTI(f)`. Then

$$c(n+1) = \sqrt{\frac{2}{L+1}} \sum_{m=0}^{L-1} f(m+1) \sin\left(\frac{\pi (n+1)(m+1)}{L+1}\right)$$

The implementation of this functions uses a simple algorithm that require an `FFT` of length `2N+2`, which might potentially be the product of a large prime number. This may cause the function to sometimes execute slowly. If guaranteed high speed is a concern, please consider using one of the other `DST` transforms.

B.1.9.6 DSTII

Discrete Sine Transform type II

Usage:

- `c = dstii(f);`
- `c = dstii(f,N);`
- `c = dstii(f,[],dim);`
- `c = dstii(f,N,dim);`

`DSTII(f)` computes the discrete sine transform of type `II` of the input signal `f`. If `f` is a matrix, then the transformation is applied to each column. For `N-D` arrays, the transformation is applied to the first dimension.

`DSTII(f,N)` zero-pads or truncates `f` to length `N` before doing the transformation.

`DSTII(f,[],dim)` applies the transformation along dimension `dim`. `DSTII(f,N,dim)` does the same, but pads or truncates to length `N`.

The transform is real (output is real if input is real) and it is orthonormal.

The inverse transform of `DSTII` is `DSTIII`.

Let `f` be a signal of length `L`, let `c=DSTII(f)` and define the vector `w` of length `L` by

$$w(n) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } n = L-1 \\ 1 & \text{otherwise} \end{cases}$$

Then

$$c(n+1) = \sqrt{\frac{2}{L}} \sum_{m=0}^{L-1} w(n) f(m+1) \sin\left(\frac{\pi}{L} n \left(m + \frac{1}{2}\right)\right)$$

References: [87], [109]

B.1.9.7 DSTIII

Discrete sine transform type III

Usage:

- `c = dstiii(f);`
- `c = dstiii(f,N);`
- `c = dstiii(f,[],dim);`
- `c = dstiii(f,N,dim);`

`DSTIII(f)` computes the discrete sine transform of type `III` of the input signal `f`. If `f` is a matrix, then the transformation is applied to each column. For `N-D` arrays, the transformation is applied to the first dimension.

`DSTIII(f,N)` zero-pads or truncates `f` to length `N` before doing the transformation.

`DSTIII(f,[],dim)` applies the transformation along dimension `dim`. `DSTIII(f,N,dim)` does the same, but pads or truncates to length `N`.

The transform is real (output is real if input is real) and it is orthonormal.

This is the inverse of `DSTII`

Let `f` be a signal of length `L`, let `c=DSTIII(f)` and define the vector `w` of length `L` by

$$w(n) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } n = L-1 \\ 1 & \text{otherwise} \end{cases}$$

Then

$$c(n+1) = \sqrt{\frac{2}{L}} \sum_{m=0}^{L-1} w(n) f(m+1) \sin\left(\frac{\pi}{L} \left(n + \frac{1}{2}\right) m\right)$$

B.1.9.8 DSTIV

Discrete Sine Transform type IV

Usage:

- `c = dstiv(f);`
- `c = dstiv(f,N);`
- `c = dstiv(f,[],dim);`

- `c = dstiv(f,N,dim);`

`DSTIV(f)` computes the discrete sine transform of type `IV` of the input signal `f`. If `f` is a matrix, then the transformation is applied to each column. For `N-D` arrays, the transformation is applied to the first dimension.

`DSTIV(f,N)` zero-pads or truncates `f` to length `N` before doing the transformation.

`DSTIV(f,[],dim)` applies the transformation along dimension `dim`. `DSTIV(f,N,dim)` does the same, but pads or truncates to length `N`.

The transform is real (output is real if input is real) and it is orthonormal. It is its own inverse.

Let `f` be a signal of length `L` and let `c=DSTIV(f)`. Then

$$c\left(n+1\right)=\sqrt{\frac{2}{L}}\sum_{m=0}^{L-1}f\left(m+1\right)\sin\left(\frac{\pi}{L}\left(n+\frac{1}{2}\right)\left(m+\frac{1}{2}\right)\right)$$

B.2 LTFAT - Object oriented interface

B.2.1 Setup and deletion of object

B.2.1.1 TFR_CREATE

Create time-frequency representation object

Usage:

- `tfr = tfr_create(tt,a,M,awin,swin);`

Input parameters:

- `tt`: Name of transform.
- `a`: Time shift.
- `M`: Number of channels.
- `awin`: Analysis window.
- `swin`: Synthesis window.
- `options`: Cell array of `options` (optional)

Output: `tfr` : Handle to this `TF` representation.

This function creates a time-frequency representation represented by the variable `tfr`.

`tt` can be one of: 'dgt', 'rdgt', 'rdgt2', 'dwilt', 'edgt6'

`awin` and `swin` can be one of 'none','pgauss','candual','cantight' 'pherm'

`options` can be one of 'phaselock'

B.2.1.2 TFR_SET_WIN

Set analysis/synthesis window

Usage:

- `tfr_set_win(self,window)`

Window can be a column vector or a string denoting the type of `window` chosen. In the latter case, a appropriate `window` of that type will be choosen.

B.2.1.3 TFR_CLEAR

Clear time-frequency represenation object

Usage:

- `tfr_clear(handle);`

This clears the `TFR` object represented by `handle`. This will free some memory.

B.2.1.4 TFR_CLEARALL

Clear all time-frequency represenation objects

Usage:

- `tfr_clearall();`

This clears the `TFR` object represented by handle. This will free some memory.

B.2.2 Analysis and synthesis

B.2.2.1 TFR_A

Analysis transform

Usage:

- `c = tfr_a(tfr,f)`
- `c = tfr_a(tfr,f,L)`
- `[c,Ls] = tfr_a(tfr,f);`
- `[c,Ls] = tfr_a(tfr,f,L);`
- `[c,Ls] = tfr_a(tfr,f,[],dim);`
- `[c,Ls] = tfr_a(tfr,f,L,dim);`

Input parameters:

- `tfr`: Time/frequency representation handle.
- `f`: Input data
- `L`: Length of transform to do.
- `dim`: Dimension of which to do the transform.

Output parameters:

- `c`: Array of coefficients.
- `Ls`: Length of input signal.

`TFR_A(tfr,f)` computes the analysis transform of the signal `f` as specified by `tfr`.

The length of the transform will be the smallest possible that is larger than the signal. `f` will be zero-extended to the length of the transform. If `f` is a matrix, the transformation is applied to each column.

The length of the transform done can be obtained by `L=size(c,2)*a;`

`TFR_A(tfr,f,L)` computes the analysis transform as above, but does a transform of length `L`. `f` will be cut or zero-extended to length `L` before the transform is done.

`[c,Ls]=TFR_A(tfr,f)` or `[c,Ls]=TFR_A(tfr,f,L)` additionally returns the length of the input signal `f`. This is handy for reconstruction:

```
[c,Ls]=tfr_a(tfr,f)
fr=tfr_s(c,Ls);
```

will reconstruct the signal `f` no matter what the length of `f` is, provided that the analysis and synthesis window of `tfr` are dual windows.

`TFR_A(tfr,f,[],dim)` applies the transformation along dimension `dim`. `TFR_A(tfr,f,L,dim)` does the same, but pads or truncates to length `L`.

B.2.2.2 TFR_AR

Analysis transform, rectangular layout

Usage:

- `c = tfr_ar(tfr,f)`
- `c = tfr_ar(tfr,f,L)`
- `[c,Ls] = tfr_ar(tfr,f);`
- `[c,Ls] = tfr_ar(tfr,f,L);`

Input parameters:

- tfr: Time/frequency representation handle.
- f: Input data
- L: Length of transform to do.

Output parameters:

- c: Array of coefficients.
- Ls: Length of input signal.

TFR_AR(tfr,f) computes the analysis transform of the signal f as speficied by tfr. The coefficients are returned in a rectangular layout.

The length of the transform will be the smallest possible that is larger than the signal. f will be zero-extended to the length of the transform. If f is a matrix, the transformation is applied to each column.

The length of the transform done can be obtained by L=size(c,2)*a;

TFR_AR(tfr,f,L) computes the analysis transform as above, but does a transform of length L. f will be cut or zero-extended to length L before the transform is done.

[c,Ls]=TFR_AR(tfr,f) or [c,Ls]=TFR_AR(tfr,f,L) additionally returns the length of the input signal f. This is handy for reconstruction:

```
[c,Ls]=tfr_ar(tfr,f)
fr=tfr_sr(c,Ls);
```

will reconstuct the signal f no matter what the length of f is, provided that the analysis and synthesis window of tfr are dual windows.

Not all transforms support this method. The supported transforms are: 'dgt', 'dgtii', 'dgtiii', 'dgtiv' 'rdgt', 'rdgtiii' 'edgtii'

B.2.2.3 TFR_S

Synthesis transform

Usage:

- f = tfr_s(tfr,c)
- f = tfr_s(tfr,c,Ls)
- f = tfr_s(tfr,c,[],dim)
- f = tfr_s(tfr,c,Ls,dim)

Input parameters:

- tfr: Time/frequency representation handle.

- `c`: Array of coefficients.
- `Ls`: length of signal.
- `dim`: Dimension of which to do the transform.

Output parameters:

- `f`: Signal.

`TFR_S(tfr,c)` computes the synthesis transform of the coefficients `c` as speficied by `tfr`.

`TFR_S(tfr,c,Ls)` does as above but cuts or extends `f` to length `Ls`.

`TFR_S(tfr,c,[],dim)` applies the transform along dimension `dim`. `TFR_S(tfr,c,Ls,dim)` does the same, but cuts or extends `f` to length `Ls`.

B.2.2.4 TFR_SR

Synthesis transform, rectangular layout

Usage:

- `f = tfr_sr(tfr,c)`
- `f = tfr_sr(tfr,c,Ls)`

Input parameters:

- `tfr`: Time/frequency representation handle.
- `c`: Array of coefficients.
- `Ls`: length of signal.

Output parameters:

- `f`: Signal.

