The LTFAT tutorial

Jordy van Velthoven, Peter L. Søndergaard

October 8, 2015

Contents

Introduction

The Large Time-Frequency Analysis Toolbox (LTFAT) is a well-documented 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, Gabor and wavelet analysis along with associated operators and plotting routines. The routines that are included in the toolbox are primarily programs written in Matlab/Octave, but the toolbox contains also MEX/OCT interfaces written in C/C++, which function as a backend library.

This tutorial provides an introduction to LTFAT by giving a summary of the basic methods used in the toolbox. Each chapter and section of the tutorial starts with the theory and mathematical background of the routines. This theoretical treatment is written in an informal style and the interested reader is referred to several references for proofs and further details. After the theoretical treatment of the methods, it will be specified how these methods can be used in LTFAT. Note that not every function that is included in LTFAT is discussed in the tutorial. For a complete overview of the toolbox, the interested reader may consult the on-line documentation[1] and the LTFAT reference manual[2], which both contain a complete overview of the toolbox.

Terminology and notation All functions and operators in the tutorial are finite-dimensional. Functions are considered as elements of the L-dimensional complex space $\mathbb{C}^L$. Any element $f \in \mathbb{C}^L$ is represented as a function on the finite Abelian group $\mathbb{Z}_L$, i.e., $f : \mathbb{Z}_L \to \mathbb{C}$, where $\mathbb{Z}_L := \{0, 1, ..., L-1\}$ denotes the commutative ring of integers modulo L. In this setting, all operations on the indices are computed modulo L. Furthermore, the indices have a cyclic indexing, i.e., $f(l + kL) = f(l)$ for all $l, k \in \mathbb{Z}$. The value of the l'th element of f is denoted by $f(l)$, where $l \in \mathbb{Z}_L$. A linear operator $\mathcal{O} : \mathbb{C}^L \to \mathbb{C}^M$ is represented as a matrix multiplication $(\mathcal{O}f)(m) = \sum_{l=0}^{L-1} o(m, l)f(l)$, 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 formulae in this tutorial are indexed starting from 0. The formulae in this tutorial can therefore be translated into procedures for Matlab and Octave by adding +1 to the argument when indexing a data structure.

In order to distinguish the mathematical background from 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.

[1] `http://ltfat.sourceforge.net/doc/start`
[2] `http://ltfat.sourceforge.net/doc/ltfat.pdf`

Chapter 1

Fourier analysis

Fourier analysis is based on the notion that arbitrary functions can be represented in terms of complex exponentials. The coefficients associated with these complex coefficients can be obtained through the *Fourier transform*, which maps a function to the so-called *frequency domain*. The inverse of the Fourier transform, the *inverse Fourier transform*, maps a function from the frequency domain back to the *time domain*.

1.1 Periodic functions

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

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

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

$$f(l) = f(-l),$$

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

$$-f(l) = f(-l).$$

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

$$f(l) = f(N - l).$$

In this case, the function is said to be symmetric around N. If a function is symmetric around an even N, it is called *whole-point symmetric*. If a function is symmetric around an odd N, then it is called *half-point symmetric*. In general, a function is half-point symmetric if it is symmetric around $l = -\frac{1}{2}$, and whole-point symmetric if it is symmetric around $l = 0$. Using these definitions, a function $f \in \mathbb{C}^L$ is said to have *half-point even symmetry* if, for all $l \in \mathbb{Z}_L$,

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

Similarly, a function $f \in \mathbb{C}^L$ is said to have *whole-point even symmetry* if, for all $l \in \mathbb{Z}_L$,

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

In the same manner, a function $f \in \mathbb{C}^L$ is called *half-point odd symmetric* if for all $l \in \mathbb{Z}_L$,

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

and is called *whole-point odd symmetric* if for all $l \in \mathbb{Z}_L$,

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

Routines for periodic functions There are several routines in LTFAT that can be used in order to get the even or odd part of a periodic function. For the even part, the routine `peven` can be used. This routine returns the even part of an input array `f` as an output array `fe`, i.e., `fe = peven(f)`. In a similar way, the routine `podd` returns the odd part of an input array as an output array. To check whether an array is even, the LTFAT routine called `isevenfunction` can be used. The input of this routine is an array `f`, the output is the scalar `1` if the array `f` is whole-point even and the scalar `0` if the array `f` is not whole-point even. The routine `isevenfunction` does the same for a half-point even input array 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 the routine `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 respectively half-point even symmetry.

Examples of periodic functions LTFAT contains several periodic functions. Examples of periodic functions that are included are the complex exponential, chirp, rectangular function and sinc function. All these functions are characterized by their properties in relation to the Fourier transform.

