The LTFAT tutorial

Jordy van Velthoven, Peter L. Søndergaard

Contents

Preface

The Large Time-Frequency Analysis Toolbox (LTFAT) is a collection of routines for time-frequency analysis and synthesis. It is intended both as an educational and a computational tool. It consists of a large number of linear transforms for Fourier analysis, Gabor analysis and wavelet analysis along with associated operators and plotting routines. The routines that are included in the toolbox are primarily programs written for Matlab and Octave, but the toolbox contains also MEX/OCT interfaces written in C/C++ that functions as a backend library.

The tutorial provides an introduction to the LTFAT by giving a summary of the basic methods used in the toolbox. Each chapter or section starts with the theory and mathematical background related to the routines and concludes with specifying how these methods can be used in LTFAT. For a complete overview of the toolbox the on-line documentation or the LTFAT reference manual can be consulted. The on-line documentation of the toolbox is available from: `http://ltfat.sourceforge.net/doc/start`, and the reference manual can be downloaded from: `http://ltfat.sourceforge.net/doc/ltfat.pdf`.

Terminology and notation

All functions and operators in the tutorial are finite-dimensional. Any $f \in \mathbb{C}^L$ is represented as a function $f : \mathbb{Z}_L \to \mathbb{C}$ where $\mathbb{Z}_L = \mathbb{Z}/L\mathbb{Z} = \{0, 1, ..., L-1\}$ is the commutative ring of integers modulo L. In this case, operations on the indices are computed modulo L. These indices have a cyclic indexing, that is, $f(l + kL) = f(l)$ for all $l, k \in \mathbb{Z}$.The value of the l'th element of f is denoted as $f(l)$ where $l \in \mathbb{Z}_L$. A linear operator $\mathcal{O} : \mathbb{C}^L \to \mathbb{C}^M$ is represented as a matrix multiplication $(\mathcal{O}f)(m) = \sum_{l=0}^{L-1} o(m, l)f(l)$ with $m \in \mathbb{Z}_M$ and where $\mathcal{O} = o(m, l)$ is a $M \times L$ matrix.

In Matlab and Octave all data structures are indexed starting by 1. However, all formulas in this tutorial are indexed starting from 0. The formulas in this tutorial can be translated into procedures for Matlab and Octave by adding + 1 to the argument when indexing a data structure.

To distinguish the mathematical background and the actual routines of LTFAT, the Matlab and Octave functions are called routines. To the names of these routines is referred in a typewriter style (`routine-name`). Aside the names of the routines, also the input and output parameters will be written in the typewriter style.

Chapter 1

Fourier analysis

Fourier analysis is based on the notion that general functions can be represented as a linear combination of sinusoids. In this case the domain into which a general function is mapped is the so-called frequency domain. The transform that maps a function into the frequency domain is the Fourier transform and the transform that constructs a function from its Fourier transform is called the inverse Fourier transform.

1.1 Periodic functions

Routines for periodic functions A vector $f \in \mathbb{C}^L$ can be interpreted as a periodic function. In this case, it is interpreted as a period of length L. If the index of such a vector is denoted by $l \in \mathbb{Z}_L$, then it is a L-periodic function of l on $\mathbb{Z}$, that is,

$$f(l + kL) = f(l), \quad k \in \mathbb{Z}.$$

Such a periodic function is called even if it satisfies for all $l \in \mathbb{Z}_L$ the following equation

$$f(l) = f(-l) \tag{1.1}$$

and it is called odd when it satisfies for all $l \in \mathbb{Z}_L$

$$-f(l) = f(-l). \tag{1.2}$$

In the LTFAT there are routines for getting the even or odd part of a periodic function. To get the even part of a periodic function, the routine `peven` can be used. This routine returns the even part of an input `f` as output. The routine `podd` does the same for the odd part of an input `f`.

A function $f \in \mathbb{C}^L$ is in general called symmetric if there is a $N \in \mathbb{Z}_L$ such that for all $l \in \mathbb{Z}_L$.

$$f(l) = f(N - l). \tag{1.3}$$

In this case, the function is symmetric around N. If a function is symmetric for an even N, it is called whole-point symmetric. If a function is symmetric for

an odd N, it is called half-point symmetric. In general is a function half-point symmetric when it is symmetric around $l = -\frac{1}{2}$, and whole-point symmetric if it is symmetric around $l = 0$.

Using these definitions, a function $f \in \mathbb{C}^L$ has half-point even symmetry when for all $l \in \mathbb{Z}_L$

$$f(l) = \overline{f(L-1-l)} \tag{1.4}$$

and has whole-point even symmetry when for all $l \in \mathbb{Z}_L$.

$$f(l) = \overline{f(-l)} = \overline{f(L-l)}. \tag{1.5}$$

In the same manner is a function $f \in \mathbb{C}^L$ called half-point odd symmetric when for all $l \in \mathbb{Z}_L$, $f(l) = -\overline{f(L-1-l)}$, and whole-point odd symmetryic when $f(l) = -\overline{f(-l)} = -\overline{f(L-l)}$ for all $l \in \mathbb{Z}_L$.

To check whether a vector is even, the LTFAT routine called `isevenfunction` can be used. The input of this routine is a function `f`, the output is a `1` if `f` is whole-point even and a `0` if it is not. The `isevenfunction` does the same for a half-point even function `f` if the additional flag `'hp'` is added as an input parameter.

A function that has half-point or whole-point even symmetry can be symmetrically extended or cut using `middlepad`. This routine extends or cuts the function by inserting zeros in the middle of the vector or by cutting in the middle of the vector. The input parameters of `middlepad` are a function `f` and the length `L` to which the input should be extended or cut. Adding the additional flag `'wp'` or `'hp'` as an input parameter of `middlepad` will cut or extend functions with a whole-point or half-point even symmetry.

Examples of periodic functions The LTFAT contains a collection of periodic functions. Examples of periodic functions that are included in the LTFAT are the complex exponential, chirp, rectangular function and sinc function. All these functions are characterized by their properties in relation to the Fourier transform or a time-frequency transform.

A discrete complex exponential h with discrete frequency k is computed as

$$h(l) = e^{2\pi i l k/L}, \quad l \in \mathbb{Z}_L. \tag{1.6}$$

The collection of complex exponentials with $k = 0, ..., L-1$, that is, the collection $\{e^{2\pi i l k/L}\}_{k,l\in\mathbb{Z}_L}$, forms the basis of the discrete Fourier transform. The LTFAT routine for a complex exponential is called `expwave` and has the length `L` and discrete frequency `k` as input parameters.

A periodic, discrete chirp g is computed by

$$g(l) = e^{\pi i n(l-\lceil L/2\rceil)^2(L+1)/L}, \quad l \in \mathbb{Z}_L. \tag{1.7}$$

Such a chirp revolves $n \in \mathbb{Z}$ times around the time-frequency plane in frequency. The LTFAT routine for the periodic, discrete chirp is called `pchirp` and has the length `L` and number of revolutions around the time-frequency plane in frequency `n` as input parameters.

The discrete, periodic rectangle function is given by

$$r(l) = \begin{cases} 1, & \text{if } |l| \leq \frac{1}{2}(n-1) \\ 0, & \text{otherwise} \end{cases}, \quad l \in \mathbb{Z}_L \tag{1.8}$$

where $n \in \mathbb{Z}_L$ denotes the length of the support of r. The discrete, periodic rectangle function is related to a discrete, periodic sinc function through a discrete Fourier transform. The discrete periodic rectangle and sinc function form therefore a so-called Fourier pair. The discrete, periodic sinc function p of order n is given by

$$p(l) = \frac{\sin(n\pi l)}{n \sin(\pi l)}, \quad l \in \mathbb{Z}_L \tag{1.9}$$

The discrete, periodic rectangle and sinc function are implemented in LTFAT by the routines `prect` and `psinc`, respectively. Both routines have the length of the function `L` and the associated number `n` as input parameters.

Window functions A window function is a function that is in general centered around the origin and zero-valued outside a certain interval. In the LTFAT both so-called Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) windows functions are implemented. A FIR window is in this case defined as a window function whose length of its support is shorter than its total length L and an IIR window is a window function whose length of its support equals its total length L. Both the FIR and IIR window functions are implemented as whole-point even functions.

Examples of the IIR windows functions that are included in the LTFAT are the periodic Gaussian, B-spline, Sech and Gauss-Hermite window functions. These functions are implemented in the LTFAT as the routines `pgauss`, `pbspline`, `psech` respectively `pherm`. They all have their length `L` as input parameter and the `pbsline` and `pherm` have their order `a` as additional input parameter. The periodic, discrete B-spline, Sech and Gauss-Hermite window functions are all related to the periodic, discrete Gaussian: The B-spline functions converge to a Gaussian when its order grows, the Sech distribution can be defined in terms of a Gaussian and the Gauss-Hermite functions are defined as the product of a Hermite polynomial and the Gaussian. The periodic, discrete Gaussian $\varphi \in \mathbb{C}^L$ is given by

$$\varphi(l) = \left(\frac{L}{2}\right)^{-1/4} \sum_{k=0}^{L-1} e^{-\pi(\frac{l}{\sqrt{L}} - k\sqrt{L})^2}, \quad l \in \mathbb{Z}_L \tag{1.10}$$

and is invariant under the discrete Fourier transform.

Examples of FIR window functions that are included in the LTFAT are the Von Hann, sine, triangular, the Hamming and Blackman window functions. These functions can all be derived from the routine `firwin` which has as input parameters the name of the window function `(FIR-name)` and its length `L`.