`TFR_SR(tfr,c)` computes the synthesis transform of the coefficients `c` as speficied by `tfr`. The coefficients must be arranged in a rectangular layout as returned by `TFR_AR`.

`TFR_SR(tfr,c,Ls)` does as above but cuts or extends `f` to length `Ls`.

B.2.3 Information

B.2.3.1 TFR_GET_WIN

Get analysis/synthesis window.

Usage:

- `g = tfr_get_win(tfr,win,L);`

`TFR_GET_WIN(tfr,'analysis',L)` will return the analysis window corresponding to a transform of length `L`. Instead of `'analysis'` you may use `'a'` or `1`.

To get the synthesis window, use `'synthesis'`, `'s'` or `2`.

B.2.4 Condition numbers etc.

B.2.4.1 TFR_BOUNDS

Calculate frame bounds

Usage:

- `fcond = tfr_bounds(tfr);`
- `[A,B] = tfr_bounds(tfr,win);`
- `[A,B] = tfr_bounds(tfr,win,L);`

Input parameters:

- `tfr`: Time/frequency representation handle.
- `L`: Length of transform to consider.

Output parameters:

- `fcond`: Frame condition number `(B/A)`
- `A,B`: Frame bounds.

`TFR_BOUNDS(tfr)` calculates the ratio `B/A` of the frame bounds of the frame specified by `tfr`. The analysis window is used.

`TFR_BOUNDS(tfr,win)` allows you to specify whether the analysis or synthesis window should be used. Setting `win` to 'analysis', 'a' or 1 will result in the analysis window being used, and setting `win` to 'synthesis','s' or 2 will result in the synthesis window being used.

`TFR_BOUNDS(tfr,win,L)` or `[A,B]=TFR_BOUNDS(tfr,win,L)` does the same assuming a transform length of `L`. If `L` is not specified, the smallest possible transform length will be used.

B.2.5 Multipliers

B.2.5.1 TFR_MUL

Multiplier

Usage:

- `h = tfr_mul(tfr,f,c)`
- `h = tfr_mul(tfr,f,c,L)`

`TFR_MUL(tfr,f,c)` will filter `f` by a multiplier using the symbol `c`. The symbol `c` can be either a vector or a matrix.

`TFR_MUL(tfr,f,c,L)` do the same using a transform of length `L`.

B.3 LTFAT - Signal processing tools

B.3.1 Working with coefficients.

B.3.1.1 LARGESTR

Keep largest ratio of coefficients.

Usage:

- `xo = LARGESTR(x,p);`
- `[xo,N] = LARGESTR(x,p);`

`LARGESTR(x,p)` returns an array of the same size as `x` keeping the fraction `p` of the coefficients. The largest coefficients are kept.

`[xo,N]=LARGESTR(xi,p)` will additionally return the number of coefficients kept.

Note that if this function is used on coefficients coming from a redundant transform or from a transform where the input signal was padded, the coefficient array will be larger than the original input signal. Therefore, the number of coefficients kept might be higher than expected.

References: [70]

B.3.1.2 LARGESTN

Keep N largest coefficients.

Usage:

- `xo = largestn(x,N);`

`LARGESTN(x,N)` returns an array of the same size as `x` keeping the `N` largest coefficients.

If the coefficients represents a signal expanded in an orthonormal basis, then this will be the best N-term approximation.

References: [70]

B.3.1.3 THRESH

Threshold (hard/soft)

Usage:

- `x = thresh(ttype,x,lambda);`
- `[x,N] = thresh(ttype,x,lambda);`

`THRESH('hard',x,lambda)` or `THRESH(1,x,lambda)` will perform hard thresholding on `x`, i.e. all element with absolute value less than `lambda` will be set to zero.

`THRESH('soft',x,lambda)` or `THRESH(2,x,lambda)` will perform soft thresholding on `x`, i.e. `lambda` will be subtracted from the absolute value of every element of `x`.

`[x,N]=THRESH(ttype,x,lambda)` additionally returns a number `N` specifying how many numbers where kept.

The function `WTHRESH` in the Matlab Wavelet toolbox implements the same functionality.

B.3.1.4 UQUANT

Simulate uniform quantization.

Usage:

- `x = uquant(x,nbits);`
- `x = uquant(x,nbits,xmax);`
- `x = uquant(qtype,x,nbits,xmax);`
- `x = uquant(qtype,x,nbits,xmax);`

`UQUANT(x,nbits)` simulates the effect of uniform quantization of `x` using `nbits` bit. The output is simply `x` rounded to 2^{nbits} different values.

`UQUANT(x,nbits,xmax)` does as above, but allows you to specify the maximal value that should be quantifiable. If not specified, the maximal value of the signal will be used.

`UQUANT(qtype,x,nbits)` or `UQUANT(qtype,x,nbits,xmax)` uses quantization depending on `qtype`. `qtype` may be one of:

's','signed'	Default quantization type. This assumes that the signal has a both positive and negative part. Usefull for sound signals.
'u','unsigned'	Assumes the signal is positive. Negative values are silently rounded to zero. Usefull for images.

If this function is applied to a complex signal, it will simply be applied to the real and imaginary part separately.

B.3.1.5 PHASELOCK

Phaselock Gabor coefficients

Usage:

- `c = phaselock(c,a);`

`PHASELOCK(c,a)` phaselocks the Gabor coefficients `c`. The coefficient must have been obtained from a `DGT` with parameter `a`.

Phaselocking the coefficients modifies them so as if they were obtained from a time-invariant Gabor system. A filter bank produces phase locked coefficients.

References: [84]

B.3.1.6 IPHASELOCK

Inverse phase lock of Gabor coefficients

Usage:

- `c = iphaselock(c,a);`

`IPHASELOCK(c,a)` removes phase locking from the Gabor coefficients `c`. The coefficient must have been obtained from a `DGT` with parameter `a`.

Phaselocking the coefficients modyfies them so as if they were obtained from a time-invarient Gabor system. A filter bank produces phase locked coeffiecients.

References: [84]

B.3.2 Extra signal processing tools.

B.3.2.1 MULAWENCODE

Mu-Law compand signal

Usage:

- `[outsig, sigweight] = mulawencode(insig,mu);`

`[outsig, sigweight]=MULAWENCODE(insig,mu)` mu-law compands the input signal `insig` using mu-law companding with parameters `mu`.

References: [61]

B.3.2.2 MULAWDECODE

Inverse of Mu-Law companding

Usage:

- `sig = mulawdecode(codedsig,mu,sigweight);`

`MULAWDECODE(codedsig,mu,sigweight)` inverts a previously applied mu-law companding to the signal `codedsig`. The parameters `mu` and `sigweight` must match those from the call to `MULAWENCODE`

References: [61]

B.3.2.3 QAM

QAM modulation of input signal.

Usage:

- `:  xo = qam(xi,order);`

`xi` must be a vector of 0's and 1's. order must be a power of 2.

B.3.2.4 IQAM

Inverse QAM modulation of signal.

order must be a power of 2. xo is a vector of 0's and 1's.

B.3.2.5 CIRCBITFLIP

Compute sequence of numbers by flipping one bit.

Usage:

- `bitorder = circbitflip(order)`

`CIRCBITFLIP(order)` computes a sequence of numbers from 0 until `order-1` such that each number differs from the previous one by a single flipped bits.

B.4 LTFAT - Examples

B.4.1 Simple examples

B.4.1.1 EXAMP_SPEECH

Spectrogram of speech signal.

This example show how to work with almost real world data.

Figure B.1: This show a spectrogram of Linus Thorvalds saying "My name is Linus Thorvalds, and I pronounce Linux as Linux"

The 90cleaner plot.

B.4.2 Applications

B.4.2.1 EXAMP_COMPRESSION

Image compression example

This examples shows how to do image compression using a two-dimensional Wilson transform.

The image is transformed using an orthonormal `DWILT2` transform followed by N-term approximation and quantization. Then the image is reconstructed and compared with the original.

This example demonstrates both lossless and lossy compression. An image compression is lossless if the truncation error is below the quantization error, as the truncation error is not visible in that case.

Figure B.2: This figure to the left shows lossless compression of the cameraman. The figure to the right shows a lossy compression.