A discrete complex exponential h of length L with discrete frequency k is given by

$$h(l) = e^{2\pi i l k/L}, \quad l \in \mathbb{Z}_L. \tag{1.1}$$

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 — see §1.2. The routine that constructs a complex exponential is called `expwave`. This routine has the length and the frequency of the complex exponential as input parameters.

A periodic, discrete chirp g of length L is given by

$$g(l) = e^{\pi i n (l-\lceil L/2\rceil^2(L+1)/L}, \quad l \in \mathbb{Z}_L. \tag{1.2}$$

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

The discrete, periodic rectangle function r of length L is given by

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

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

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

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.

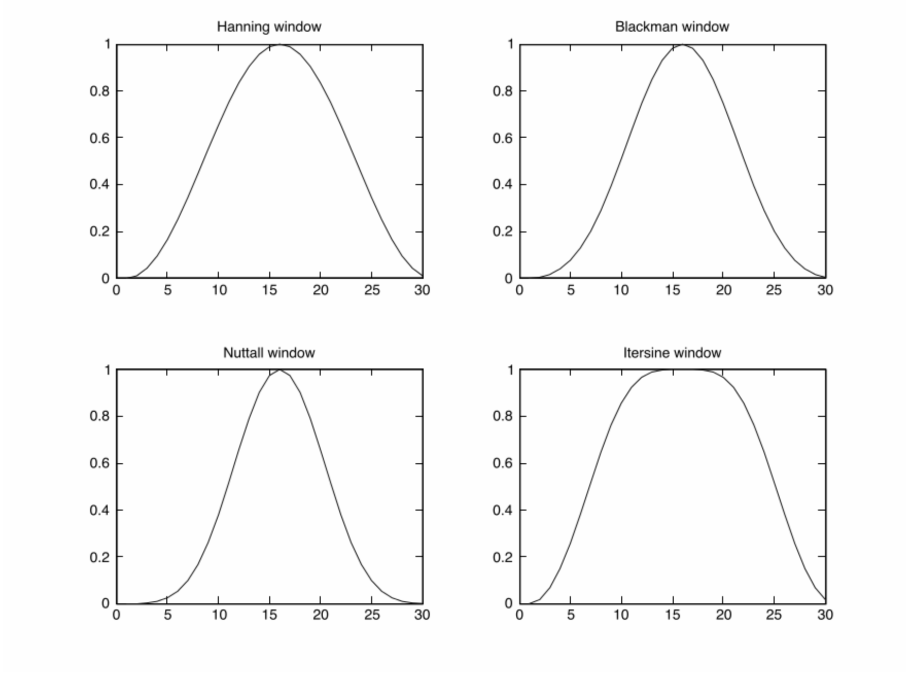

Figure 1.1: FIR window functions

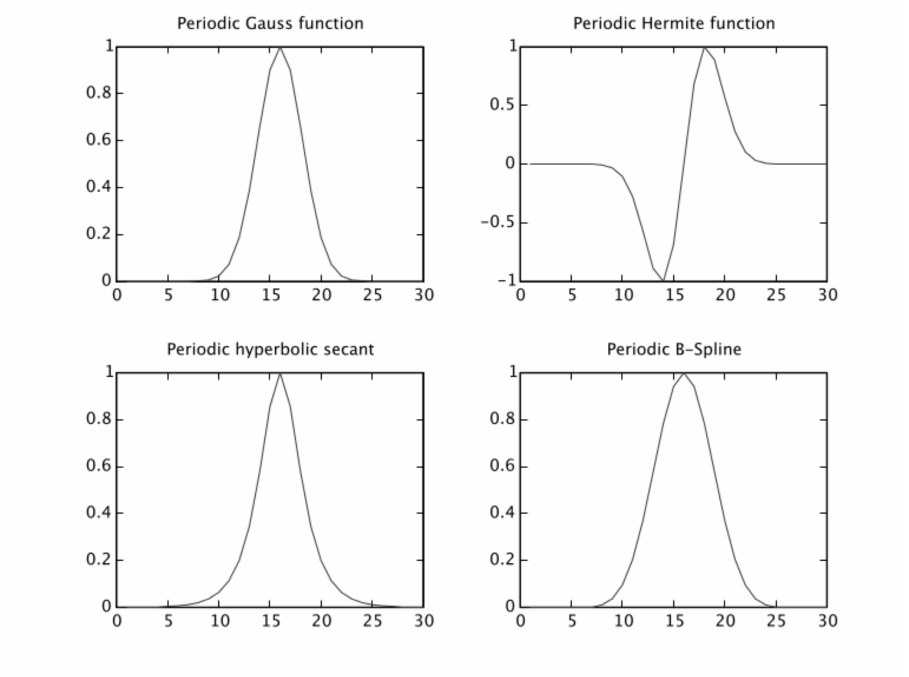

Figure 1.2: IIR window functions

Window functions A window function is a function that is in general centred around the origin and zero-valued outside a certain interval. In LTFAT both so-called Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) windows functions are implemented. In the finite, discrete setting, an FIR window is defined as a window function of which the length of its support is shorter than its total length, and an IIR window is a window function of which the length of its support equals its total length L. In LTFAT, 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

$$\varphi(l) = \left(\frac{L}{2}\right)^{-1/4} \sum_{k=0}^{L-1} e^{-\pi(\frac{l}{\sqrt{L}} - k\sqrt{L})^2}, \quad l \in \mathbb{Z}_L \tag{1.3}$$

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`.