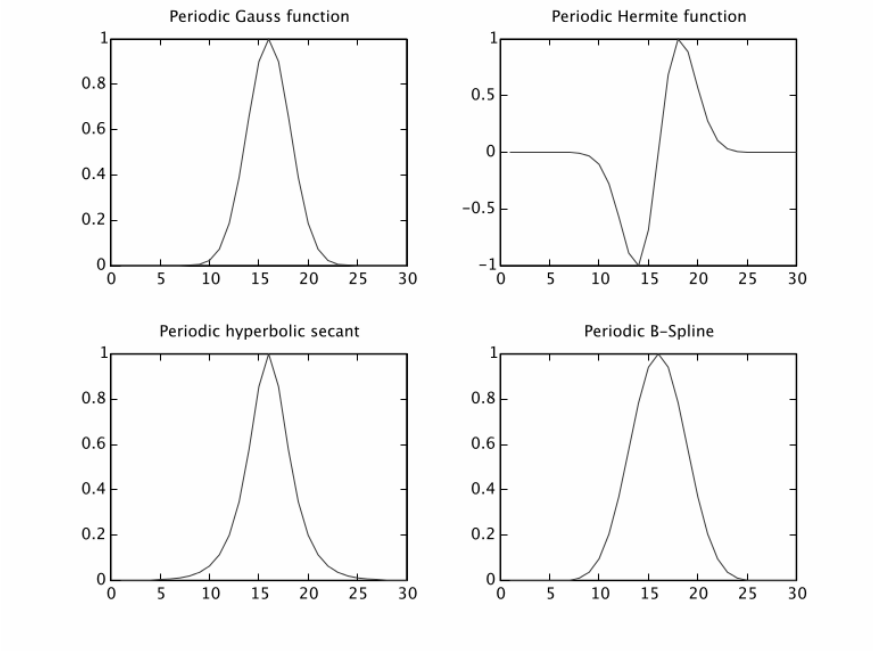

Figure 1.1: IIR window functions in the time domain

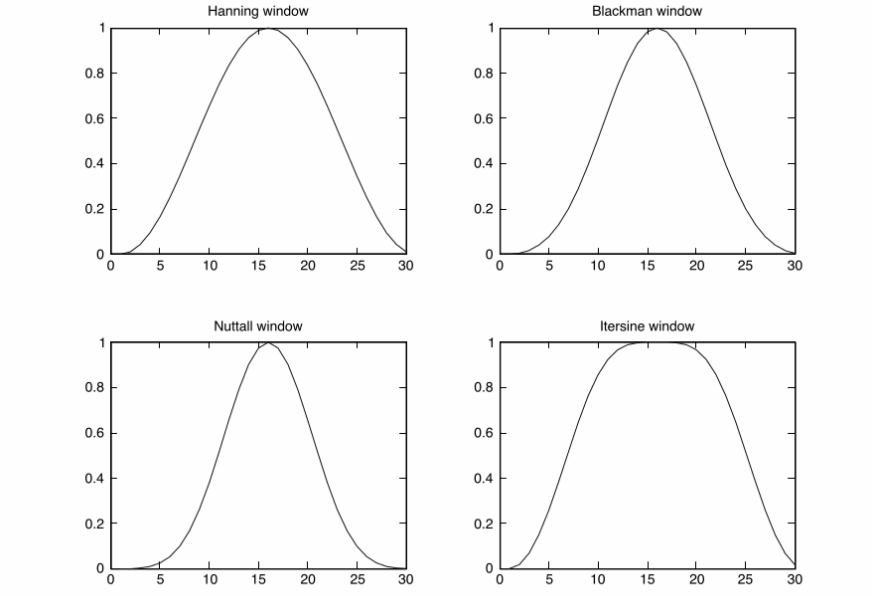

Figure 1.2: FIR window functions in the time domain

1.2 Discrete Fourier transforms

Discrete Fourier transform The discrete Fourier transform represents a function in terms of discrete complex exponentials. The discrete Fourier transform $\mathcal{F}$ of a function $f \in \mathbb{C}^L$ is defined as

$$(\mathcal{F}f)(k) = \sum_{l=0}^{L-1} f(l)\, e^{-2\pi i k l/L}, \quad k \in \mathbb{Z}_L. \tag{1.11}$$

The discrete Fourier transform that is implemented in the LTFAT is the unitary discrete Fourier transform. The unitary discrete Fourier transform differs from an ordinary discrete Fourier transform in that the output is scaled by $\frac{1}{\sqrt{L}}$.

The unitary discrete Fourier transform satisfies Parseval's and Placherel's equality. Parseval's equality is given by

$$\sum_{l=0}^{L-1} f(l)\overline{g(l)} = \sum_{k=0}^{L-1} (\mathcal{F}f)(k)\overline{(\mathcal{F}g)(k)}. \tag{1.12}$$

and Plancherel's equality can be obtained from (1.12) by setting $f = g$,

$$\sum_{l=0}^{L-1} |f(l)|^2 = \sum_{k=0}^{L-1} |(\mathcal{F}f)(k)|^2, \tag{1.13}$$

The unitary discrete Fourier transform can be written as a matrix product as

$$(\mathcal{F}f)(k) = \mathbf{F}_L f \tag{1.14}$$

where $\mathbf{F}_L$ is the so-called Fourier matrix of size $L \times L$ given by

$$\mathbf{F}_L = \frac{1}{\sqrt{L}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{(L-1)} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(L-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{(L-1)} & \omega^{2(L-1)} & \ldots & \omega^{(L-1)(L-1)} \end{pmatrix} \tag{1.15}$$

where $\omega = e^{-i2\pi/L}$.

The unitary discrete Fourier transform is implemented in the LTFAT as the routine `dft`. The input of the `dft` is a vector, e.g. the unitary discrete Fourier transform of a vector `f` can be calculated by `dft(f)`. The output of the `dft` is a vector that contains the values of the discrete Fourier transform of `f` scaled by the constant $L^{-\frac{1}{2}}$. If the input vector is real valued, then the routine `fftreal` could be used which computes only the coefficients associated with the positive frequencies of the function. To plot the output vector of `dft` or `plotfftreal` the routines `plotfft` and `plotfftreal` can be used, respectively.

Inverse Fourier transform A function can be constructed from its Fourier transform $c \in \mathbb{C}^L$ by the inverse discrete Fourier transform $\mathcal{F}^{-1}$ given by

$$(\mathcal{F}^{-1}c)(l) = \frac{1}{L}\sum_{k=0}^{L-1} c\,(k)\, e^{2\pi i k l/L}, \quad l \in \mathbb{Z}_L. \tag{1.16}$$

The inverse discrete Fourier transform that is implemented in the LTFAT is unitary and equals the inverse discrete Fourier transform defined in (6) scaled by $\sqrt{L}$. The unitary discrete Fourier transform can be expressed as a matrix product as

$$f = \mathbf{F}_L^{-1} c = \overline{\mathbf{F}^T}_L c \tag{1.17}$$

where $\overline{\mathbf{F}^T}_L$ is the conjugate transpose of the Fourier matrix $\mathbf{F}_L$. The unitary discrete Fourier transform is implemented by the routine `idft`. The input of the routine `idft` is a vector `c`, the output of `idft` is the vector `f`.

Example 1. In listing 1.1 the function `bat` is perfectly reconstructed from its Fourier transform.

```
f = bat;
c = dft(f);
r = idft(c);
norm(r - f)
```

Listing 1.1: dft_idft.m

The output of listing 1.1 is

```
ans =    4.3653e-16
```

Listing 1.2: Output of listing 1.1

which shows that there is no remarkable difference between the original function f and its reconstruction r. $\Diamond$

Discrete fractional fourier transform The discrete fractional Fourier transform is a generalization of the discrete Fourier transform. The discrete fractional Fourier transform can be interpreted as a discrete Fourier transform to the power n, that is, $\mathcal{F}^n$. For example, $\mathcal{F}^n$ with $n = 1, 2, 3, 4, ...$ of a function $f(l)$ is

$$(\mathcal{F}^2 f)(-l) = f(-l),\ (\mathcal{F}^3 f)(-k) = c(-k),\ (\mathcal{F}^4 f)(l) = f(l),\ ...$$

The discrete Fourier transform, $\mathcal{F}^1$, maps a function from the so-called time-domain to the so-called frequency domain, that is, a rotation of $\frac{\pi}{2}$ radians in the time-frequency plane. In general, $\mathcal{F}^n$ with $n \in \mathbb{R}$ corresponds to a rotation of $\frac{n\pi}{2}$ radians in the time-frequency plane and allow representations between the time and the frequency domain.

The routine in LTFAT for the discrete fractional Fourier transform is `dfract` and has a function `f` and power `a` as input arguments, e.g. `dfract(f,a)` computes the discrete Fourier transform to the power `a` of the vector `f`.