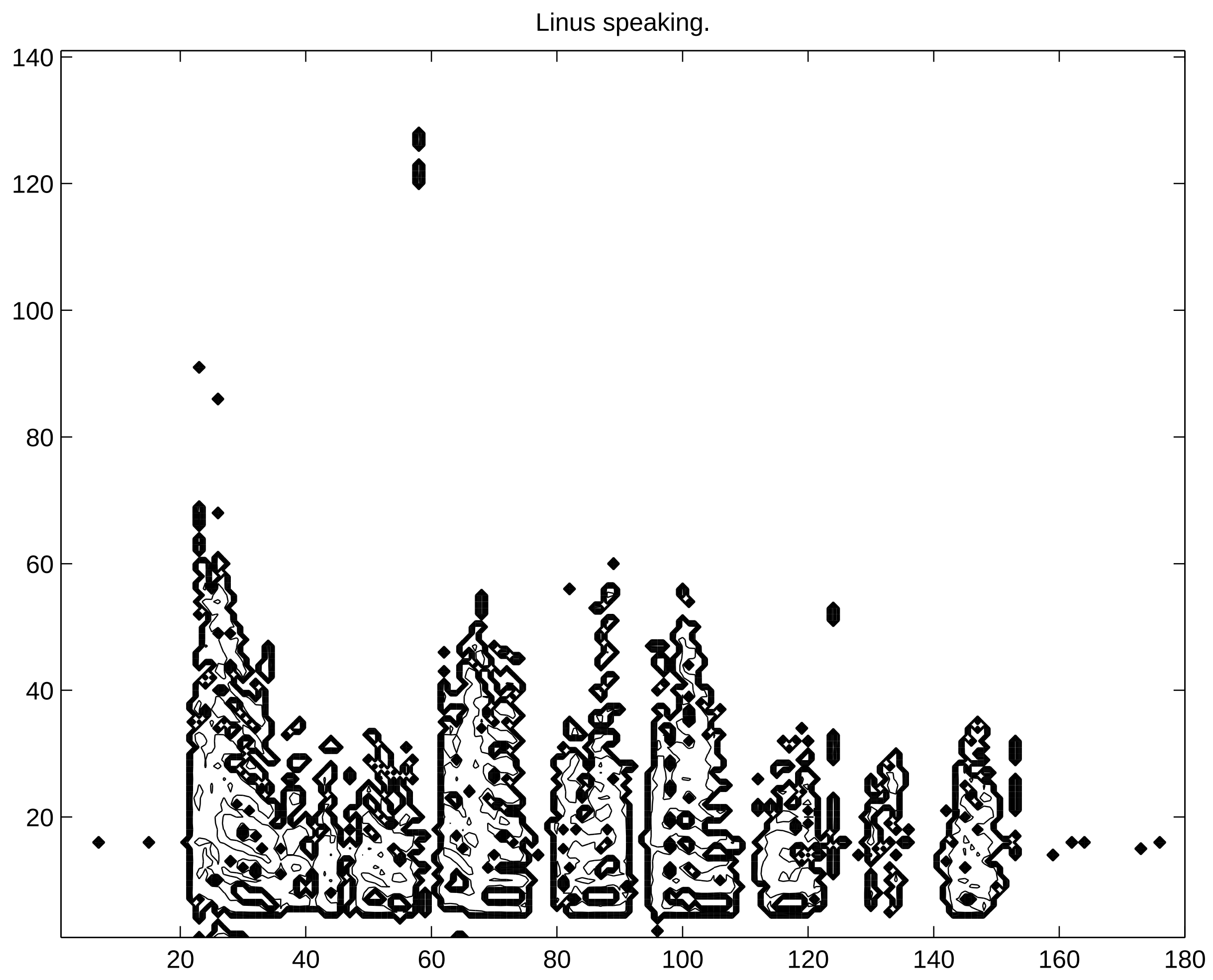


Figure B.1: EXAMP_SPEECH: Linus

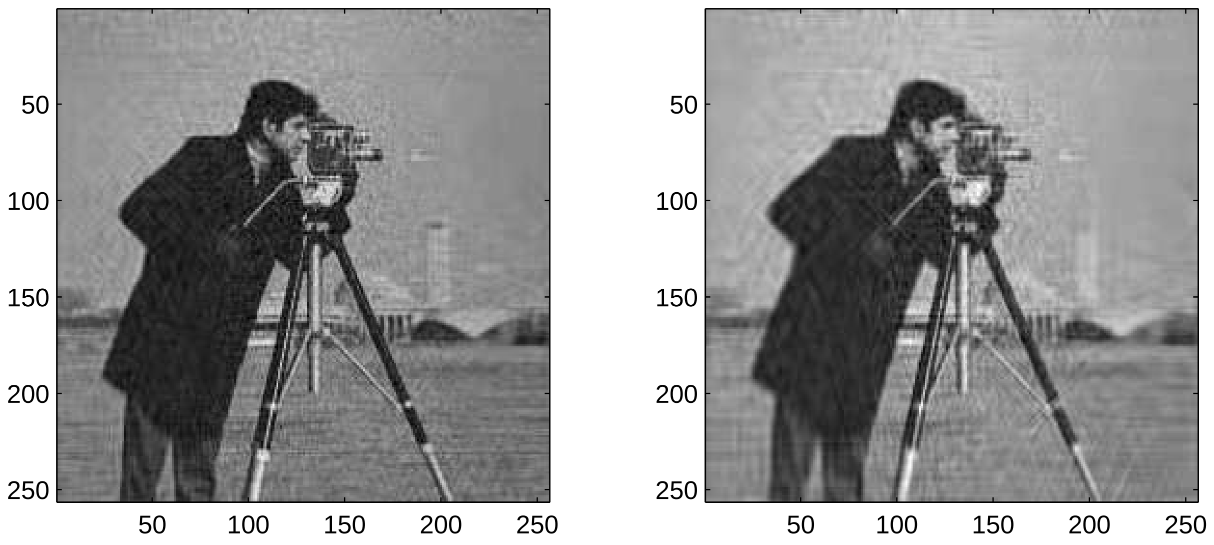


Figure B.2: EXAMP_COMPRESSION: The cameraman.

B.4.2.2 EXAMP_FORMANTS

Find the formants in speech

This script shows a simple method for finding formants in a speech signal. Formants are peaks in a spectrum or spectogram, which can be linked to resonant frequencies of the related physical system. They are most commonly used in phonetics, searching for the formants of the human vocal tract. This is for example used for speaker recognition.

References: [20], [29], [95]

B.4.2.3 EXAMP_OFDM

Example of Gabor systems used for OFDM

This example shows how to use a Gabor Riesz basis for `OFDM`. The example is simple, and assumes that all the whole spectrum is available for transmission.

Some further simplifications used to make this example simple:

- The window and its dual have full length support. This is not practical, because all data would have to be processed at once. Instead, a filter bank approach should be used, with both the window and its dual having a short length.

- The window is periodic. The data at the very end interferes with the data at the very beginning. A simple way to solve this is to transmit zeros at the beginning and at the end, to flush the system properly.

- The channel should be modelled by a pseudo-differential operator that causes time and frequency shifts / dispersion in the signal. This example just uses white noise.

Figure B.3: This figure shows the distribution in the complex plane of the received coefficients.

B.4.2.4 EXAMP_PHASEPLOT

Give examples of nice phaseplots

This script creates a synthetic signal and then uses 'phaseplot' on it, using several of the possible options

For real-life signal only small parts should be analyzed. In the chosen example the fundamental frequency of the speaker can be nicely seen.

Figure B.4: Compare this to the pictures in reference 2 and 3. In the first two figures a synthetic signal is analyzed. It consists of a sinusoid, a small Delta peak, a periodic triangular function and a Gaussian. In the phaselocked version in the first part the periodicity of the sinusoid can be nicely seen also in the phase coefficients. Also the points of discontinuities can be seen as asymptotic lines approached by parabolic shapes. In the third part both properties, periodicity and discontinuities can be nicely seen. A comparison to the spectogram shows that the rectangular part in the middle of the signal can be seen by the phase plot, but not by the spectogram.

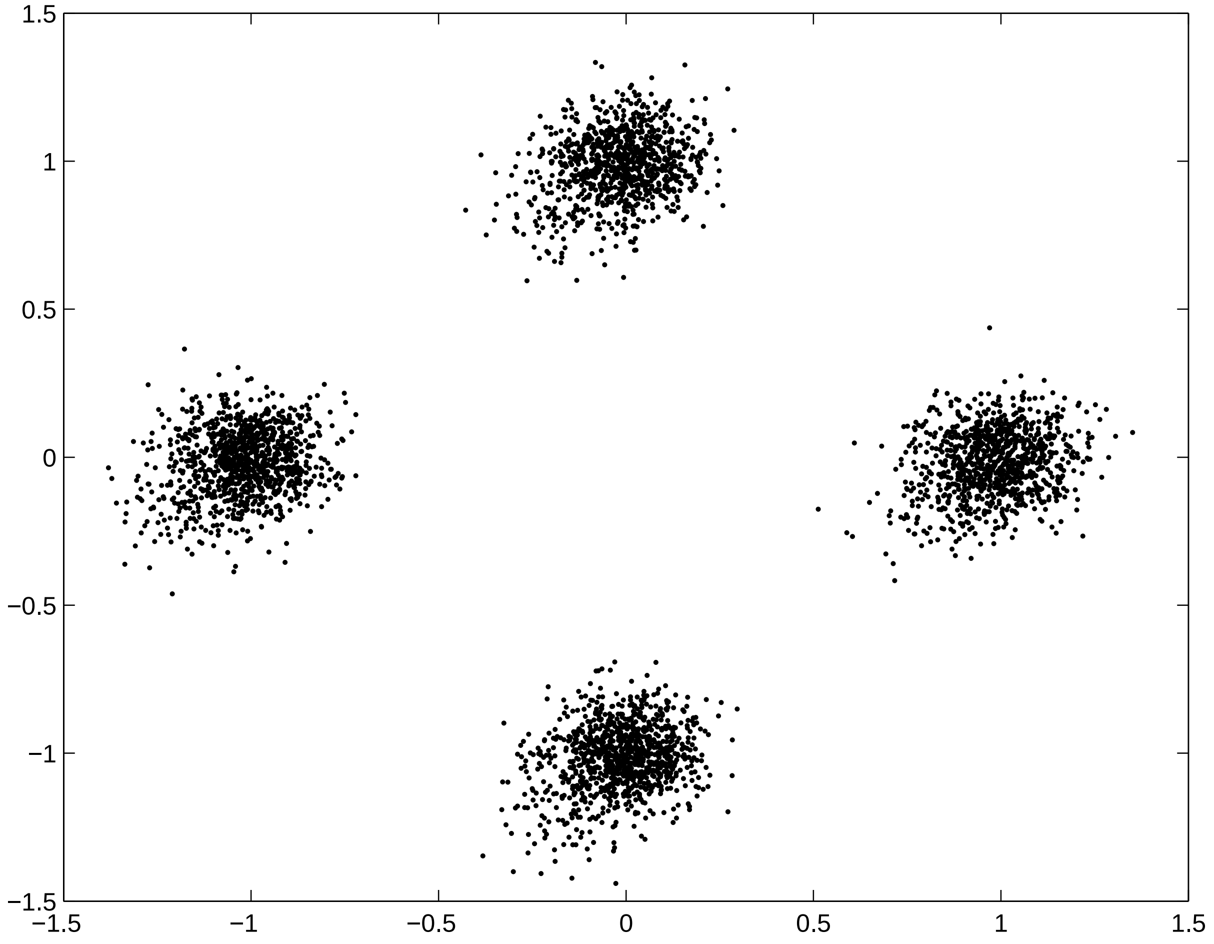


Figure B.3: EXAMP_OFDM: Received coefficients.