1.2 Discrete Fourier transforms

The Fourier transform on $\mathbb{C}^L$ is in general called the *discrete Fourier transform* or *finite Fourier transform*. The unitary discrete Fourier transform $\mathcal{F} : \mathbb{C}^L \to \mathbb{C}^L$ is the operator given by

$$(\mathcal{F}f)(k) = \frac{1}{\sqrt{L}} \sum_{l=0}^{L-1} f(l)\, e^{-2\pi ikl/L}, \quad k \in \mathbb{Z}_L. \tag{1.4}$$

The operator given in (1.4) is a normalized version of the discrete Fourier transform in the sense that it corresponds to the discrete Fourier transform up to $1/\sqrt{L}$. This normalization yields that the discrete Fourier transform is a unitary transform on $\mathbb{C}^L$, i.e.,

$$\langle f, g\rangle_{\mathbb{C}^L} = \langle \mathcal{F}f, \mathcal{F}g\rangle_{\mathbb{C}^L}, \tag{1.5}$$

for any $f, g \in \mathbb{C}^L$. The equality given in (1.5) is in general called *Parseval's identity* of the discrete Fourier transform. By setting $f = g$, *Plancherel's identity* follows as a special case,

$$\|f\|_{\mathbb{C}^L} = \|\mathcal{F}f\|_{\mathbb{C}^L}.$$

The discrete Fourier transform can be considered as a unitary transform, but it can also be considered as a matrix product. In this case, the unitary discrete Fourier transform of a function $f \in \mathbb{C}^L$ is given by

$$\mathcal{F}f = \mathbf{F}_L f,$$

where $\mathbf{F}_L$ is a $L \times L$ matrix, the so-called *Fourier matrix*, and is given by

$$\mathbf{F}_L = \frac{1}{\sqrt{L}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{(L-1)} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(L-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{(L-1)} & \omega^{2(L-1)} & \cdots & \omega^{(L-1)(L-1)} \end{pmatrix}, \tag{1.6}$$

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

Any function can be reconstructed from its Fourier transform through the so-called *inverse Fourier transform.* The unitary inverse discrete Fourier transform $\mathcal{F}^{-1} : \mathbb{C}^L \to \mathbb{C}^L$ is the operator given by

$$(\mathcal{F}^{-1}c)(l) = \frac{1}{\sqrt{L}} \sum_{k=0}^{L-1} c\,(k)\, e^{2\pi ikl/L}, \tag{1.7}$$

where $l \in \mathbb{Z}_L$. As the unitary discrete Fourier transform is a normalized version of the discrete Fourier transform, the unitary inverse discrete Fourier transform is a normalized version of the inverse discrete Fourier transform. The unitary inverse discrete Fourier transform corresponds to the inverse discrete Fourier transform up to $\sqrt{L}$. The unitary discrete Fourier transform possess aside its definition in terms of an operator also a definition in terms of a matrix. In terms of a matrix, the inverse unitary discrete Fourier transform $\mathcal{F}^{-1}$ is the $L \times L$ matrix given by

$$\mathcal{F}^{-1}c = \mathbf{F}_L^{-1}c = \overline{\mathbf{F}^T}_L c$$

where $\overline{\mathbf{F}^T}_L$ is the conjugate transpose of the Fourier matrix $\mathbf{F}_L$, which was defined in (1.6).

The fact that the Fourier transform and its inverse, as defined in (1.4) respectively (1.7), are both unitary, leads, for all $f \in \mathbb{C}^L$, to the following reproducing formula

$$f(l) = \sum_{k=0}^{L-1} (\mathcal{F}f)(k) e^{2\pi ikl/L}, \tag{1.8}$$

where $l \in \mathbb{Z}_L$. Similarly, for all $f \in \mathbb{C}^L$,

$$f = \overline{\mathbf{F}^T}_L \mathbf{F}_L f. \tag{1.9}$$

The discrete fractional Fourier transform is a generalization of the discrete Fourier transform. The discrete fractional Fourier transform can be considered as a discrete Fourier transform to the power n, i.e., $\mathcal{F}^n$. For $n = 1, 2, 3, 4, ...$,

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

The ordinary discrete Fourier transform, $\mathcal{F}^1$, maps a function from the so-called time-domain to the so-called frequency domain, which corresponds to 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 — see figure 2.2. Therefore, the discrete fractional Fourier transform allows representations of a function between the time and the frequency domain.