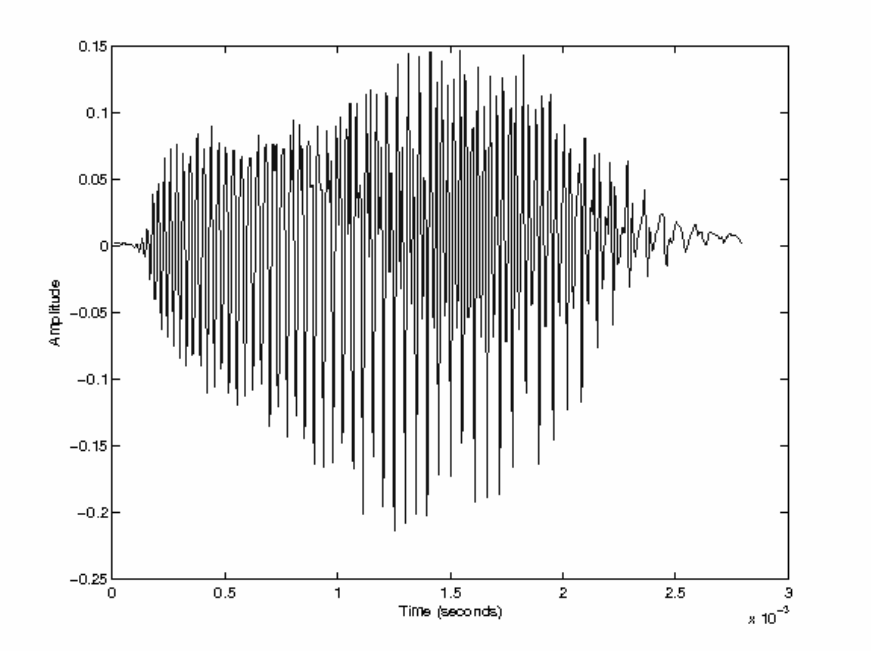

Figure 1.3: Bat signal in time domain

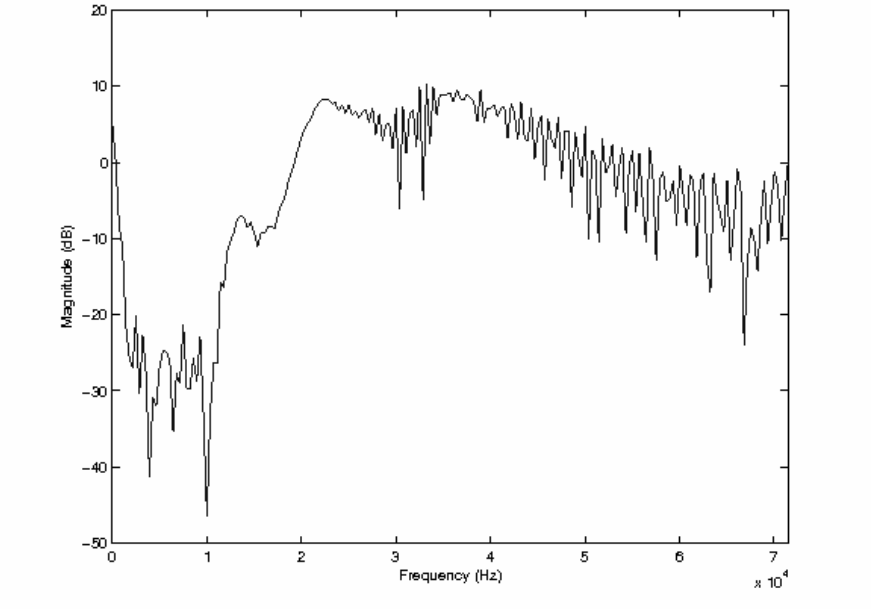

Figure 1.4: Bat signal in frequency domain

1.3 Operations for periodic functions

Involution An involution is in general an operator $\diamond$ that satisfies

$$(f^\diamond)^\diamond = f. \tag{1.18}$$

In Fourier analysis, the involution $f^\diamond$ of a function $f \in \mathbb{C}^L$ is defined as

$$f^\diamond(l) \;=\; \overline{f(-l)} \tag{1.19}$$

for $l \in \mathbb{Z}_L$. That $f^\diamond$ satisfies (1.18) is easily verified by

$$(f^\diamond)^\diamond(l) = \overline{f^\diamond(-l)} = \overline{\overline{f(-(-l))}} = f(l). \tag{1.20}$$

Involution is related to the discrete Fourier Transform $\mathcal{F}$ by

$$(\mathcal{F}f^\diamond)(k) = \overline{(\mathcal{F}f)}(k). \tag{1.21}$$

The inverse discrete Fourier transform $\mathcal{F}^{-1}$ of $\mathcal{F}f = c$ can therefore be expressed in terms of involution as

$$(\mathcal{F}^{-1}c)(l) = (\overline{\mathcal{F}^\diamond c})(l). \tag{1.22}$$

The routine for the involution of a function as defined in (1.19) is implemented in the LTFAT as `involute` and takes a function `f` as input.

Example 2. In listing 1.3 the relations between involution, conjugation and the discrete Fourier transform are given and verified.

```
f = bat;
f_i = involute(f);

c = dft(f);
r = idft(c);
r_i = conj(involute(dft(c)));
norm(r - r_i)
```

Listing 1.3: dft_involute.m

```
ans =    3.4391e-16
```

Listing 1.4: Output of listing 1.3

The output of listing 1.3, listing 1.4, shows the relations between involution, conjugation and the discrete Fourier transform. $\Diamond$

Periodic convolution The periodic or circular convolution h of two functions $f, g \in \mathbb{C}^L$ is given by

$$h(l) = (f \circledast g)(l) = \sum_{n=0}^{L-1} f(n)g(l-n), \quad l \in \mathbb{Z}_L. \tag{1.23}$$

It can be deduced from direct computations that the periodic convolution is commutative, associative and distributive over addition. From direct computations it can also be deduced that the periodic convolution of two functions $f, g \in \mathbb{C}^L$ is related to the discrete Fourier transforms of f and g by the relations

$$(\mathcal{F}(f \circledast g))(k) = (\mathcal{F}f \cdot \mathcal{F}g)(k), \tag{1.24}$$

$$(\mathcal{F}(f \cdot g))(k) = (\mathcal{F}f \circledast \mathcal{F}g)(k). \tag{1.25}$$

If $(\mathcal{F}g)(k) \neq 0$ for all $k \in \mathbb{Z}_L$, then the function f can be reconstructed from the convolution product h, as defined in (1.23), by

$$(\mathcal{F}f)(k) = \frac{(\mathcal{F}h)(k)}{(\mathcal{F}g)(k)}, \tag{1.26}$$

and applying the inverse discrete Fourier transform on $(\mathcal{F}f)(k)$.

The periodic convolution of f and the involution of g, $g^\diamond$, is in general called periodic cross-correlation and is given by

$$x(l) = (f \circledast g^\diamond)(l) = \sum_{n=0}^{L-1} f(n)\overline{g(n-l)}. \tag{1.27}$$

The periodic cross-correlation of a function f with itself is in general called the periodic autocorrelation of f and is given by.

$$a(l) = (f \circledast f^\diamond)(l) = \sum_{n=0}^{L-1} f(n)\overline{f(n-l)}. \tag{1.28}$$

The periodic cross-correlation x and autocorrelation a are, since they can be defined in terms of convolution and involution, also related to the discrete Fourier transform. The periodic cross-correlation is related to the discrete Fourier transform as

$$(\mathcal{F}(f \circledast g^\diamond))(k) = (\mathcal{F}f \cdot \overline{\mathcal{F}g})(k) \tag{1.29}$$

and the periodic autocorrelation is related to the discrete Fourier transform as

$$(\mathcal{F}(f \circledast f^\diamond))(k) = (\mathcal{F}f \cdot \overline{\mathcal{F}f})(k) = |\mathcal{F}f(k)|^2 \tag{1.30}$$

The periodic convolution and periodic cross-correlation are implemented in the LTFAT by the routines `pconv` and `pxcorr`, respectively. For both routines the inputs are vectors `f` and `g`.

Linear convolution The linear convolution h of two functions $f, g \in \mathbb{C}^{\mathbb{Z}}$ is given by

$$h(l) = (f * g)(l) = \sum_{n \in \mathbb{Z}} f(n) g(l-n), \quad l \in \mathbb{Z}. \tag{1.31}$$

For two functions with a finite length L, that is, $f, g \in \mathbb{C}^L$ this becomes

$$h(l) = (f * g)(l) = \sum_{n=0}^{l} f(n) g(l-n), \quad l \in \mathbb{Z}_L$$

Since the periodic convolution of a L-periodic function as defined in (1.23) can be written as

$$h(l) = (f \circledast g)(l) = \sum_{n=0}^{l} f(n) g(l-n) + \sum_{n=l+1}^{L-1} f(n) g(l-n)$$

it can be deduced that the linear convolution of two functions f and g with finite length is equivalent to a periodic convolution of these functions when the period L_h of the periodic convolution satisfies

$$L_h \geq L_f + L_g - 1 \tag{1.32}$$

where L_f is the length of the function f and L_g the length of the function g. Since the autocorrelation and cross-correlation functions can be defined in terms of convolution they also have their linear versions.

The routines for linear convolution and linear cross-correlation in LTFAT are `lconv` and `lxcorr`, respectively. The input of those routines are two vectors `f` and `g`.

Example 3. In listing 1.5 the equivalence of linear and periodic convolution is verified. In this example the routine `postpad` extends a vector `x` to a length `N` by zero-padding.

```
f = rand(100,1);
g = rand(100,1);
L = length(f) + length(g) - 1;

h_p = pconv(postpad(f, L), postpad(g, L));
h_l = lconv(f,g);

norm(h_p - h_l)
```

Listing 1.5: lconv_pconv.m

The output of listing 1.5 is given in listing 1.6,

```
ans =    0
```

Listing 1.6: Output of listing 1.5

which shows the equivalence of the periodic and linear convolution. $\Diamond$

Chapter 2

Gabor analysis

Gabor analysis is based on the notion that general functions can be represented as a linear combination of translated and modulated window functions. In this case a function is mapped from the time domain to a time-frequency domain. The transform that maps a function from the time domain to a time-frequency domain is in general called a time-frequency transform. The inverse transform, that maps a function from the time-frequency domain to the time domain is called an inverse time-frequency transform. Time-frequency transforms that are associated with translated and modulated window functions are called Gabor transforms.