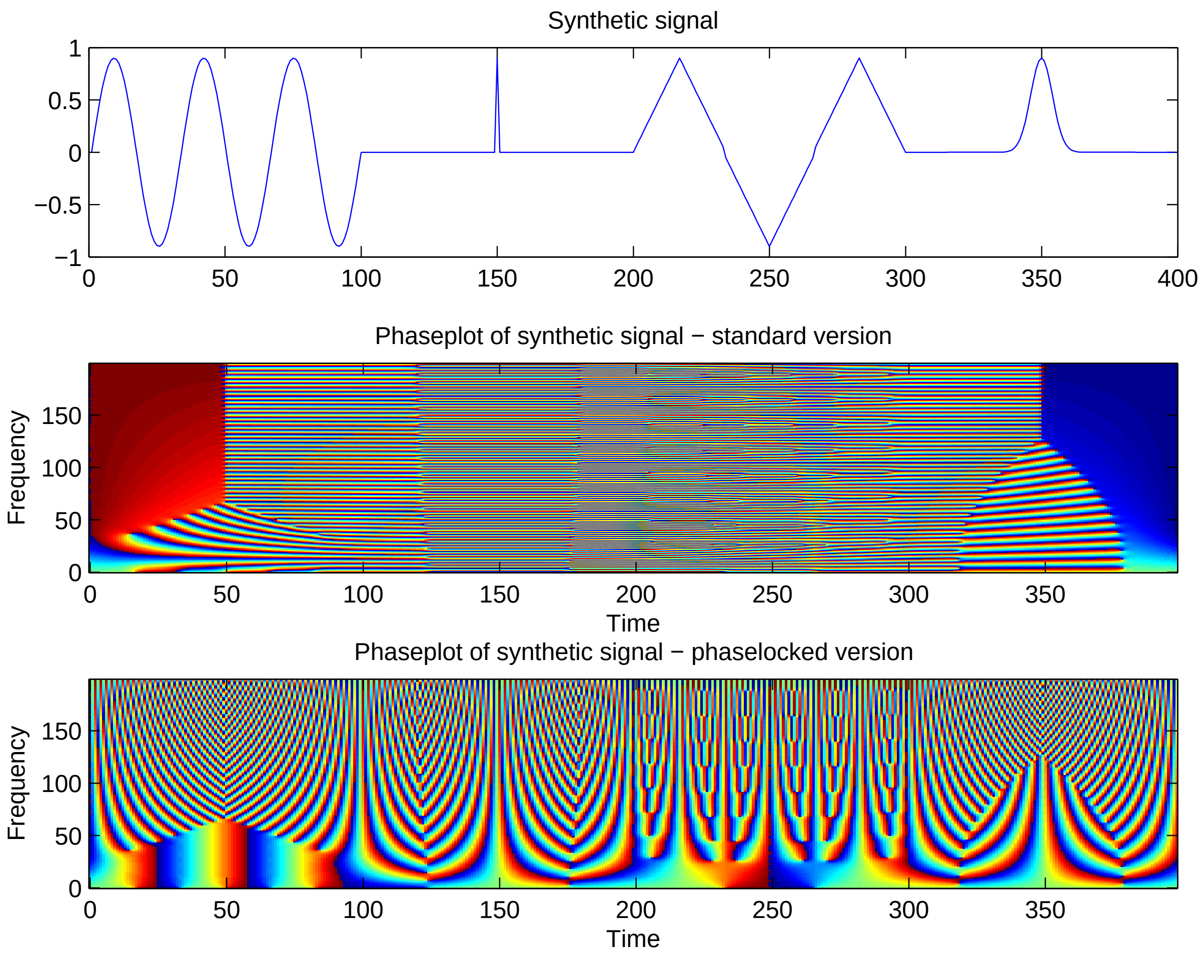


Figure B.4: EXAMP_PHASEPLOT: Synthetic signal

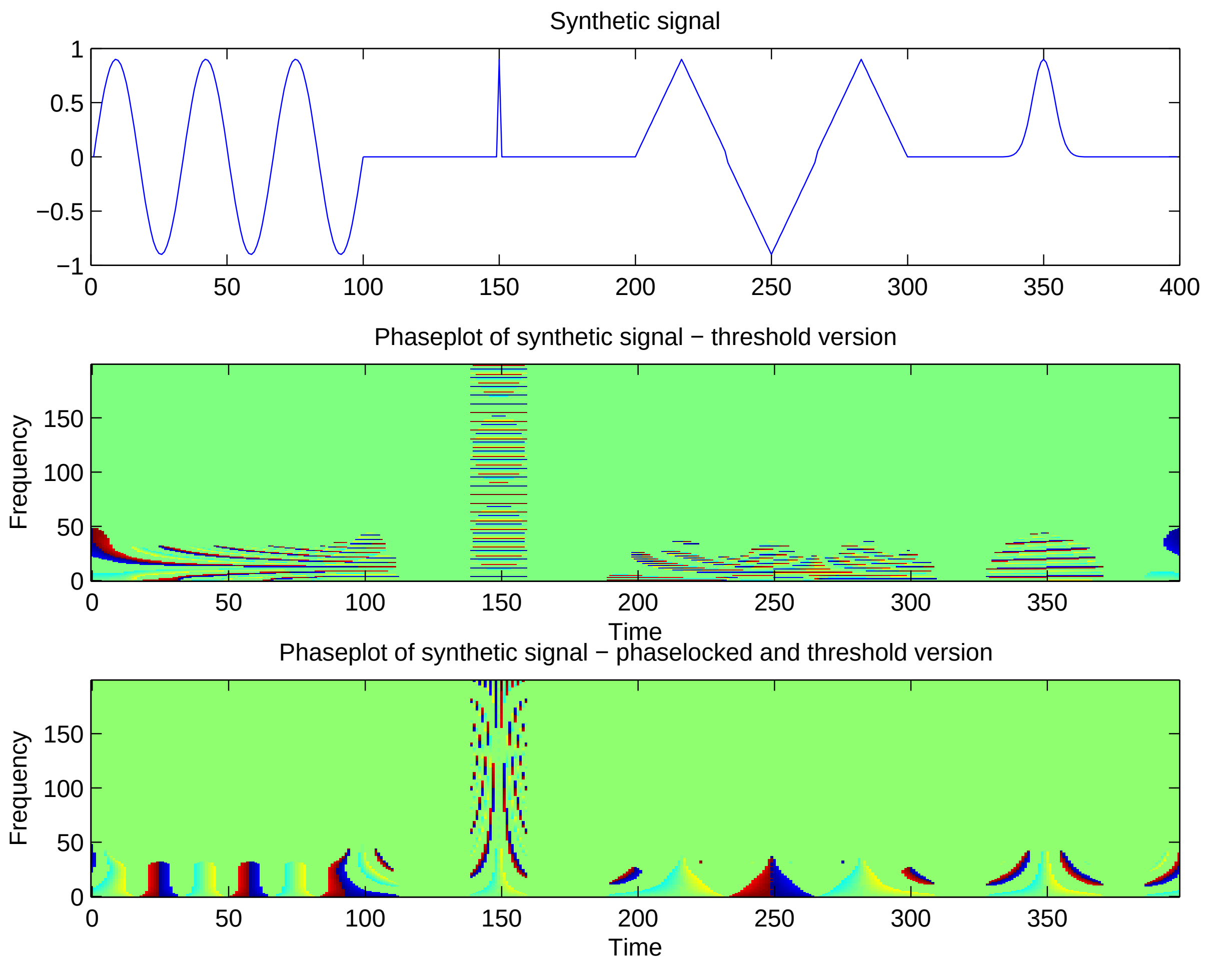


Figure B.5: EXAMP_PHASEPLOT: Synthetic signal, thresholded.

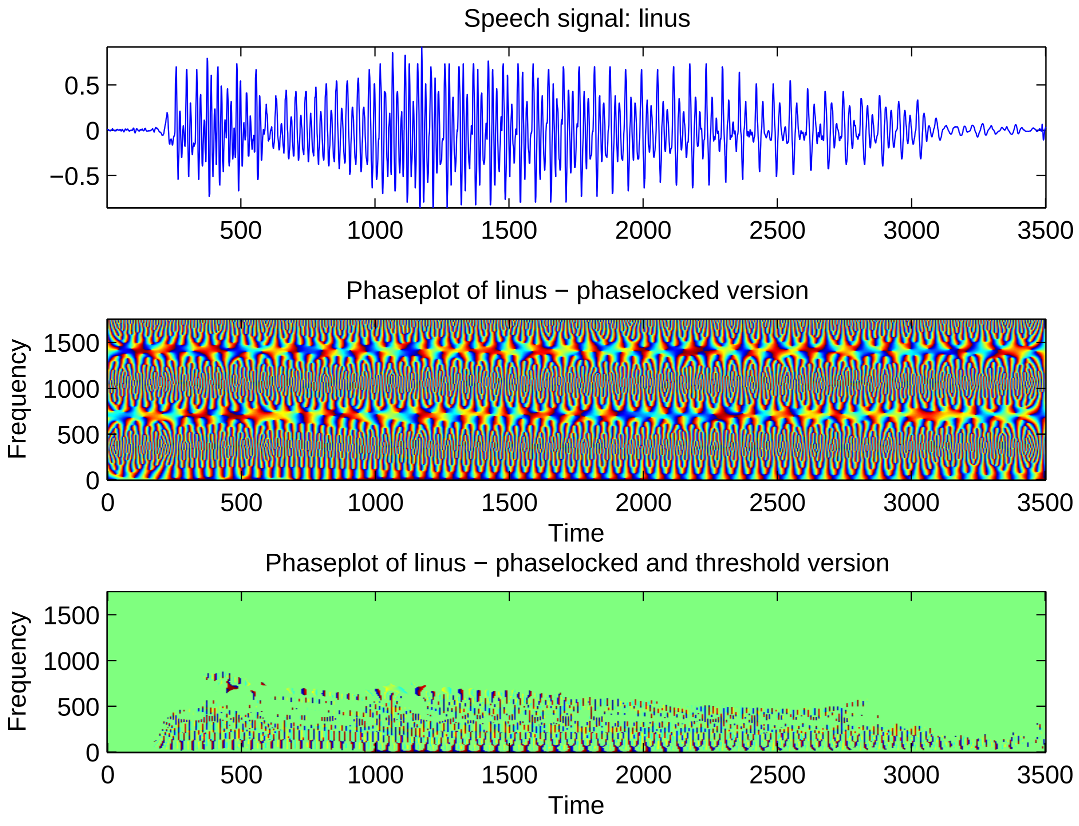


Figure B.6: EXAMP_PHASEPLOT: Speech signal.