Routines for discrete Fourier transforms The unitary discrete Fourier transform is implemented in LTFAT as the routine `dft`. The input of the routine `dft` is an array and so is its output. If the input array consists of real-valued scalars only, then the routine `fftreal` could, instead of the routine `dft`, be used. The routine `fftreal` differs from `dft` in that the output of the routine `dft` consists of both positive and negative frequencies, whereas the output of `fftreal` consists of positive frequencies only. In order to plot the output array of `dft` and `fftreal`, the routines called `plotfft` and `plotfftreal` can be used, respectively. The unitary discrete Fourier transform is implemented by the routine `idft`. The input of the routine `idft` is a vector and so is its output. 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 array `f`.

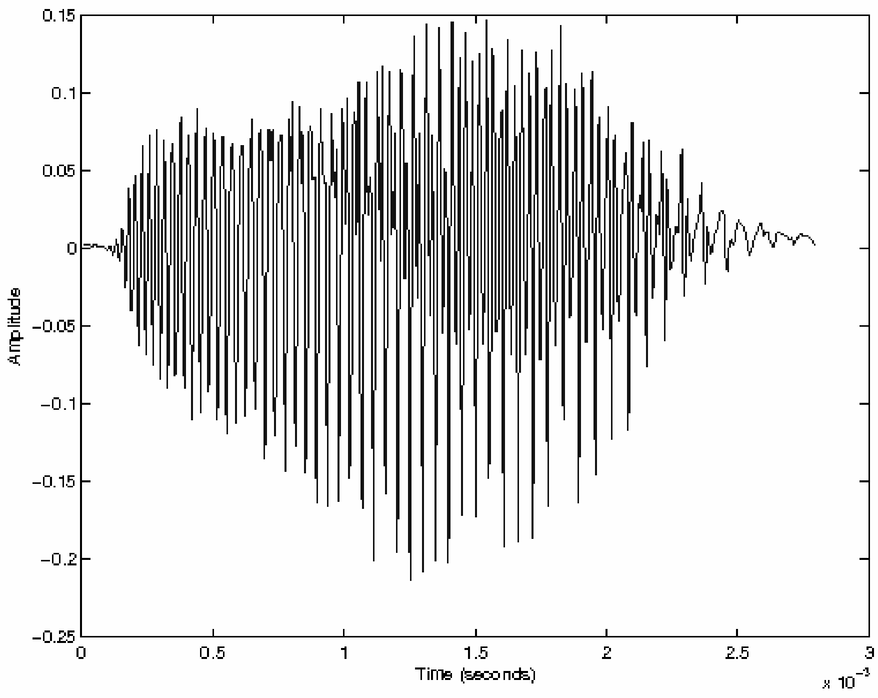

Figure 1.3: Bat signal in time domain

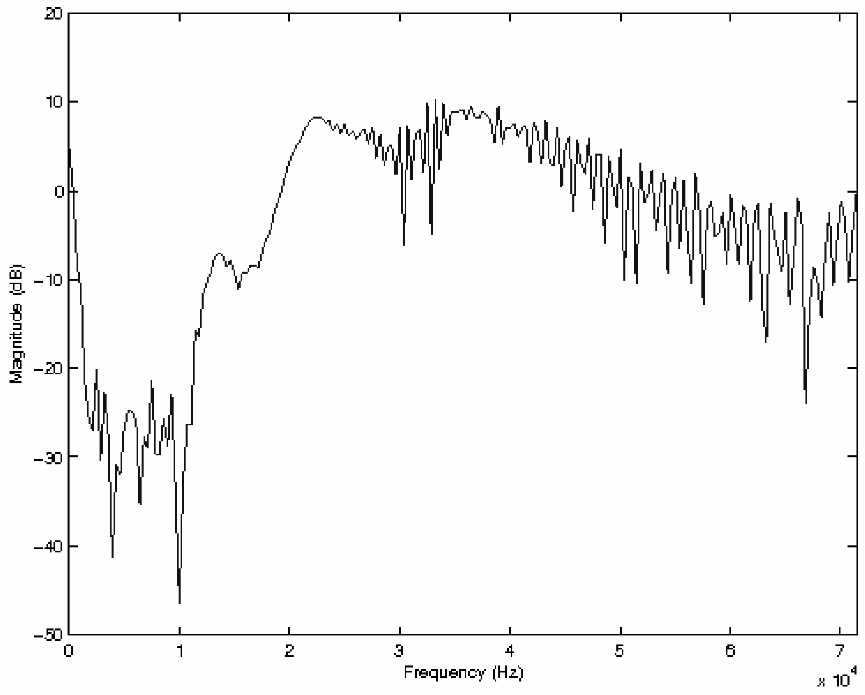

Figure 1.4: Bat signal in frequency domain

Example 1. In this example, the reproducing formulas of the discrete Fourier transform, as given in (1.8) and (1.9), will be verified. In listing 1.1, the discrete Fourier transform of the array `f` of length 400 is computed, which is a 400×400 array called `c`. Then the inverse discrete Fourier transform of `c` is computed, which is an array of length 400 called `r`. Finally, the difference between `f` and `r` is computed.

```
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 array `f` and its reconstruction `r`. □

1.3 Operations for periodic functions

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

$$(f^{\diamond})^{\diamond} = f. \tag{1.10}$$

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

$$f^{\diamond}(l) \;=\; \overline{f(-l)}, \tag{1.11}$$

for all $l \in \mathbb{Z}_L$. That $f^{\diamond}$ is an involution is easily verified,

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

The involution (1.11) is an important operator in Fourier analysis since it possess several important relations with respect to the Fourier transform. The involution (1.11) is related to the discrete Fourier transform $\mathcal{F}$ by

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

The inverse discrete Fourier transform $\mathcal{F}^{-1}$ of a function $c \in \mathbb{C}^L$ can therefore be expressed in terms of involution as

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

The routine for the involution (1.11) is implemented in the LTFAT as `involute`. The routine `involute` takes an array as input and returns an array as output.

Example 2. In listing 1.3 the relations between involution, conjugation and the discrete Fourier transform are 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 is given in listing 1.4, which confirms the relations between involution, conjugation and the discrete Fourier transform. □

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

$$h(l) = (f \circledast g)(l) = \sum_{n=0}^{L-1} f(n) g(l-n), \tag{1.12}$$