2.1 Time-frequency shifts

The translation operator $\mathcal{T}_n$ for a function $f \in \mathbb{C}^L$ is given by

$$(\mathcal{T}_n f)(l) = f(l-n), \quad l, n \in \mathbb{Z}_L \tag{2.1}$$

The modulation operator $\mathcal{M}_k$ for a function $f \in \mathbb{C}^L$ is given by

$$(\mathcal{M}_k f)(k) = f(l) e^{i2\pi kl/L}, \quad l, k \in \mathbb{Z}_L \tag{2.2}$$

The translation operator shifts the index of a function by n, the modulation operator modulates a function by k. The translation operator $\mathcal{T}_n$ and modulation operator $\mathcal{M}_k$ satisfy the following commutation relations

$$(\mathcal{T}_n \mathcal{M}_k f)(l) = e^{i2\pi k(l-n)/L} f(l-n), \quad l, k, n \in \mathbb{Z}_L \tag{2.3}$$

$$= e^{-i2\pi kn/L} (\mathcal{M}_k \mathcal{T}_n f)(l), \quad l, k, n \in \mathbb{Z}_L \tag{2.4}$$

The translation operator and modulation operator are related to the discrete Fourier transform as

$$\mathcal{F}(\mathcal{T}_n f) = \mathcal{M}_{-n}(\mathcal{F} f), \tag{2.5}$$

respectively

$$\mathcal{F}(\mathcal{M}_k f) = \mathcal{T}_k(\mathcal{F} f). \tag{2.6}$$

These relations can be verified by the direct computations

$$\begin{aligned}(\mathcal{F}(\mathcal{T}_n f))(k) &= \sum_{l=0}^{L-1} f(l-n)e^{-2\pi ikl/L} \\ &= \sum_{p=-n}^{L-1-n} f(p)e^{-2\pi ik(p+n)/L}, \quad p := l-n \\ &= \left(\sum_{p=0}^{L-1} f(p)e^{-2\pi ikp/L}\right)e^{-2\pi ikn/L} \\ &= (\mathcal{M}_{-n}(\mathcal{F}f))(-n)\end{aligned}$$

$$\begin{aligned}(\mathcal{F}(\mathcal{M}_b f))(k) &= \sum_{l=0}^{L-1} (f(l)e^{i2\pi bl/L})e^{-2\pi ikl/L} \\ &= \sum_{l=0}^{L-1} f(l)e^{-2\pi i(k-b)l/L} \\ &= (\mathcal{T}_b(\mathcal{F}f))(b)\end{aligned}$$

From this it can be deduced that the modulation operator $\mathcal{M}_b$ applied on a function f corresponds with a shift of the discrete Fourier transform of the-function f by b. For this reason is the modulation operator also called the frequency-shift operator. In the same vein can the translation operator $\mathcal{T}_n$ be called the time-shift operator.

A so-called time-frequency shift operator can be defined by the concatenation of the time-shift operator and frequency-shift operator. The time-frequency shift operator π_Λ for a function $f \in \mathbb{C}^L$ is given by

$$(\pi_\Lambda f)(\Lambda) = (\mathcal{M}_k \mathcal{T}_n f)(k,n), \quad \Lambda = (k,n) \in \mathbb{Z}_L \times \mathbb{Z}_L \tag{2.7}$$

A time-frequency shifted version of a function $g \in \mathbb{C}^L$ is thus given by

$$(\mathcal{M}_k \mathcal{T}_n g)(k,n) = g(l-n)e^{i2\pi kl/L} \tag{2.8}$$

If the function g in (2.8) is a window function, then a collection of such time-frequency shifted window functions is given by

$$\{(\pi_\Lambda g)(\Lambda)\}_{\Lambda \in \mathbb{Z}_L^2} = \{\mathcal{M}_k \mathcal{T}_n g\}_{k,n \in \mathbb{Z}_L} \tag{2.9}$$

and is called a local Fourier basis. Since the time-shift operator localizes a function in time and the frequency-shift operator localizes a function in frequency, a local Fourier basis consists of functions that are both localized in time and localized in frequency.

2.2 Discrete short-time Fourier transform

The short-time Fourier transform represents a function in terms of time-frequency shifted versions of a window function. The discrete short-time Fourier transform $\mathcal{V}_g$ of a function $f \in \mathbb{C}^L$ is given by

$$(\mathcal{V}_g f)(k,n) = \langle f, \mathcal{M}_k \mathcal{T}_n g \rangle, \quad k,n \in \mathbb{Z}_L \tag{2.10}$$

$$= \sum_{l=0}^{L-1} f(l)\overline{g\,(l-n)}e^{-2\pi ikl/L}, \quad k,n \in \mathbb{Z}_L \tag{2.11}$$

where g is a window function. The discrete short-time Fourier transform maps a function $f \in \mathbb{C}^L$ to a matrix $c \in \mathbb{C}^{L\times L}$ of coefficients. In this matrix k denotes the frequency around which the window function g of (2.10) is positioned and n denotes time around which the window function is positioned. Therefore the sequence $(\mathcal{V}_g f)(k,n)_{k=0,1,\ldots,L-1}$ where $n \in \mathbb{Z}_L$ is fixed, denotes the discrete Fourier transform coefficients of the function $f(l)g(l-n)$.

The discrete short-time Fourier transform satisfies Moyal's formula which is in this case explicitly given by

$$\sum_{k=0}^{L-1}\sum_{n=0}^{L-1}(\mathcal{V}_{g_1} f_1)(k,n)\overline{(\mathcal{V}_{g_2} f_2)(k,n)} = \left(\sum_{l=0}^{L-1} f_1(l)\overline{f_2(l)}\right)\left(\sum_{l=0}^{L-1} g_1(l)\overline{g_2(l)}\right) \tag{2.12}$$

where $f_1, f_2 \in \mathbb{C}^L$ are general functions and $g_1, g_2 \in \mathbb{C}^L$ are window functions. By setting $f = f_1 = f_2$ and $g = g_1 = g_2$ in (2.12), the following equality is obtained as a consequence of Moyal's formula

$$\sum_{k=0}^{L-1}\sum_{n=0}^{L-1}|(\mathcal{V}_g f)(k,n)|^2 = \left(\sum_{l=0}^{L-1}|f\,(l)|^2\right)\left(\sum_{l=0}^{L-1}|g\,(l)|^2\right). \tag{2.13}$$

From this formula the inverse discrete Fourier transform can be derived. The inverse discrete short-time Fourier transform $\mathcal{V}_g^{-1}$ of $c \in \mathbb{C}^{L\times L}$ is

$$(\mathcal{V}_g^{-1} c)(l) = \frac{1}{\|g\|^2}\sum_{m=0}^{L-1}\sum_{n=0}^{L-1} c(m,n)g(l-n)e^{2\pi ikl/L}, \quad l \in \mathbb{Z}_L. \tag{2.14}$$

The inversion formula of the discrete short-time Fourier transform clearly shows that a function f can be represented in terms of time-frequency shifted versions of a window function.

Spectrogram The spectrogram is a time-frequency representation based on the short-time Fourier transform. The spectrogram is defined as the squared modulus of the short-time Fourier transform, that is, $|c(m,n)|^2$, where $c(m,n)$ denotes the coefficients of the short-time Fourier transform. The spectrogram is implemented in LTFAT as the routine `sgram` and is mainly meant as a plotting routine. The input parameter of `sgram` is a function `f`. The spectrogram could be inverted trough an iterative algorithm which is available as the routine `isgram` in LTFAT.

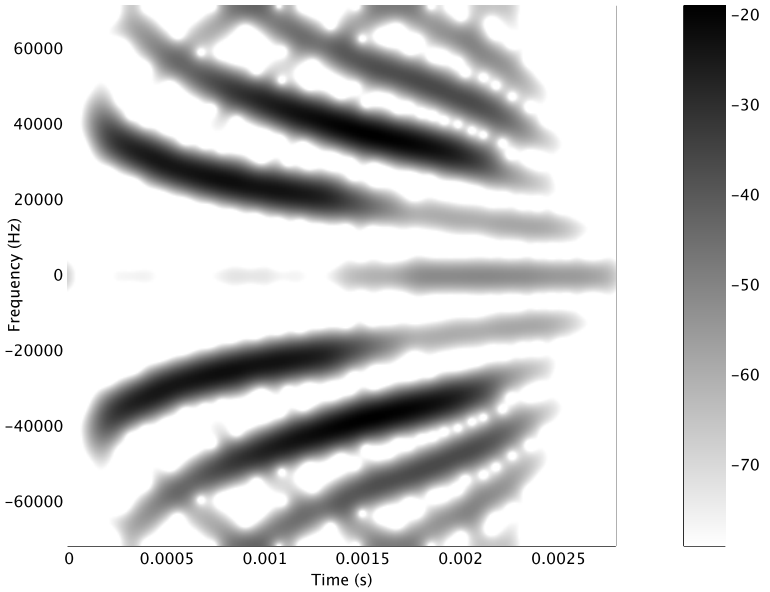

Figure 2.1: Spectrogram of bat signal

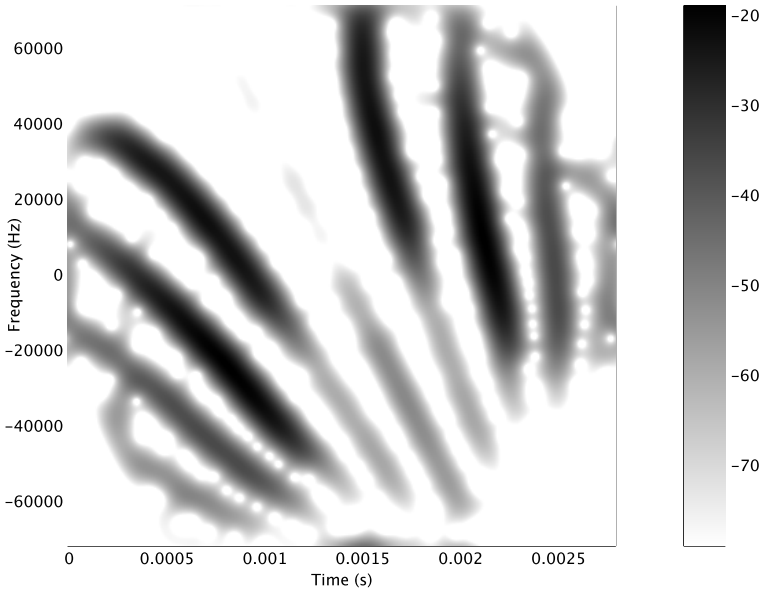

Figure 2.2: Spectrogram of a fractional Fourier transform of bat signal.