In the not phase-locked version still the fundamental frequency of the sinusoid can be guessed as the position of an horizontal asymptotic line.

Figure B.5: This figure shows the same as Figure 1, except that values with low magnitude has been removed.

Figure B.6: The figure shows a part of the 'linus' signal. The fundamental frequency of the speaker can be nicely seen.

B.4.3 Aspect of particular functions

B.4.3.1 EXAMP_PGAUSS

How to use PGAUSS

This script illustrates various properties of the Gaussian function.

Figure B.7: This figure shows an optimally centered Gaussian for a given Gabor system, its canonical dual and tight windows and the DFTs of these windows.

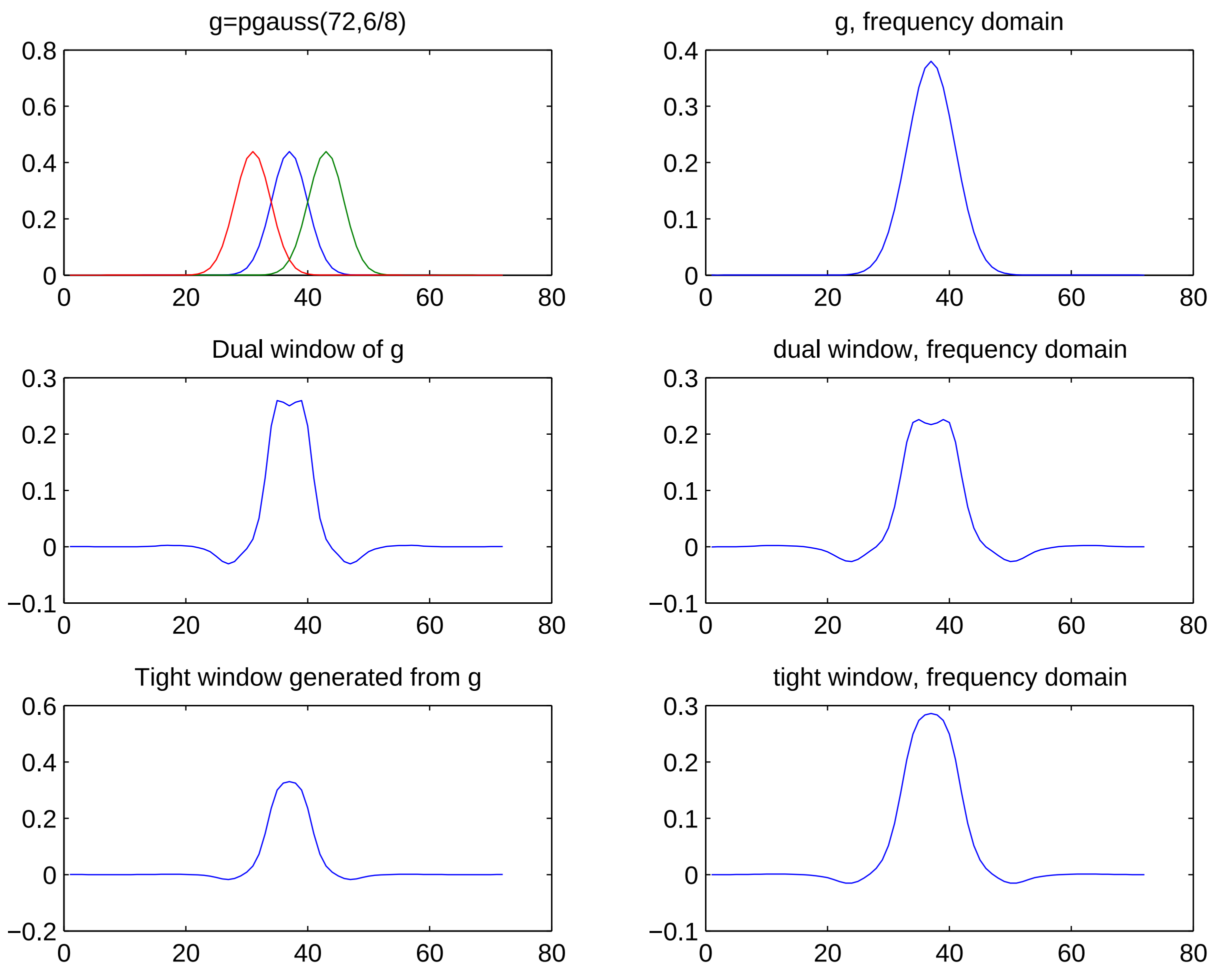


Figure B.7: EXAMP_PGAUSS: Window+Dual+Tight

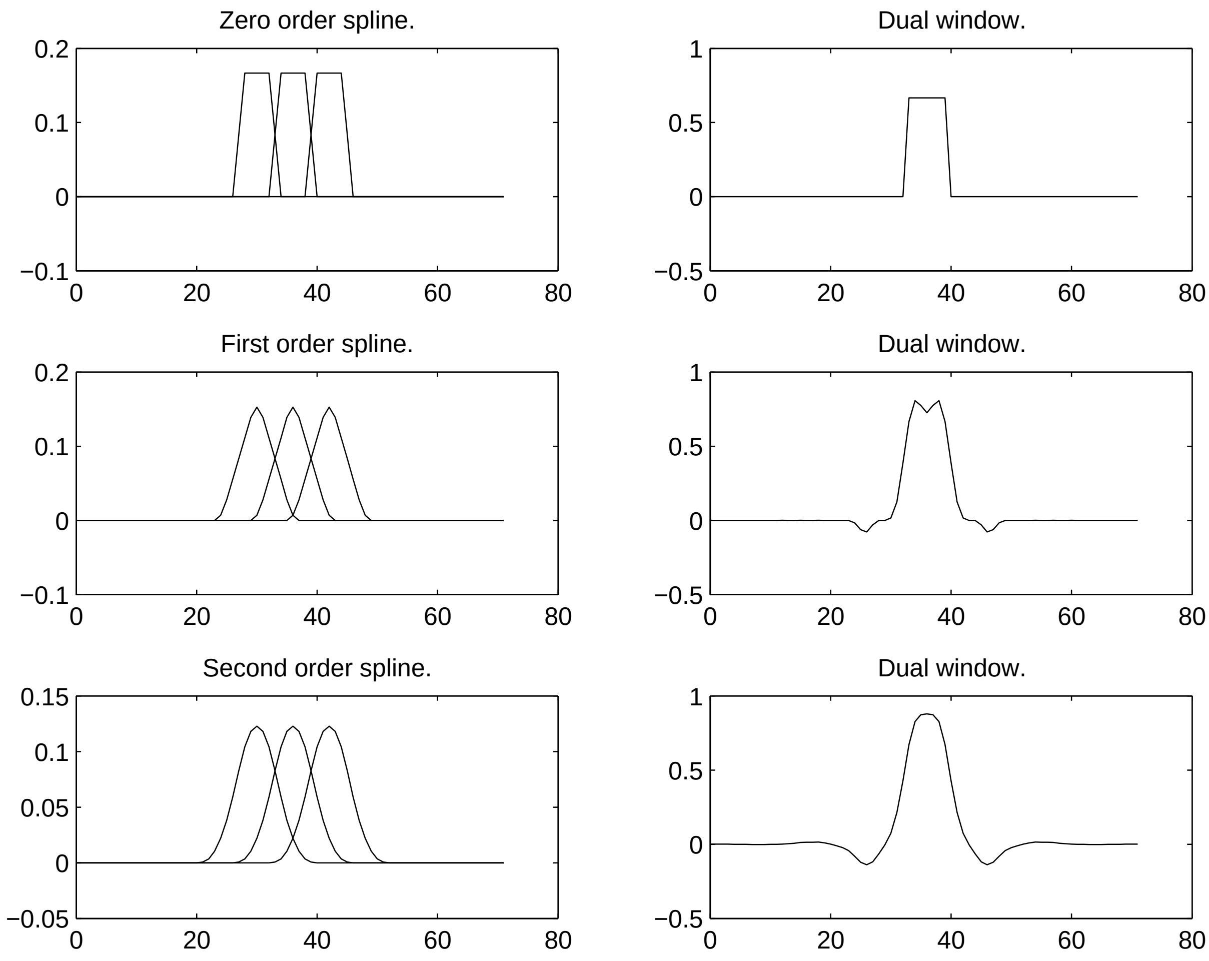


Figure B.8: EXAMP_PBSPLINE: Three first splines

B.4.3.2 EXAMP_PBSPLINE

How to use PBSPLINE

This script illustrates various properties of the Gaussian function.

Figure B.8: This figure shows the three first splines (order 0,1 and 2) and their dual windows.

Note that they are calculated for an even number of the parameter 'a', meaning that they are not exactly splines, but a slightly smoother construction, that still form a partition of unity.

B.4.3.3 EXAMP_MIXDUAL

How to use MIXDUAL

This script illustrates how one can produce dual windows using `MIXDUAL`

The example constructs a dual window that is more concentrated in the time domain by mixing the original Gabor window by one that is extremely well concentrated. The result is somewhat in the middle of these two.