for all $l \in \mathbb{Z}_L$. The periodic convolution $h \in \mathbb{C}^L$ of two functions $f, g \in \mathbb{C}^L$ is related to the discrete Fourier transforms of f and g by

$$\begin{aligned}(\mathcal{F}(f \circledast g))(k) &= (\mathcal{F}f \cdot \mathcal{F}g)(k),\\ (\mathcal{F}(f \cdot g))(k) &= (\mathcal{F}f \circledast \mathcal{F}g)(k).\end{aligned}$$

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 by the inverse discrete Fourier transform of the quotient

$$\frac{(\mathcal{F}h)(k)}{(\mathcal{F}g)(k)}. \tag{1.13}$$

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

$$x(l) = (f \circledast g^\diamond)(l) = \sum_{n=0}^{L-1} f(n) \overline{g(n-l)}, \tag{1.14}$$

and the periodic cross-correlation of a function f with itself is called the *periodic autocorrelation* of f,

$$a(l) = (f \circledast f^\diamond)(l) = \sum_{n=0}^{L-1} f(n) \overline{f(n-l)}. \tag{1.15}$$

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

$$(\mathcal{F}(f \circledast g^\diamond))(k) = (\mathcal{F}f \cdot \overline{\mathcal{F}g})(k), \tag{1.16}$$

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

$$(\mathcal{F}(f \circledast f^\diamond))(k) = (\mathcal{F}f \cdot \overline{\mathcal{F}f})(k) = |\mathcal{F}f(k)|^2. \tag{1.17}$$

The periodic convolution and periodic cross-correlation are implemented in LTFAT by the routines `pconv` and `pxcorr`, respectively. For both routines the inputs are one-dimensional arrays.

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

$$h(l) = (f * g)(l) = \sum_{n \in \mathbb{Z}} f(n) g(l-n), \quad l \in \mathbb{Z}. \tag{1.18}$$

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 (1.12) 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

$$L_h \geq L_f + L_g - 1, \tag{1.19}$$

where L_f and L_g is the lengths of the function f and g, respectively. 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. ♢

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* $\mathcal{T}_n$, which translates a function $f \in \mathbb{C}^L$ by n, is given by

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

The *modulation operator* $\mathcal{M}_k$, which modulates a function $f \in \mathbb{C}^L$ by k, is given by

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

The translation operator $\mathcal{T}_n$ and modulation operator $\mathcal{M}_k$ satisfy the following commutation relations

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

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

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

and

$$\mathcal{F}(\mathcal{M}_k f) = \mathcal{T}_k(\mathcal{F} f),$$

respectively. 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, the modulation operator is also called the frequency-shift operator. In the same vein, the translation operator $\mathcal{T}_n$ is sometimes 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

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

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

$$(\mathcal{M}_k \mathcal{T}_n g)(k, n) = g(l-n)e^{i2\pi kl/L}. \tag{2.2}$$