2.3 Discrete Gabor transform

The discrete short-time Fourier transform as defined in (2.10) has a redundancy L since it represents a function $f \in \mathbb{C}^L$ as a matrix $c \in \mathbb{C}^{L \times L}$. A subsampled version of a short-time Fourier transform is in general called a Gabor transform.

The Gabor transform of a function $f \in \mathbb{C}^L$ represents a function in terms of a collection of subsampled, time-frequency shifted versions of a window function. Such a window function is given by

$$(\mathcal{M}_{mb}\mathcal{T}_{na}g)(m,n), \quad (m,n) \in \mathbb{Z}_M \times \mathbb{Z}_N \tag{2.15}$$

where $L = Mb = Na$. Here a denotes the time-shift, b the frequency-shift and N and M the number of time-shifts respectively the number of frequency-shifts. A collection of such subsampled, translated and modulated window functions $g \in \mathbb{C}^L$,

$$\{\pi_\Lambda g\}_{\Lambda \subseteq \mathbb{Z}_M \times \mathbb{Z}_N} = \{\mathcal{M}_{mb}\mathcal{T}_{na}\}_{m \in \mathbb{Z}_M, n \in \mathbb{Z}_N} \tag{2.16}$$

is called a Gabor or Weyl-Heisenberg system. The discrete Gabor transform represents a function in terms of elements of such a Gabor system. The Gabor transform $\mathcal{G}$ of a function $f \in \mathbb{C}^L$ is therefore given by

$$(\mathcal{G}f)(m,n) = \langle f, \mathcal{M}_{mb}\mathcal{T}_{na}g \rangle, \tag{2.17}$$

$$= \sum_{l=0}^{L-1} f(l)\overline{g\,(l-na)}e^{-2\pi imbl/M}. \tag{2.18}$$

where $m \in \mathbb{Z}_M$, $n \in \mathbb{Z}_N$ and $L = Mb = Na$. The Gabor transform maps a function $f \in \mathbb{C}^L$ into a matrix $c \in \mathbb{C}^{M \times N}$ of so-called Gabor coefficients. The redundancy of the discrete Gabor transform is in general given by the ratio $\frac{MN}{L}$. Since for $a = b = 1$ the Gabor transform equals the discrete short-time Fourier transform, it can be deduced that the discrete short-time Fourier transform of a function $f \in \mathbb{C}^L$ has indeed a redundancy L while the Gabor transform can in general have a much lower redundancy and still perfectly reconstruct a function by a corresponding inverse Gabor transform. The inverse Gabor transform $\mathcal{G}^{-1}$ of $c \in \mathbb{C}^{M \times N}$ is given by

$$(\mathcal{G}^{-1}c)(l) = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} c\,(m,n)\,e^{2\pi iml/M}\gamma\,(l-an)\,, \quad l \in \mathbb{Z}_L \tag{2.19}$$

where γ is a so-called dual window function of g. A window function $\gamma \in \mathbb{C}^L$ is called a dual window function of $g \in \mathbb{C}^L$ if it satisfies

$$\langle \gamma, \mathcal{T}_{n/b}\mathcal{M}_{m/a}g \rangle = ab\delta_{n,0}\delta_{m,0}, \quad m \in \mathbb{Z}_M,\ n \in \mathbb{Z}_N \tag{2.20}$$

The construction of these dual window functions is discussed in chapter 4.

The discrete Gabor transform is implemented in LTFAT as the routine `dgt`. It has as input parameters a vector `f`, window function `g`, time-shift `a` and number of frequency shifts `M`. The output of `dgt` is a matrix `c` with Gabor coefficients. If the input vector `f` is real valued, then the function `dgtreal` could be used instead. This routine computes only the coefficients associated with the positive frequencies. To construct a window function that is suitable to use with a Gabor transform, the function `gabwin` can be used. The input parameters of `gabwin` are exactly the same as `dgt`. To plot the matrix of Gabor coefficients `c` the routine `plotdgt` or `plotdgtreal` can be used. The input parameters of `plotdgt` are the Gabor coefficients `c` and the time-shift `a`. The input parameters of `plotdgtreal` are the same as `plotdgt`, but also needs the number of frequency shifts `M`. The routine that computes the inverse discrete Gabor transform is called `idgt` and has as input parameters the Gabor coefficients `c`, window function `h` and time-shift `a`. The inverse discrete Gabor transform for the Gabor coefficients that are computed using the routine `dgtreal` is called `idgtreal` and has the same input parameters as `idgt`, but needs also the number of frequency shifts `M`. To construct a dual window function `h` of a window function `g` the routine `gabdual` can be used. This routine has the window function `g`, time-shift `a` and number of frequency shifts `M` as input parameters.

Example 4. In listing 2.1 a random vector is constructed from its discrete Gabor transform.

```
f = rand(1000,1);
g = gabwin('gauss', 5, 100, 1000);
c = dgt(f, g, 5, 100);

h = gabdual(g, 5, 100);
r = idgt(c, h, 5);

norm(f-r)
```

Listing 2.1: dgt_rand.m

The output of listing 2.1 is listed in listing 2.2.

```
ans =    5.7459e-15
```

Listing 2.2: Output of listing 2.1

Listing 2.2 shows that there isn't any significant difference between the original random vector and its reconstruction from Gabor coefficients. The output of listing 2.1 when the window function `g` was used in both discrete Gabor transforms is given in listing 2.3.

```
ans =  345.69
```

Listing 2.3: Output of listing 2.1 with `g` equal to `h`

This shows that perfect reconstruction without duals Gabor windows isn't possible in general. ◊

2.3.1 Lattices associated with Gabor transforms

The Gabor transform represents in general the discrete short-time Fourier transform sampled at a lattice. The discrete Gabor transform defined in (2.17) represents the discrete short-time Fourier transform at the points (an, bm) where $n \in \mathbb{Z}_N, m \in \mathbb{Z}_M, a, b, N, M \in \mathbb{Z}_L$ with $L = Na = Mb$. This corresponds to a discrete short-time Fourier transform sampled at a lattice Λ of the form

$$\Lambda = \left\{ (an, bm) \;\middle|\; n \in \mathbb{Z}_N,\ m \in \mathbb{Z}_M \right\} \tag{2.21}$$

This lattice $\Lambda \subseteq \mathbb{Z}_L^2$. Here $\mathbb{Z}_L^2$ is the lattice that corresponds with the short-time Fourier transform as defined in (2.10). A lattice as defined in (2.21) is in general called separable or regular. In general, such a lattice $\Lambda \subseteq \mathbb{Z}_L \times \mathbb{Z}_L$ can be written as

$$\Lambda = A(\mathbb{Z}_L \times \mathbb{Z}_L), \quad A \in \mathbb{Z}_L^{2\times 2} \tag{2.22}$$

If $A \in \mathbb{Z}_L^{2\times 2}$ is of the form

$$\begin{pmatrix} a & 0 \\ s & b \end{pmatrix} \tag{2.23}$$

with $a, b | L, 0 \leq s < b$ and $s \in \frac{ab}{gcd(ab,L)}\mathbb{Z}$, then a Gabor system on a lattice $\Lambda = A\mathbb{Z}_L^2$ becomes

$$G(g, \Lambda) = \left\{ \mathcal{M}_{sn+mb}\mathcal{T}_{na} g \;\middle|\; (n, m) \in \mathbb{Z}_N \times \mathbb{Z}_M \right\} \tag{2.24}$$

From this it can be deduced that a lattice with $s = 0$ is a separable or rectangular lattice as defined in (2.21) and that the Gabor system on such a lattice corresponds to the Gabor system defined in (2.16). For $s \neq 0$, the lattice is called nonseparable and is a subgroup of $\mathbb{Z}_L^2$.

The discrete Gabor transform associated with a general lattice as defined in (2.22) is the Gabor transform that corresponds with the Gabor system defined in (2.24). This discrete Gabor transform is defined as

$$(\mathcal{G}f)(m, n) = \sum_{l=0}^{L-1} f(l)\overline{g\,(l - na)} e^{-2\pi i l(m+w(n))/M}, \quad m \in \mathbb{Z}_M,\ n \in \mathbb{Z}_N \tag{2.25}$$

where $w(n) = \mathrm{mod}(ns, b)/b$.

In LTFAT the discrete Gabor transform on a general lattice can be computed through the routine `dgt` by adding the lattice type `lt` as an additional input argument. The general lattices can be constructed through the routine `matrix2latticetype`.