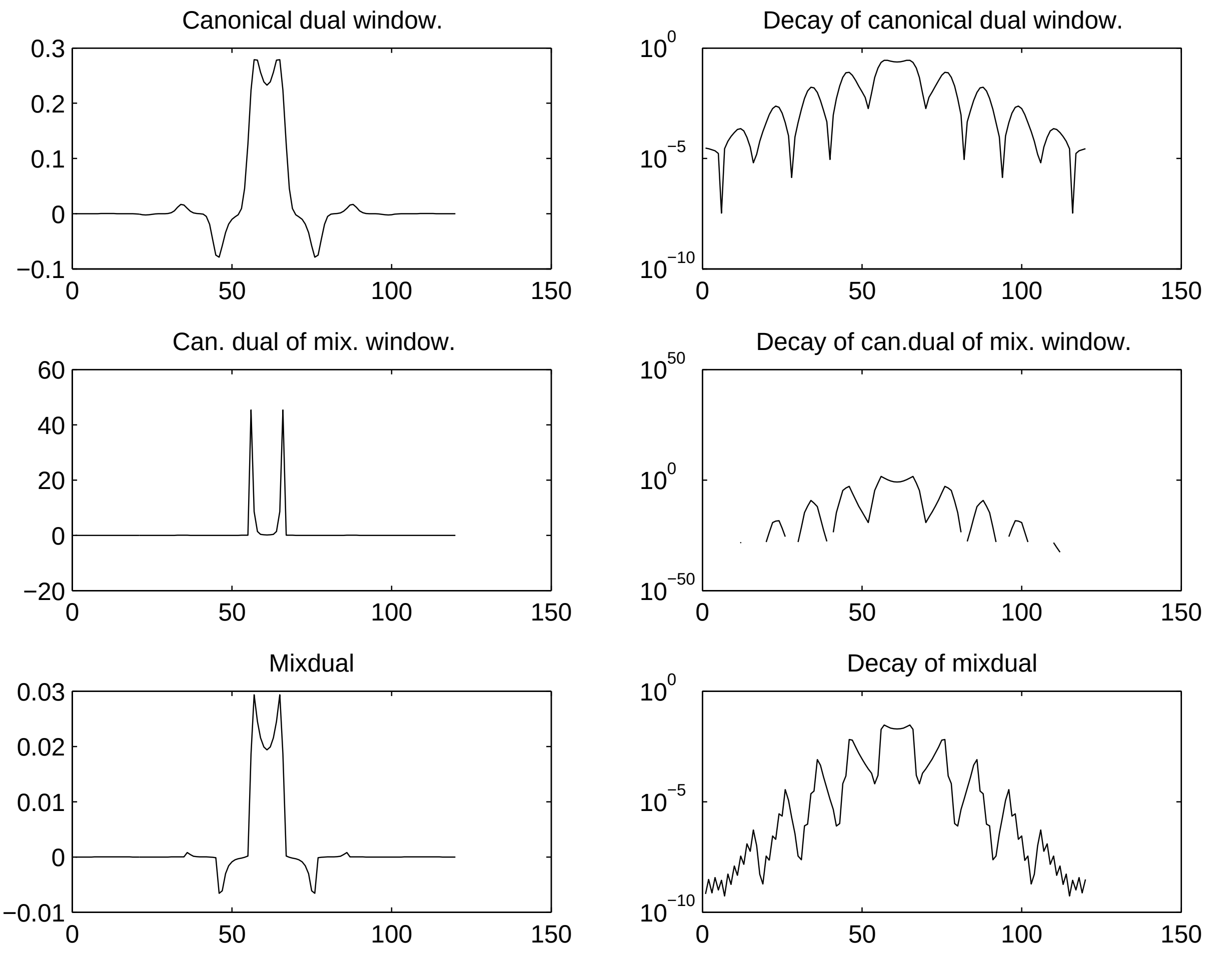


Figure B.9: EXAMP_MIXDUAL: Mixdual of two Gaussians.

The lower framebound of the mixing Gabor system is horrible, but this does not carry over to the mixdual.

Figure B.9: The first row of the figure shows the canonical dual window of the input window, which is a Gaussian function perfectly localized in the time and frequency domains.

The second row shows the canonical dual window of the window we will be mixing with: This is a Gaussian that is 10 times more concentrated in the time domain than in the frequency domain. The resulting canonical dual window has rapid decay in the time domain.

The last row shows the mixdual of these two. This is a non-canonical dual window of the first Gaussian, with decay resembling that of the second.

B.4.3.4 EXAMP_GABMUL

Time-frequency localization by a Gabor multiplier

This script creates several different time-frequency symbols and demonstrate their effect on a random, real input signal.

This figure shows the random signal.

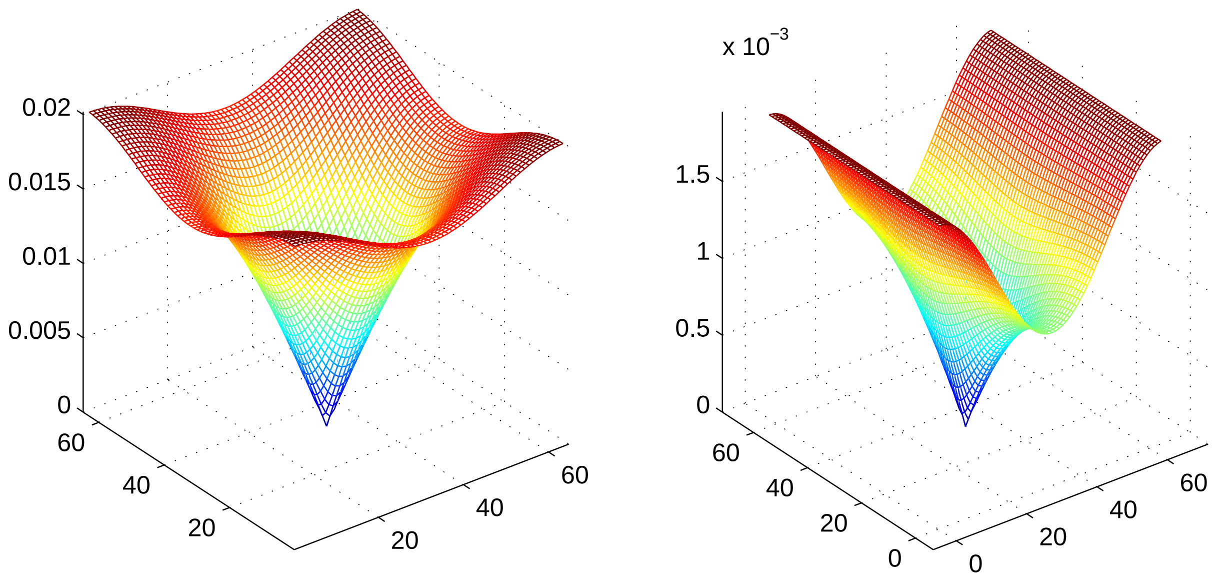


Figure B.10: EXAMP_ZAK: Gaussian and spline

B.4.3.5 EXAMP_ZAK

Some Zak-transforms

This scripts creates plots of three different Zak-transforms of a Gaussian, a spline and a Hermite function

Figure B.10: This figure shows the absolute value of the Zak-transform of a Gaussian to the left, and a second order spline to the right. Notice that both Zak-transforms are 0 in only a single point right in the middle of the plot. The spline is constructed by sampling and periodization of the continuous second order spline.

Figure B.11: The left figure shows the absolute value of the Zak-transform of a 4th order Hermite function and the left shows a chirp. Notice how the Zak transform of the Hermite functions is zero on a circle centered on the corner.

B.4.4 Misc:

B.4.4.1 CTESTFUN

Complex 1-D test function.

Usage:

- `ftest = ctestfun(L);`

`CTESTFUN(L)` returns a test signal consisting of a superposition of a chirp and an indicator function.

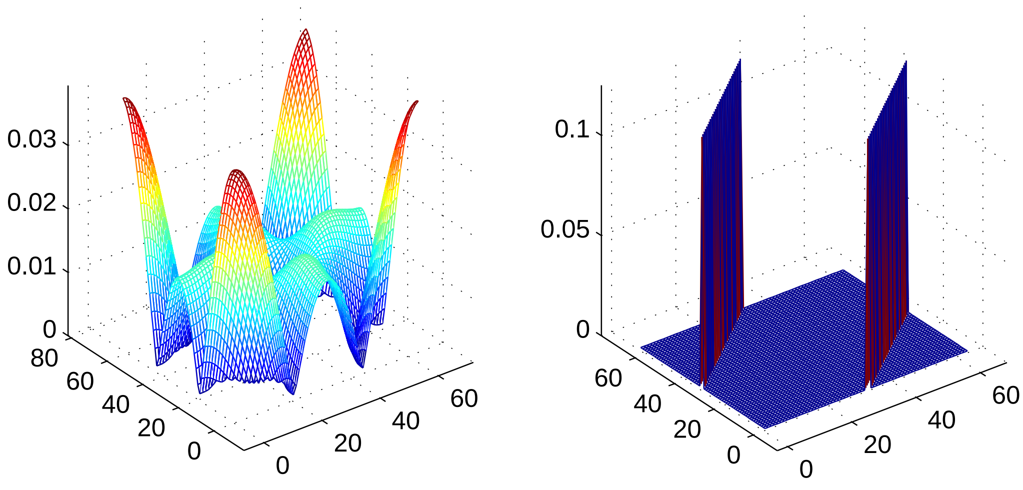


Figure B.11: EXAMP_ZAK: Funny Zaks

B.5 LTFAT - Signals

B.5.1 Sound signals.

B.5.1.1 BAT

Load the 'bat' test signal.

Usage:

- `s = bat();`

`BAT()` loads the 'bat' signal. It is a 400 samples long recording of a `bat` chirp.

The signal can be obtained from http://dsp.rice.edu/software/bat.shtml http://dsp.rice.edu/software/o

Please acknowledge use of this data in publications as follows: "The author wishes to thank Curtis Condon, Ken White, and Al Feng of the Beckman Institute of the University of Illinois for the `bat` data and for permission to use it in this paper."

B.5.1.2 BATMASK

Load a Gabor multiplier symbol for the 'bat' test signal.

Usage:

- `c = batmask();`

`BATMASK()` loads a Gabor multiplier with a 0/1 symbol that mask out the main contents of the 'bat' signal. The symbol fits a Gabor multiplier with lattice given by `a=10` and `M=40`.