If the function $g \in \mathbb{C}^L$ in (2.2) is a window function, then a collection of such time-frequency shifted window functions is given by

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

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 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

$$(\mathcal{V}_g f)(k,n) = \langle f, \mathcal{M}_k \mathcal{T}_n g \rangle, \quad k,n \in \mathbb{Z}_L \tag{2.3}$$

$$= \sum_{l=0}^{L-1} f(l)\overline{g\,(l-n)}e^{-2\pi ikl/L}, \quad k,n \in \mathbb{Z}_L \tag{2.4}$$

where g is a window function. The discrete short-time Fourier transform maps a function $f \in \mathbb{C}^L$ to a matrix $\mathcal{V}_g f \in \mathbb{C}^{L \times L}$ of coefficients. In this matrix k denotes the frequency around which the window function g of (2.3) is positioned and n denotes the 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 windowed function $f(l)g(l-n)$.

The discrete short-time Fourier transform satisfies Moyal's formula, which is given by

$$\sum_{k=0}^{L-1}\sum_{n=0}^{L-1}(\mathcal{V}_{g_1} f_1)(k,n)\overline{(\mathcal{V}_{g_2} f_2)(k,n)} = \Bigg(\sum_{l=0}^{L-1} f_1(l)\overline{f_2(l)}\Bigg)\Bigg(\sum_{l=0}^{L-1} g_1(l)\overline{g_2(l)}\Bigg), \tag{2.5}$$

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 (2.5), the following equality is obtained as a consequence of Moyal's formula

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

This formula gives rise to the inversion formula of the discrete short-time Fourier transform. The inversion formula of the discrete short-time Fourier transform shows that any $f \in \mathbb{C}^L$ can be expressed in termes of the short-time Fourier transform,

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

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. In general, the inverse discrete Fourier transform of a $c \in \mathbb{C}^{L \times L}$ is given by

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

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.

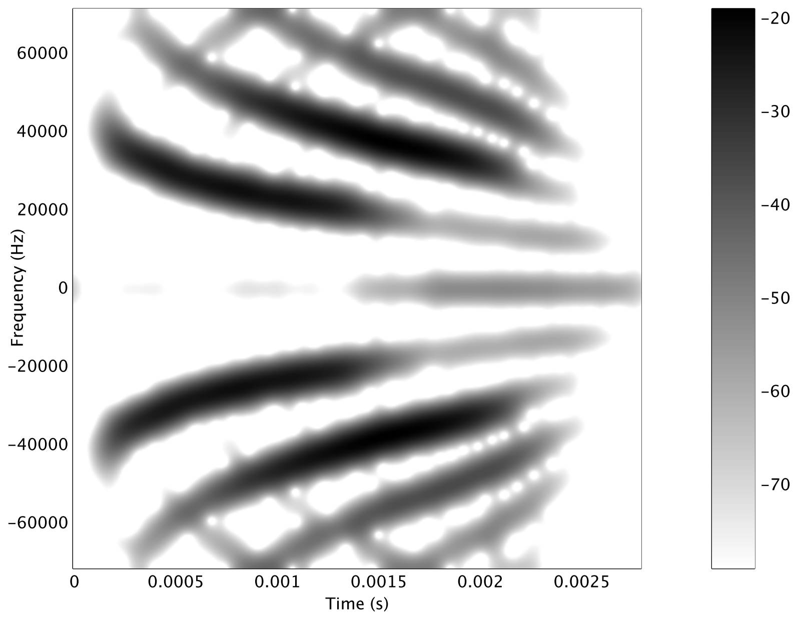

Figure 2.1: Spectrogram of bat signal

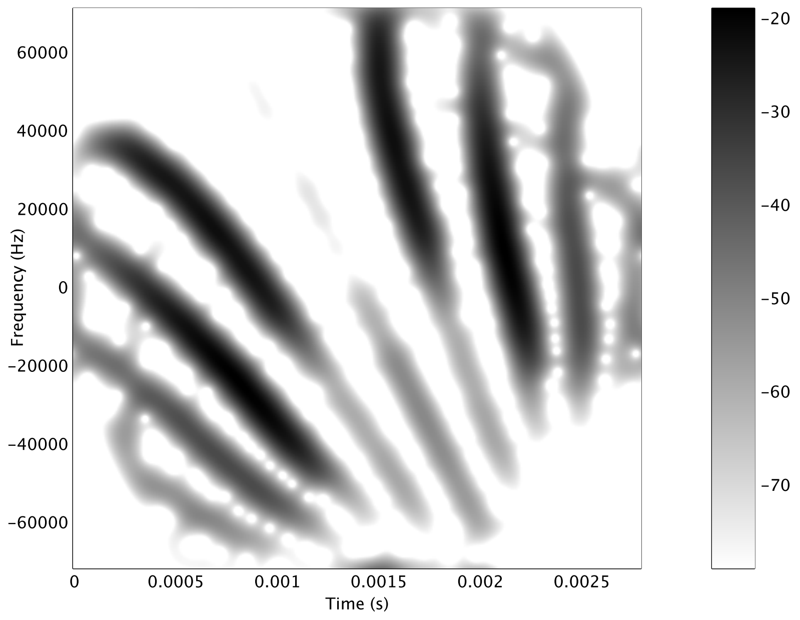

Figure 2.2: Spectrogram of a fractional Fourier transform of bat signal.

2.3 Discrete Gabor transform

The discrete short-time Fourier transform as defined in (2.3) has a redundancy L since it represents a function $f \in \mathbb{C}^L$ by 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 version of a window function is given by

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

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

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

is called a *Gabor system.* The discrete Gabor transform of a function is formally defined as the inner products between this function and the elements of a Gabor system. The Gabor transform $\mathcal{G}$ of a function $f \in \mathbb{C}^L$ is therefore explicitly given by