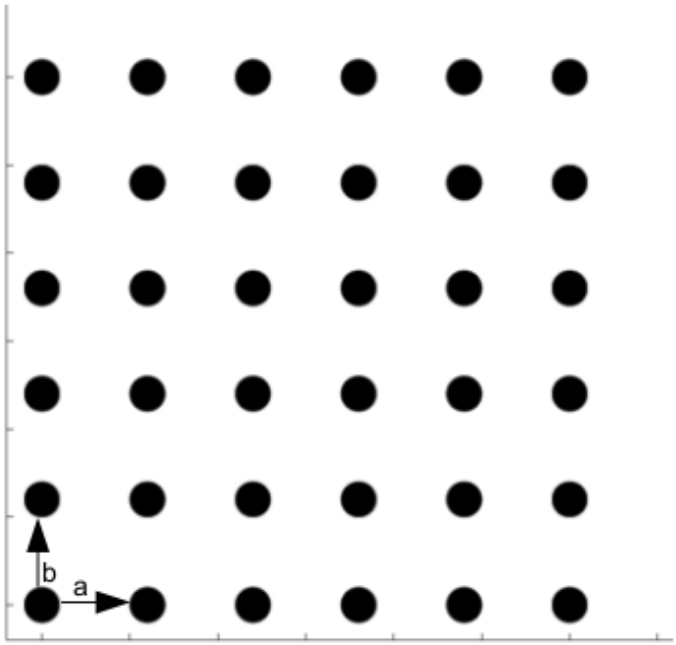

Figure 2.3: Seperable lattice

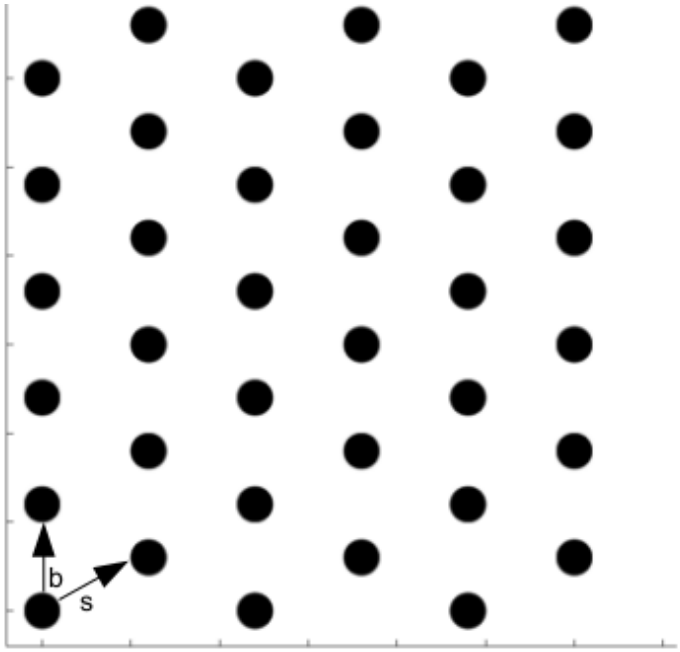

Figure 2.4: Nonseperable 'Quincux' lattice

2.4 Discrete Wilson transform

Since the redundancy of a Gabor transform as defined in (2.17) is $\frac{MN}{L}$, a Gabor transform has redundancy 1 when $MN = L$. In the case that $M = L$ and $N = 1$, the Gabor transform has redundancy 1, and $b = 1$ and $a = L$. In the case that $M = 1$ and $N = L$, the Gabor transform has also redundancy 1, and $b = L$ and $a = 1$. In the first case, the Gabor transform results in a $L \times 1$ matrix and in the second case it results in a $1 \times L$ matrix. In both cases the Gabor transform doesn't result in a valuable time-frequency description of a function $f \in \mathbb{C}^L$. An alternative to the Gabor transform for a time-frequency transform that results in a valuable time-frequency description with low redundancy is the Wilson transform.

The Wilson transform represents a function in terms of so-called Wilson functions. These Wilson functions are formed by a linear combination of elements of the Gabor system $\{\mathcal{M}_{mb}\mathcal{T}_{na}g\}$ where $g \in \mathbb{C}^L$ and $L = 2Nb = Na$.

The Wilson functions are explicitly given as

$$w(m,n) = \begin{cases} g(l-2na), & \text{for } m = 0 \\ \sqrt{2}g(l-2na)\sin(\pi ml/M), & \text{for } m < M,\ 2 \nmid m \\ \sqrt{2}g(l-2na)\cos(\pi ml/M), & \text{for } m < M,\ 2 \mid m \\ (-1)^l g(l-2na), & \text{for } m = M,\ 2 \mid m \\ (-1)^l g(l-2(n+1)a), & \text{for } m = M,\ 2 \nmid m \\ \sqrt{2}\cos(\pi ml/M)g(l-2(n+1)a), & \text{for } m = m+M,\ m < M,\ 2 \nmid m \\ \sqrt{2}\sin(\pi ml/M)g(l-2(n+1)a), & \text{for } m = m+M,\ m < M,\ 2 \mid m \end{cases}$$

where $m \in \mathbb{Z}_{2M}, n \in \mathbb{Z}_{\frac{L}{2M}}$. The Wilson functions expressed in terms of modulation and translation operators are given by

$$w(m,n) = \begin{cases} \mathcal{M}_0\mathcal{T}_{2na}g, & \text{for } m = 0 \\ \frac{-i}{\sqrt{2}}(\mathcal{M}_{mb}\mathcal{T}_{na}g - \mathcal{M}_{2M-m}\mathcal{T}_{na}g), & \text{for } m < M,\ 2 \nmid m \\ \frac{1}{\sqrt{2}}(\mathcal{M}_{mb}\mathcal{T}_{na}g + \mathcal{M}_{2M-m}\mathcal{T}_{na}g), & \text{for } m < M,\ 2 \mid m \\ \mathcal{M}_M\mathcal{T}_{2na}g, & \text{for } m = M,\ 2 \mid m \\ \mathcal{M}_M\mathcal{T}_{2(n+1)a}g, & \text{for } m = M,\ 2 \nmid m \\ \frac{1}{\sqrt{2}}(\mathcal{M}_{mb}\mathcal{T}_{na}g + \mathcal{M}_{2M-m}\mathcal{T}_{(n+1)a}g), & \text{for } m < M,\ 2 \mid m \\ \frac{-i}{\sqrt{2}}(\mathcal{M}_{mb}\mathcal{T}_{na}g - \mathcal{M}_{2M-m}\mathcal{T}_{(n+1)a}g), & \text{for } m = m+M,\ m < M,\ 2 \mid m \end{cases}$$

from which it can easily be deduced that the Wilson functions are constructed from translated and modulated window functions as the Gabor atoms are.

The discrete Wilson transform $\mathcal{W}$ of a function $f \in \mathbb{C}^L$ results in a matrix $c \in \mathbb{C}^{2M \times \frac{L}{2M}}$ of coefficients. These coefficients are computed as

$$(\mathcal{W}f)(m,n) = \langle f, w \rangle, \quad m \in \mathbb{Z}_{2M},\ n \in \mathbb{Z}_{\frac{L}{2M}}. \tag{2.26}$$

These Wilson coefficients are for all $m \in \mathbb{Z}_{2M},\ n \in \mathbb{Z}_{\frac{L}{2M}}$ given by

$$
\begin{aligned}
c(0,n) &= \sum_{l=0}^{L-1} f(l)g(l-2na), \quad m=0 \\
c(m,n) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin(\pi ml/M)g(l-2na), \quad m<M,\ 2\nmid m \\
c(m+M,n) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos(\pi ml/M)g(l-2(n+1)a), \quad m<M,\ 2\nmid m \\
c(m,n) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos(\pi ml/M)g(l-2na), \quad m<M,\ 2\mid m \\
c(m+M,n) &= \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin(\pi ml/M)g(l-(2n+1)a), \quad m<M,\ 2\mid m \\
c(M,n) &= \sum_{l=0}^{L-1} f(l)(-1)^l g(l-2na), \quad m=M, 2\mid M \\
c(M,n) &= \sum_{l=0}^{L-1} f(l)(-1)^l g(l-(2n+1)a), \quad m=M, 2\nmid M
\end{aligned}
$$

The inverse discrete Wilson transform $\mathcal{W}^{-1}$ constructs a function $f \in \mathbb{C}^L$ from the coefficients $c \in \mathbb{C}^{2M\times\frac{L}{2M}}$. The inverse discrete Wilson transform is given by

$$
(\mathcal{W}^{-1}c)(l) = \sum_{m=0}^{2M-1}\sum_{n=0}^{L/2M-1} c(m,n)w(m,n), \quad l \in \mathbb{Z}_L. \tag{2.27}
$$

where $w(m,n)$ are the Wilson functions constructed from a window function g.

The routine to compute the discrete Wilson transform in LTFAT is `dwilt`. This routine has a function `f`, window function `g` and the number of frequency shifts `M` as input parameters. To construct a window function that is suitable to use with a Wilson transform, the function `wilwin` can be used. The input parameters of `wilwin` are exactly the same as `dwilt`. The output of `dwilt` is a $2M \times N$ matrix with coefficients. These coefficients can be plotted through the routine `plotwilt`. The matrix of coefficients can be converted through a rectangular layout by the routine `wil2rect`. The inverse can be done through the routine `rect2wil`. The inverse discrete Wilson transform is called `idwilt` and has the $2M \times N$ matrix of coefficients and a window function as input parameters.

Chapter 3

Wavelet analysis

Wavelet analysis is based on the notion that general functions can be represented as a linear combination of translated and dilated or scaled wavelets. In this case a function is mapped from the time domain to a time-scale domain. A transform that maps a function from the time domain to the time-scale domain is in general called a time-scale transform. The inverse transform, that maps a set of coefficients from the time-scale domain to the time domain is called an inverse time-scale transform. Time-scale transforms that are associated with wavelets are called wavelet transforms.

3.1 Discrete scaling and translation

Discrete scaling The discrete scaling of a function is based on the convolution of that function with a so-called scaling function. In general two types of discrete scaling can be distinguished.

The first type of discrete scaling of a function $f \in \mathbb{C}^L$ consists of the convolution of $f(\frac{l}{N})$ with a scaling sequence ϕ. In this case is $f(\frac{l}{N})$ defined as

$$f(\frac{l}{N}) = \begin{cases} f(l/N), & \text{for } l/N \in \mathbb{Z}_L \\ 0, & \text{for } l/N \notin \mathbb{Z}_L \end{cases} \tag{3.1}$$

The discrete scaling operator associated with this type of scaling is given by

$$(\mathcal{U}_N f)(l) = \phi(l) \circledast f(\frac{l}{N}) = \sum_{n=0}^{L/N-1} f(n)\phi(l - Nn). \tag{3.2}$$

The second type of discrete scaling of a function $f \in \mathbb{C}^L$ consists of the convolution of f with a scaling function ϕ and is defined by

$$(\mathcal{D}_N f)(l) = (f \circledast \phi)(lN) = \sum_{n=0}^{LN-1} \phi(l) f(nN - l) \tag{3.3}$$

Discrete wavelet systems A discrete wavelet system consists in general of translated and discrete scaled versions of a wavelet function. A wavelet function is typically a function that consists of high frequencies only. The scaling operator associated with discrete scaled versions of a wavelet function is derived from the scaling operator defined in (3.2).