The mask was created manually using a image processing program. The mask is symmetric, such that the result will be real valued if the multiplier is applied to a real valued signal using a real valued window.

B.5.1.3 DOPPLER

Load the 'doppler' test signal.

Usage:

- `s = doppler();`

`DOPPLER()` loads the 'doppler' signal. It is a recording of a train passing close by. The signal is 157058 samples long

The signal was obtained from http://www.fourmilab.ch/cship/doppler.html

B.5.1.4 GREASY

Load the 'greasy' test signal.

Usage:

- `s = greasy();`

`GREASY()` loads the 'greasy' signal. It is a recording of a woman pronouncing the word "greasy".

The signal is 8192 samples long and recorded at 16 khz.

The signal was obtained from Wavelab: http://www-stat.stanford.edu/ wavelab/

References: [71]

B.5.1.5 LINUS

Load the 'linus' test signal.

Usage:

- `s = linus();`

`LINUS()` loads the 'linus' signal. It is a recording of Linus Thorvalds pronouncing the words "Hello. My name is Linus Thorvalds, and I pronounce Linux as Linux".

The signal is 41984 samples long.

B.5.2 Images.

B.5.2.1 CAMERAMAN

Load the 'cameraman' test image

Usage:

- `s = cameraman();`

`CAMERAMAN()` loads a 256x256 greyscale image of a `cameraman`.

The returned matrix `s` consists of unsigned integers between 0 and 255. In some situations, converting to double precision by

```
s=double(cameraman);
```

is necessary.

Bibliography

[1] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide, Third Edition.* SIAM, Philadelphia, 1999.

[2] L. Auslander and Y. Meyer. A generalized Poisson summation formula. *Appl. Comput. Harmon. Anal.*, 3(4):372–376, 1996.

[3] P. Balazs. *Regular and Irregular Gabor Multipliers with Application to Psychoacoustic Masking.* PhD thesis, Fakultät für Mathematik der Universität Wien, Vienna, june 2005.

[4] P. Balazs, H. G. Feichtinger, M. Hampejs, and G. Kracher. Double preconditioning for Gabor frames. *IEEE Trans. Signal Process.*, 54(12):4597–4610, December 2006.

[5] P. Balazs, H. G. Feichtinger, M. Hampejs, and G. Kracher. Double preconditioning for Gabor frames. *IEEE Trans. Signal Process.*, submitted for publication, 2005.

[6] P. Balazs, B. Laback, G. Eckel, and W. A. Deutsch. Perceptional sparsity modelled by simultaneous masking. preprint, 2007.

[7] M. J. Bastiaans and M. C. Geilen. On the discrete Gabor transform and the discrete Zak transform. *Signal Process.*, 49(3):151–166, 1996.

[8] J. Benedetto, C. Heil, and D. F. Walnut. Gabor systems and the Balian-Low theorem. In Feichtinger and Strohmer [35], pages 85–122.

[9] K. Bittner. Error estimates and reproduction of polynomials for biorthogonal local trigonometric bases. *Appl. Comput. Harmon. Anal.*, 6:75–102, 1999.

[10] K. Bittner. Wilson bases on the interval. In Feichtinger and Strohmer [36], chapter 9, pages 197–222.

[11] Å. Björck and C. Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM Jour. Num. Anal.*, 8(2):358–364, june 1971.

[12] T. Blu and M. Unser. The fractional spline wavelet transform: definition end implementation. *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, 1:512 –515 vol.1, 2000.

[13] T. Blu and M. Unser. A complete family of scaling functions: the (α, τ)-fractional splines. *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, 1:VI–421–4 vol.6, 2003.

[14] H. Bölcskei. Orthogonal frequency division multiplexing based on offset QAM. In Feichtinger and Strohmer [36], pages 321–352.

[15] H. Bölcskei, H. G. Feichtinger, K. Gröchenig, and F. Hlawatsch. Discrete-time Wilson expansions. In *Proc. IEEE-SP 1996 Int. Sympos. Time-Frequency Time-Scale Analysis*, june 1996.

[16] H. Bölcskei and F. Hlawatsch. Oversampled Wilson-type cosine modulated filter banks with linear phase. In *Asilomar Conf. on Signals, Systems, and Computers*, pages 998–1002, nov 1996.

[17] H. Bölcskei and F. Hlawatsch. Discrete Zak transforms, polyphase transforms, and applications. *IEEE Trans. Signal Process.*, 45(4):851–866, april 1997.

[18] H. Bölcskei, F. Hlawatsch, and H. G. Feichtinger. Equivalence of DFT filter banks and Gabor expansions. In *SPIE 95, Wavelet Applications in Signal and Image Processing III*, volume 2569, part I, San Diego, july 1995.

[19] R. Carmona, W. Hwang, and B. Torrésani. *Practical Time-Frequency Analysis: continuous wavelet and Gabor transforms, with an implementation in S*, volume 9 of *Wavelet Analysis and its Applications.* Academic Press, San Diego, 1998.

[20] R. Carmona, W. Hwang, and B. Torrésani. Multiridge detection and time-frequency reconstruction. *IEEE Trans. Signal Process.*, 47:480–492, 1999.

[21] O. Christensen. Frames and pseudo-inverses. *J. Math. Anal. Appl.*, 195:401–414, 1995.

[22] O. Christensen. Finite-dimensional approximation of the inverse frame operator. *J. Fourier Anal. Appl.*, 6(1):79–91, 2000.

[23] O. Christensen. *An Introduction to Frames and Riesz Bases.* Birkhäuser, 2003.

[24] O. Christensen and T. Strohmer. Methods for the approximation of the inverse (Gabor frame operator. In Feichtinger and Strohmer [36], chapter 8, pages 171–196.

[25] I. Daubechies, S. Jaffard, and J. Journé. A simple Wilson orthonormal basis with exponential decay. *SIAM J. Math. Anal.*, 22:554–573, 1991.

[26] I. Daubechies, H. Landau, and Z. Landau. Gabor time-frequency lattices and the Wexler-Raz identity. *J. Fourier Anal. Appl.*, 1(4):437–498, 95.

[27] J. Dongarra, J. Du Croz, S. Hammarling, and I. Duff. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Software*, 16(1):1–17, 1990.

[28] J. S. (ed.). *Digital Audio Signal Processing - An Anthology.* William Kaufmann, Inc., 1985.

[29] G. Fant. *Acoustic Theory of Speech Production.* Mouton. The Hague., 1960.

[30] H. G. Feichtinger. On a new Segal algebra. *Monatsh. Math.*, 92(4):269–289, 1981.

[31] H. G. Feichtinger and K. Gröchenig. Gabor frames and time-frequency analysis of distributions. *J. Func. Anal.*, 146:464–495, 1997.

[32] H. G. Feichtinger, M. Hazewinkel, N. Kaiblinger, E. Matusiak, and M. Neuhauser. Metaplectic operators on c^n. *To appear*, 2006.

[33] H. G. Feichtinger and W. Kozek. Operator quantization on LCA groups. In Feichtinger and Strohmer [35], chapter 7, pages 233–266.

[34] H. G. Feichtinger and K. Nowak. A first survey of Gabor multipliers. In Feichtinger and Strohmer [36], chapter 5, pages 99–128.

[35] H. G. Feichtinger and T. Strohmer, editors. *Gabor Analysis and Algorithms.* Birkhäuser, Boston, 1998.

[36] H. G. Feichtinger and T. Strohmer, editors. *Advances in Gabor Analysis.* Birkhäuser, 2003.

[37] H. G. Feichtinger and G. Zimmermann. A Banach space of test functions and Gabor analysis. In Feichtinger and Strohmer [35], chapter 3, pages 123–170.

[38] P. Flandrin. *Time-frequency/time-scale analysis*, volume 10 of *Wavelet Analysis and its Applications.* Academic Press Inc., San Diego, CA, 1999. With a preface by Yves Meyer, Translated from the French by Joachim Stöckler.

[39] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[40] G. H. Golub and C. F. van Loan. *Matrix computations, third edition.* John Hopkins University Press, 1996.

[41] K. Gröchenig. *Foundations of Time-Frequency Analysis.* Birkhäuser, 2001.

[42] K. Gröchenig, A. J. E. M. Janssen, N. Kaiblinger, and G. Pfander. Note on b-splines, wavelet scaling functions, and gabor frames. *Information Theory, IEEE Transactions on*, 49(12):3318–3320, 2003.

[43] K. Gröchenig and M. Leinert. Wiener's lemma for twisted convolution and Gabor frames. *J. Amer. Math. Soc.*, 17(1), 2004.

[44] N. J. Higham. Computing the polar decomposition—with applications. *SIAM J. Sci. Statist. Comput.*, 7(4):1160–1174, Oct. 1986.

[45] N. J. Higham. A new `sqrtm` for MATLAB. Numerical Analysis Report No. 336, Manchester Centre for Computational Mathematics, Manchester, England, Jan. 1999.

[46] N. J. Higham and R. S. Schreiber. Fast polar decomposition of an arbitrary matrix. *SIAM J. Sci. Statist. Comput.*, 11(4):648–655, July 1990.

[47] A. Hurwitz. Einige Eigenschaften der Dirichlet'schen Funktionen $F(s) = \sum(\frac{D}{n})\frac{1}{n^s}$, die bei der Bestimmung der Klassenanzahlen Binärer quadratischer Formen auftreten. *Z. für Math. und Physik*, 27:86–101, 1882.

[48] A. J. E. M. Janssen. Duality and biorthogonality for discrete-time Weyl-Heisenberg frames. Unclassified report, Philips Electronics, 002/94.

[49] A. J. E. M. Janssen. The Zak transform: a signal transform for sampled time-continuous signals. *Philips Journal of Research*, 43(1):23–69, 1988.

[50] A. J. E. M. Janssen. Signal analytic proofs of two basic results on lattice expansions. *Appl. Comput. Harmon. Anal.*, 1(4):350–354, 1994.

[51] A. J. E. M. Janssen. Duality and biorthogonality for Weyl-Heisenberg frames. *J. Fourier Anal. Appl.*, 1(4):403–436, 1995.