$$\begin{aligned}(\mathcal{G}f)(m,n) &= \langle f, \mathcal{M}_{mb}\mathcal{T}_{na}g\rangle_{\mathbb{C}^L}, &(2.6)\\ &= \sum_{l=0}^{L-1} f(l)\overline{g\,(l-na)}e^{-2\pi imbl/M}. &(2.7)\end{aligned}$$

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 quotient $\frac{MN}{L}$. From this it can be deduced that the short-time Fourier transform, as defined in (2.3), has indeed a redundancy L. Although a Gabor transform may have a lower redundancy than a short-time Fourier transform, it is still possible reconstruct an analysed function from its Gabor transform. In order to do so, the inverse Gabor transform should be used. The inverse Gabor transform $\mathcal{G}^{-1}$ of $c \in \mathbb{C}^{M\times N}$ is given by

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

Whenever the window function γ is a *dual* of g, the window function used in the analysis, then $\mathcal{G}^{-1}c$ equals the analysed function f. Thus, a perfect reconstruction of a function from its Gabor transform is possible if, and only if, the analysis and synthesis are performed with respect to two dual window functions. There are several conditions that characterize when two Gabor window functions are dual window functions, including the Wexler-Raz relations and the characterization equation. By the Wexler-Raz relation, two Gabor window functions $g, \gamma \in \mathbb{C}^L$ are dual if, and only if,

$$\frac{MN}{L}\langle g, \mathcal{M}_{mN}\mathcal{T}_{nM}\gamma\rangle = \delta_m\delta_n,$$

for all $m \in \mathbb{Z}_a,\ n \in \mathbb{Z}_b$. The characterization equation states that two Gabor window functions $g, \gamma \in \mathbb{C}^L$ are dual window functions if, and only if,

$$\sum_{k=0}^{L-1}\overline{g(l-nM-ka)}\gamma(x-ka) = \frac{\delta_{n,0}}{M},$$

for all $l \in \mathbb{Z}_a, n \in \mathbb{Z}_b$.

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. $\Diamond$

2.3.1 Lattices associated with Gabor transforms

The discrete Gabor transform defined in (2.6) 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

$$\Lambda = \left\{ (an, bm) \;\middle|\; n \in \mathbb{Z}_N,\ m \in \mathbb{Z}_M \right\}. \tag{2.8}$$

Note that the lattice Λ is a subset of $\mathbb{Z}_L^2$. Here $\mathbb{Z}_L^2$ is the lattice that corresponds with the short-time Fourier transform as defined in (2.3). A lattice as defined in (2.8) is in general called *separable* or *regular*, and it can be written as

$$\Lambda = A(\mathbb{Z}_L \times \mathbb{Z}_L), \tag{2.9}$$

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

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

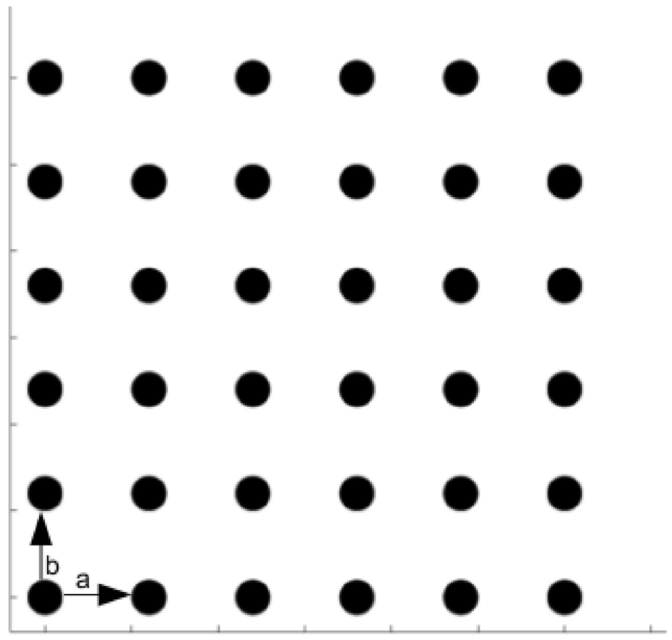

Figure 2.3: Seperable lattice

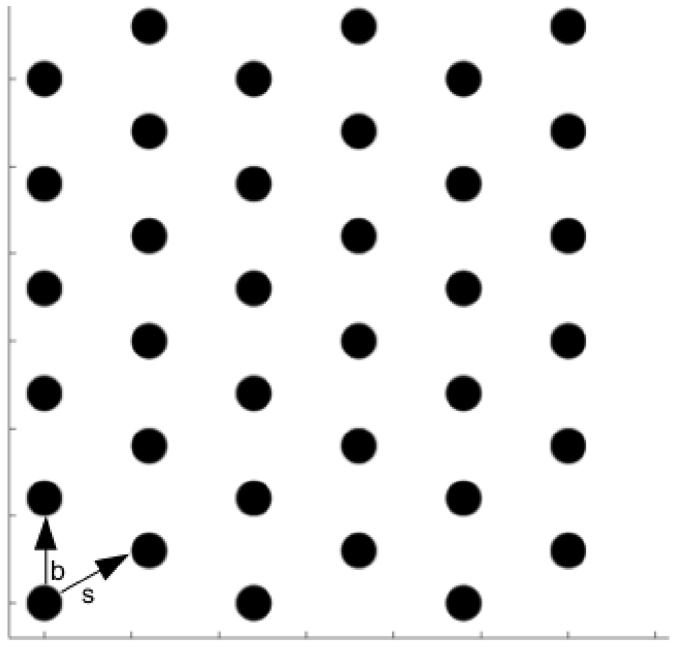

Figure 2.4: Nonseperable 'Quincux' lattice

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

$$G(g, \Lambda) = \left\{ \mathcal{M}_{sn+mb}\mathcal{T}_{na}g \;\middle|\; (n, m) \in \mathbb{Z}_N \times \mathbb{Z}_M \right\}. \tag{2.10}$$