The discrete scaling operator $\mathcal{U}_{N^{j-1}}$ is defined as

$$(\mathcal{U}_{N^{j-1}}\psi)(l) = \phi_{j-1}(l) \circledast f(\frac{l}{N^{j-1}}) \tag{3.4}$$

$$= \sum_{k=0}^{\frac{L}{N^{j-1}}-1} f(k)\phi_{j-1}(l - N^{j-1}k) \tag{3.5}$$

where ϕ_{j-1} is a scaling sequence that is itself obtained through the discrete scaling operator associated with a scaling function ϕ, that is,

$$\phi_j(l) = (\mathcal{U}_{N^{j-1}}\phi)(l) = \sum_{k=0}^{\frac{L}{N^{j-1}}-1} \phi(k)\phi_{j-1}(l - N^{j-1}k) \tag{3.6}$$

where ϕ satisfies $\phi_0 = \delta$ and $\phi_1 = \phi$.

If the translation operator $\mathcal{T}_{na}$ is defined as in (2.1), that is,

$$(\mathcal{T}_{na}\psi)(l) = \psi(l - na) \tag{3.7}$$

then a translated and discrete scaled version of a wavelet function ψ is

$$(\mathcal{U}_{N^{j-1}}\mathcal{T}_{na}\psi)(l). \tag{3.8}$$

Such a version of wavelet function can be called a discrete wavelet atom. A collection of such discrete wavelet atoms, $\{\mathcal{U}_{N^{j-1}}\mathcal{T}_{na}\psi\}$, can be called a discrete wavelet system.

The discrete scaling operator with $N = 2$ scales a wavelet ψ dyadically. Such a discrete dyadically scaled wavelet ψ_j is given by

$$\psi_j(l) = (\mathcal{U}_{2^{j-1}}\psi)(l) = \sum_{k=0}^{\frac{L}{2^{j-1}}-1} \psi(k)\phi_{j-1}(l - 2^{j-1}k) \tag{3.9}$$

where its associated scaling sequence ϕ_j is given by

$$\phi_j(l) = (\mathcal{U}_{2^{j-1}}\phi)(l) = \sum_{k=0}^{\frac{L}{2^{j-1}}-1} \phi(k)\phi_{j-1}(l - 2^{j-1}k) \tag{3.10}$$

The system that is constructed from dyadically scaled and by $2n$ translated wavelet functions can be called a discrete dyadic wavelet system. This system is given in terms of discrete scaling and translation operators as $\{\mathcal{U}_{2^{j-1}}\mathcal{T}_{2n}\psi\}$.

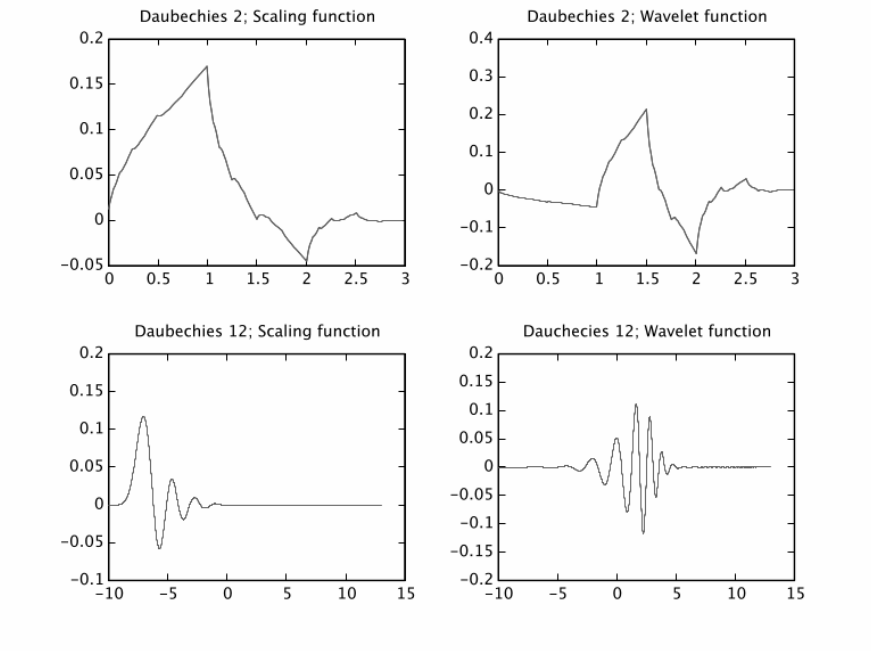

Figure 3.1: Daubechies scaling and wavelet functions

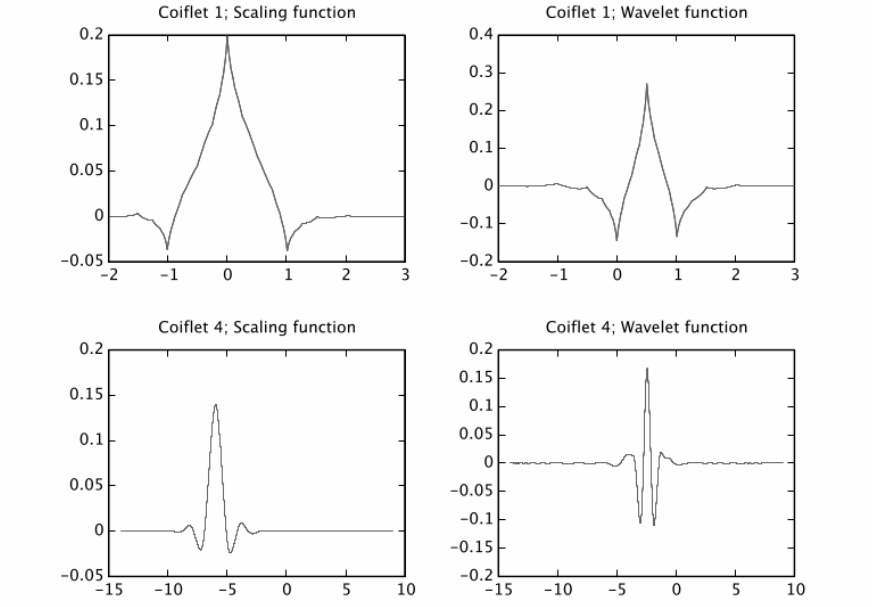

Figure 3.2: Coiflet scaling and wavelet functions

3.2 Discrete wavelet transform

The discrete wavelet transform decomposes a function dyadically into a collection of coefficients. The J-level discrete wavelet transform of a function $f \in \mathbb{C}^L$ is given by

$$\alpha_j(n) = \langle f, \mathcal{U}_{2^{j-1}} \mathcal{T}_{2n} \psi \rangle = \sum_{l=0}^{L-1} f(l) \overline{\psi_j(l - 2^j n)} \tag{3.11}$$

$$\beta_J(n) = \langle f, \mathcal{U}_{2^{J-1}} \mathcal{T}_{2n} \phi \rangle = \sum_{l=0}^{L-1} f(l) \overline{\phi_J(l - 2^J n)} \tag{3.12}$$

where $\alpha_j(n)$ with $n \in \mathbb{Z}_{2^{-j}L}$ and $j \in \{1, ..., J\}$ are called the wavelet coefficients and $\beta_J(n)$ the scaling coefficients.

A function $f \in \mathbb{C}^L$ can be constructed from the wavelet coefficients $\alpha_j(n)$ and scaling coefficients $\beta_J(n)$ trough the inverse discrete wavelet transform as

$$f(l) = \sum_{n=0}^{2^{-j}L} \beta_J(n) \tilde{\phi_J}(l - 2^J n) + \sum_{n=0}^{2^{-j}L} \sum_{j=1}^{J} \alpha_j(n) \tilde{\psi_j}(l - 2^j n) \tag{3.13}$$

where $\tilde{\phi_J}$ and $\tilde{\psi_j}$ denote the duals of ϕ_J and ψ_j, respectively. These satisfy the so-called biorthogonality relations given by

$$\sum_{l=0}^{L-1} \psi(l) \overline{\tilde{\psi}(2k - l)} = \delta_k \tag{3.14}$$

$$\sum_{l=0}^{L-1} \phi(l) \overline{\tilde{\phi}(2k - l)} = \delta_k \tag{3.15}$$

$$\sum_{l=0}^{L-1} \phi(l) \overline{\tilde{\psi}(2k - l)} = 0 \tag{3.16}$$

$$\sum_{l=0}^{L-1} \psi(l) \overline{\tilde{\phi}(2k - l)} = 0 \tag{3.17}$$

The construction of dual wavelet functions is discusses in chapter 4.

The discrete wavelet transform is implemented in the LTFAT through the so-called fast wavelet transform and is therefore called `fwt`. The input of `fwt` is a function `f`, wavelet function `w` and number of levels `J`. The wavelet functions are available in LTFAT as the routines starting with the prefix `wfilt_`. The output of `fwt` are the wavelet coefficients `c`. The inverse discrete wavelet transform is implemented as the routine `ifwt` and has as inputs the wavelet coefficients `c`, dual wavelet definition `dw`, number of levels `J` and length of the signal `L`.

Chapter 4

Frame theory

A frame of a finite-dimensional inner product space V is a collection of elements $\{f_k\}_{k=1}^N$ in V that spans V, that is, span$\{f_k\} = V$. If $\{f_k\}_{k=1}^N$ spans V and is also linear independent, it is called a basis for V. Both a frame and a basis allow that every $f \in V$ can be expressed as a linear combination of their elements. Since a basis is linear independent, the coefficients in the linear expansion of $f \in V$ are unique. The coefficients associated with the elements of a frame are non-unique when the frame is linear dependent, that is, when the frame is not a basis. A frame that is not a basis, is called overcomplete or redundant.