[52] A. J. E. M. Janssen. On rationally oversampled Weyl-Heisenberg frames. *Signal Process.*, pages 239–245, 1995.

[53] A. J. E. M. Janssen. From continuous to discrete Weyl-Heisenberg frames through sampling. *J. Fourier Anal. Appl.*, 3(5):583–596, 1997.

[54] A. J. E. M. Janssen. The duality condition for Weyl-Heisenberg frames. In Feichtinger and Strohmer [35], chapter 1, pages 33–84.

[55] A. J. E. M. Janssen. Analysis of some fast algorithms to compute canonical windows for Gabor frames. Unpublished, 2002.

[56] A. J. E. M. Janssen. Some iterative algorithms to compute canonical windows for Gabor frames. In *Proceedings of IMS Workshop on Time-Frequency Analysis and Applications*, Singapore, September 2003.

[57] A. J. E. M. Janssen and P. L. Søndergaard. Iterative algorithms to approximate canonical Gabor windows: Computational aspects. *J. Fourier Anal. Appl.*, published online, 2007.

[58] A. J. E. M. Janssen and T. Strohmer. Characterization and computation of canonical tight windows for Gabor frames. *J. Fourier Anal. Appl.*, 8(1):1–28, 2002.

[59] A. J. E. M. Janssen and T. Strohmer. Hyperbolic secants yield Gabor frames. *Appl. Comput. Harmon. Anal.*, 12(2):259–267, 2002.

[60] B. Jawerth and W. Sweldens. Biorthogonal smooth local trigonometric bases. *J. Fourier Anal. Appl.*, 2(2):109–133, 1995.

[61] S. Jayant and P. Noll. *Digital Coding of Waveforms: Principles and Applications to Speech and Video.* Prentice Hall, 1990.

[62] N. Kaiblinger. Approximation of the Fourier transform and the dual Gabor window. *J. Fourier Anal. Appl.*, 11(1):25–42, 2005.

[63] C. Kenney and A. Laub. On scaling Newton's method for polar decomposition and the matrix sign function. *Proceedings of the 1990 American Control Conference (IEEE Cat. No.90CH2896-9)*, pages 2560–4 vol.3, 1990.

[64] Z. Kovarik. Some iterative methods for improving orthonormality. *SIAM J. Num. Anal.*, 7(3):386–9, 1970.

[65] G. Kutyniok and T. Strohmer. Wilson bases for general time-frequency lattices. *SIAM J. Math. Anal.*, 37(3):685–711 (electronic), 2005.

[66] S. Lakic. An iterative method for the computation of a matrix inverse square root. *ZAMM Z. Angew. Math. Mech.*, 75(11):867–874, 1995.

[67] Y.-P. Lin and P. Vaidyanathan. Linear phase cosine modulated maximally decimated filter banks with perfect reconstruction. *IEEE Trans. Signal Process.*, 43(11):2525–2539, 1995.

[68] V. Losert. A characterization of the minimal strongly character invariant Segal algebra. *Ann. Inst. Fourier*, 30:129–139, 1980.

[69] P. Majdak and P. Balazs. Multiple exponential sweep method for fast measurement of head related transfer functions. submitted, 2006.

[70] S. Mallat. *A wavelet tour of signal processing.* Academic Press, San Diego, CA, 1998.

[71] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Process.*, 41(12):3397–3415, 1993.

[72] H. S. Malvar. *Signal Processing with Lapped Transforms.* Artech House Publishers, 1992.

[73] J. Markel and J. A. Gray. *Linear Prediction of Speech.* Springer-Verlag, Berlin Heidelberg, 1976.

[74] G. Matz and F. Hlawatsch. *Linear Time-Frequency Filters: On-line Algorithms and Applications*, chapter 6 in 'Application in Time-Frequency Signal Processing', pages 205–271. eds. A. Papandreou-Suppappola, Boca Raton (FL): CRC Press, 2002.

[75] D. Mugler, S. Clary, and Y. Wu. Discrete Hermite expansion of digital signals: applications to ECG signals. *Digital Signal Processing Workshop, 2002 and the 2nd Signal Processing Education Workshop. Proceedings of 2002 IEEE 10th*, pages 262–267, 2002.

[76] K. A. Okoudjou. Embeddings of some classical Banach spaces into modulation spaces. *Proc. Am. Math. Soc.*, 132(6), 2004.

[77] A. V. Oppenheim and R. Schafer. *Discrete-Time Signal Processing.* Oldenbourg, 3 edition, 1999.

[78] A. V. Oppenheim and R. W. Schafer. *Discrete-time signal processing.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[79] R. S. Orr. Derivation of the finite discrete Gabor transform by periodization and sampling. *Signal Process.*, 34(1):85–97, 1993.

[80] H. M. Ozaktas, Z. Zalevsky, and M. A. Kutay. *The Fractional Fourier Transform.* John Wiley and Sons, 2001.

[81] J. P. Princen and A. B. Bradley. Analysis/synthesis filter bank design based on time domain aliasing cancellation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-34(5):1153–1161, 1986.

[82] J. P. Princen, A. W. Johnson, and A. B. Bradley. Subband/transform coding using filter bank designs based on time domain aliasing cancellation. *Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2161–2164, 1987.

[83] P. Prinz. Calculating the dual Gabor window for general sampling sets. *IEEE Trans. Signal Process.*, 44(8):2078–2082, 1996.

[84] M. Puckette. Phase-locked vocoder. *Applications of Signal Processing to Audio and Acoustics, 1995., IEEE ASSP Workshop on*, pages 222 –225, 1995.

[85] S. Qiu. Discrete Gabor transforms: The Gabor-gram matrix approach. *J. Fourier Anal. Appl.*, 4(1):1–17, 1998.

[86] S. Qiu and H. G. Feichtinger. Discrete gabor structures and optimal representations. *IEEE Trans. Signal Process.*, 43(10):2258 –2268, 1995.

[87] K. Rao and P. Yip. *Discrete Cosine Transform, Algorithms, Advantages, Applications.* Academic Press, 1990.

[88] A. Ron and Z. Shen. Weyl-Heisenberg frames and Riesz bases in $l_2(\mathbb{R}^d)$. *Duke Math. J.*, 89(2):237–282, 1997.

[89] R. Schafer and J. M. (eds). *Speech Analysis.* IEEE Press, 1979.

[90] G. Schulz. Iterative Berechnung der reziproken Matrix. *ZAMM Z. Angew. Math. Mech.*, 13:57–59, 1933.

[91] N. Sherif. On the computation of a matrix inverse square root. *Computing (Vienna/New York)*, 46(4):295–305, 1991.

[92] P. L. Søndergaard. Gabor frames by sampling and periodization. *Adv. Comput. Math.*, published online, 2007.

[93] P. L. Søndergaard. An efficient algorithm for the discrete Gabor transform using full length windows. *IEEE Signal Process. Letters*, submitted for publication, 2007.

[94] P. L. Søndergaard. Symmetric, discrete fractional splines and Gabor systems. *Int. J. Wavelets Multiresolut. Inf. Process.*, submitted for publication, 2007.

[95] K. N. Stevens. *Acoustic Phonetics.* MIT Press. Cambridge, 1999.

[96] T. Strohmer. Numerical algorithms for discrete Gabor expansions. In Feichtinger and Strohmer [35], chapter 8, pages 267–294.

[97] T. Strohmer. Approximation of dual Gabor frames, window decay and wireless communications. *Appl. Comput. Harmon. Anal.*, 11:243–262, 2001.

[98] R. Tolimieri and R. S. Orr. Poisson summation, the ambiguity function, and the theory of Weyl-Heisenberg systems. *J. Fourier Anal. Appl.*, 1(3):233–247, 1995.

[99] M. Unser, A. Aldroubi, and M. Eden. On the asymptotic convergence of b-spline wavelets to gabor functions. *Information Theory, IEEE Transactions on*, 38(22):864 –872, 1992.

[100] M. Unser, A. Aldroubi, and M. Eden. B-spline signal processing. i. theory. *IEEE Trans. Signal Process.*, 41(2):821 –833, 1993.

[101] M. Unser and T. Blu. Fractional splines and wavelets. *SIAM Review*, 42(1):43–67, 2000.

[102] A. J. van Leest and M. J. Bastiaans. Gabor's discrete signal expansion and the discrete gabor transform on a non-separable lattice. *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, pages 101–4 vol.1, 2000.

[103] A. J. van Leest and M. J. Bastiaans. Implementations of non-separable Gabor schemes. *In proceddings of Eusipco 2004, the 12th European Signal Processing Conference*, pages 1–4, 2004.

[104] M. Vetterli and J. Kovačević. *Wavelets and Subband Coding.* Signal Processing Series. Prentice Hall, Englewood Cliffs, NJ, 1995.

[105] D. F. Walnut. *An introduction to wavelet analysis.* Birkhäuser, 2002.

[106] T. Werther, Y. Eldar, and N. Subbana. Dual Gabor Frames: Theory and Computational Aspects. *IEEE Trans. Signal Processing*, 53(11), 2005.

[107] J. Wexler and S. Raz. Discrete Gabor expansions. *Signal Process.*, 21(3):207–221, 1990.

[108] R. C. Whaley, A. Petitet, and J. Dongarra. Automated empirical optimization of software and the ATLAS project. Technical Report UT-CS-00-448, University of Tennessee, Knoxville, TN, Sept. 2000.

[109] M. V. Wickerhauser. *Adapted wavelet analysis from theory to software.* Wellesley-Cambridge Press, Wellesley, MA, 1994.

[110] B. Widrow and S. D. Stearns. *Adaptive Signal Processing.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

[111] Y. Y. Zeevi and M. Zibulski. Oversampling in the Gabor scheme. *IEEE Trans. Signal Process.*, 41(8):2679–2687, 1993.

[112] M. Zibulski and Y. Y. Zeevi. Analysis of multiwindow Gabor-type schemes by frame methods. *Appl. Comput. Harmon. Anal.*, 4(2):188–221, 1997.