4.1 Frames

Any finite-dimensional inner product space is a finite-dimensional Hilbert space $\mathcal{H}$. A collection of elements $\{f_k\}_{k=1}^N \subseteq \mathcal{H}$ is called a frame for $\mathcal{H}$ if there exists finite frame bounds $A, B > 0$ such that for all $f \in \mathcal{H}$

$$A \left|\left|f\right|\right|^2 \;\leq\; \sum_{k=1}^{N} \left|\, \langle f, f_k \rangle \,\right|^2 \;\leq\; B \left|\left|f\right|\right|^2 \tag{4.1}$$

The statement in (4.1) is called the frame condition and it guarantees that any $f \in \mathcal{H}$ can be written as a linear combination of elements of the frame $\{f_k\}_{k=1}^N$.

A frame $\{f_k\}_{k=1}^N$ can in general be represented as a matrix $\mathbf{S}_N$ by

$$\mathbf{S}_N = \begin{pmatrix} | & | & ... & | \\ f_1 & f_2 & ... & f_N \\ | & | & ... & | \end{pmatrix}. \tag{4.2}$$

In this case are the elements f_i of the frame $\{f_k\}_{k=1}^N$ stored as column vectors of the matrix $\mathbf{S}_N$. Since N vectors can at most span a N-dimensional space, $\{f_k\}_{k=1}^N$ is a frame for a K-dimensional Hilbert space $\mathcal{H}$ when $N \geq K$. The redundancy of a finite frame $\{f_k\}_{k=1}^N$ for a K-dimensional Hilbert space $\mathcal{H}$ is therefore defined as the ratio $\frac{N}{K}$.

Discrete Fourier transform basis The collection of complex exponentials associated with a discrete Fourier transform given by

$$\{e^{2\pi ikl/L}\}_{k,l\in\mathbb{Z}_L}$$

forms a frame for a L-dimensional Hilbert space $\mathcal{H}$ if it satisfies the frame condition given in (4.1). If it forms a frame, then this collection of complex exponentials is called a discrete Fourier transform basis of $\mathcal{H}$.

Gabor frames A Gabor system is given by

$$\{\mathcal{M}_{mb}\mathcal{T}_{na}g\}$$

where $m \in \mathbb{Z}_M,\ n \in \mathbb{Z}_N,\ a, b > 0$ and $L = Mb = Na$. If this Gabor system associated with a window function $g \in \mathbb{C}^L$ satisfies the frame condition, (4.1), then it is called a Gabor frame for $\mathbb{C}^L$.

Wilson basis The Wilson functions are given by

$$w(m,n) = \begin{cases} g(l-2na), & \text{for } m = 0 \\ \sqrt{2}g(l-2na)\sin(\pi ml/M), & \text{for } m < M,\ 2 \nmid m \\ \sqrt{2}g(l-2na)\cos(\pi ml/M), & \text{for } m < M,\ 2 \mid m \\ (-1)^l g(l-2na), & \text{for } m = M,\ 2 \mid m \\ (-1)^l g(l-2(n+1)a), & \text{for } m = M,\ 2 \nmid m \\ \sqrt{2}\cos(\pi ml/M)g(l-2(n+1)a), & \text{for } m = m+M,\ m < M,\ 2 \nmid m \\ \sqrt{2}\sin(\pi ml/M)g(l-2(n+1)a), & \text{for } m = m+M,\ m < M,\ 2 \mid m \end{cases}$$

where $m \in \mathbb{Z}_{2M}, n \in \mathbb{Z}_{\frac{L}{2M}}$. If this collection of Wilson functions associated with a window function $g \in \mathbb{C}^L$ satisfies the frame condition, (4.1), then it is called a Wilson basis for $\mathbb{C}^L$.

Wavelet frames A discrete dyadic wavelet system is given by

$$\{\mathcal{U}_{N^{j-1}}\mathcal{T}_{2^n}\psi\}$$

where $j \in \{1, ..., J\}$ and $n \in \mathbb{Z}_{2^{-j}L}$. If this discrete dyadic wavelet system associated with a wavelet function $\psi \in \mathbb{C}^L$ satisfies the frame condition, (4.1), then it is called a discrete dyadic wavelet frame for $\mathbb{C}^L$.

A frame can be constructed in LTFAT through the routine called `frame`. The input parameter of `frame` is a string containing the frame type, `'frametype'`. Any additional input parameter depends on the frame type. The frame type of a general frame is called `gen` and has a matrix that contains the frame elements as additional parameter. The discrete Fourier transform basis, Gabor frame, Wilson basis and discrete dyadic wavelet frame are called `dft`, `dgt`, `dwilt` respectively `fwt`. These frames have the input parameters of their eponymous transforms as additional input parameters.

There are several LTFAT routines to obtain information about a constructed frame, including `framered` and `framebound`. The function `framered` calculates the redundancy of the frame `F`. If the output of `framered(F)` is higher than 1, `F` is a overcomplete or redundant frame, and if it is equal to 1, `F` is a basis. The function `framebound` calculates the frame bounds of the constructed frame `F`. When the function is assigned to one variable, e.g. `Q = framebound(F)`, the value assigned to this variable is the quotient B/A of the frame bounds A and B, and when the function is assigned to two variables, it assigns the value of the frame bounds A and B to the first and second variable, respectively.

4.2 Basic operations associated with frames

There are several basic operations associated with frames, including the analysis operator, synthesis operator and frame operator.

The analysis or coefficient operator $\mathcal{C} : \mathcal{H} \to \mathbb{C}^N$ associated with the frame $\{f_k\}_{k=1}^N$ maps a function f to its frame coefficients and is given by

$$\mathcal{C}f = \{\langle f, f_k \rangle\}_{k=1}^N \tag{4.3}$$

The synthesis or representation operator $\mathcal{R} : \mathbb{C}^N \to \mathcal{H}$ associated with the frame $\{f_k\}_{k=1}^N$ maps a collection of frame coefficients $\{c_k\}_{k=1}^N$ to a function r and is given by

$$\mathcal{D}c = \sum_{k=1}^{N} c_k f_k \tag{4.4}$$

The concatenation of the synthesis operator and analysis operator, $\mathcal{RC}$, is called the frame operator $\mathcal{S}$. The frame operator $\mathcal{S} : \mathcal{H} \to \mathcal{H}$ is given by

$$\mathcal{S}f = \sum_{k=1}^{N} \langle f, f_k \rangle f_k \tag{4.5}$$

Since the frame operator $\mathcal{S}$ is invertible, a function f can be perfectly reconstructed by

$$f = \sum_{k=1}^{N} \langle f, f_k \rangle f_k \mathcal{S}^{-1} = \sum_{k=1}^{N} \langle f, \mathcal{S}^{-1} f_k \rangle f_k \tag{4.6}$$

The frame $\{\tilde{f}_k\} = \{\mathcal{S}^{-1} f_k\}$ is called the dual frame of $\{f_k\}$ and guarantees that a function f can be perfectly reconstructed from its frame coefficients.

The concatenation of the analysis operator and synthesis operator, $\mathcal{CR}$, is called the Gramian operator $\mathcal{Q}$. The Gramian operator $\mathcal{Q} : \mathbb{C}^N \to \mathcal{C}^N$ is given by

$$\mathcal{Q}c = \sum_{i=1}^{N} c_i \{\langle f, f_k \rangle\}_{k=1}^N \tag{4.7}$$

In LTFAT the analysis operator is implemented as the function `frana`. The function `frana` computes the frame coefficients `c` associated with the frame `F` of the function `f` by the command `c = frana(F, f)`. To plot the frame coefficients `c` obtained through `frana` the function `plotframe` could be used. This function has the frame `F` and its associated frame coefficients `c` as input parameters. The synthesis operator is implemented as the routine `frsyn`. It has as input parameters a constructed frame `F` and a collection of frame coefficients `c`. To construct the dual frame `Fd` of the frame `F` the function `framedual` can be used. The routine `framedual` has a frame `F` as input parameters. The frame operator is implemented as the routine `frameoperator` and has a frame `F` and a function `f` as input parameters. The Gramian operator is implemented as the routine `frgramian` and has a collection of coefficients `c` and a frame `F` as input parameters.

Example 5. Listing 4.1 combines the functions `frame`, `framedual`, `frana` and `frsyn` to get a perfect reconstruction `r` of random vector `f` from its frame coefficients `c` obtained through the wavelet frame `F`.

```
L = 1000;
f = rand(L, 1);

w = 'db4';
J = 10;
F = frame('fwt', w, J);
c = frana(F,f);

Fd = framedual(F);
r = frsyn(Fd,c);

norm(f - r)
```

Listing 4.1: framedual_reconstruction.m

The output of listing 4.1 is given in listing 4.2.

```
ans = 2.2412e-13
```

Listing 4.2: Output of listing 4.1

which shows that there is no remarkable difference between the reconstructed function `r` and the original function `f`. ◊

Appendix A

Installation

A.1 System requirements

The currently supported platforms for the LTFAT are Linux, Windows, and Mac OS X. The toolbox should work with any version of Matlab from Matlab2009b and any version of Octave from Octave 3.6.

A.2 Download

The LTFAT can be directly downloaded from the download section of the LTFAT homepage or from `http://sourceforge.net/projects/ltfat/files/`. To install the toolbox, the downloaded file should be unpacked which results in a directory called `ltfat`. The toolbox is contained in this directory and in all its subdirectories. To start the toolbox the `ltfat` directory should be in the current folder of Matlab/Octave or a path to the `ltfat` directory should be set using the Matlab/Octave command `addpath`. In Matlab the path to the `ltfat` directory can be set in the `startup.m` file and in Octave the path can be set in the `~/.octaverc` file. If the `ltfat` directory is in the current folder or a path to the directory is set, the toolbox can be started by executing the command `ltfatstart` in the prompt. This command will setup all the necessary paths and perform the necessary initializations to use the toolbox successfully.