From this it can be deduced that a lattice with $s = 0$ is a separable or rectangular lattice as defined in (2.8), and that the Gabor system on such a lattice corresponds to the Gabor system defined in (2.3). 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 (2.9) is the Gabor transform that corresponds with the Gabor system defined in (2.10). This discrete Gabor transform is defined as

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

where $m \in \mathbb{Z}_M,\ n \in \mathbb{Z}_N$ and $w(n) := \mathrm{mod}(ns, b)/b$.

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`.

2.4 Discrete Wilson transform

Since the redundancy of a Gabor transform as defined in (2.6) 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 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

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

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

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

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

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

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

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, the function $f(\frac{l}{N})$ is explicitly defined as

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

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

$$(\mathcal{U}_N f)(l) = \phi(l) \circledast f(\frac{l}{N}) = \sum_{n=0}^{L/N-1} f(n)\phi(l - Nn). \tag{3.1}$$

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

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

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 (3.1).

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

$$\begin{aligned} (\mathcal{U}_{N^{j-1}}\psi)(l) &= \phi_{j-1}(l) \circledast f(\frac{l}{N^{j-1}}) \\ &= \sum_{k=0}^{\frac{L}{N^{j-1}}-1} f(k)\phi_{j-1}(l - N^{j-1}k), \end{aligned}$$

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

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

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,

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

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

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

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

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

where its associated scaling sequence ϕ_j is given by

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

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

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

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

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

$$f(l) = \sum_{n=0}^{2^{-j}L} \beta_J(n)\tilde{\phi}_J(l - 2^J n) + \sum_{n=0}^{2^{-j}L}\sum_{j=1}^{J} \alpha_j(n)\tilde{\psi}_j(l - 2^j n)$$

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

$$\sum_{l=0}^{L-1} \psi(l)\overline{\tilde{\psi}(2k-l)} = \delta_k, \tag{3.2}$$

$$\sum_{l=0}^{L-1} \phi(l)\overline{\tilde{\phi}(2k-l)} = \delta_k, \tag{3.3}$$

$$\sum_{l=0}^{L-1} \phi(l)\overline{\tilde{\psi}(2k-l)} = 0, \tag{3.4}$$

$$\sum_{l=0}^{L-1} \psi(l)\overline{\tilde{\phi}(2k-l)} = 0. \tag{3.5}$$

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`.

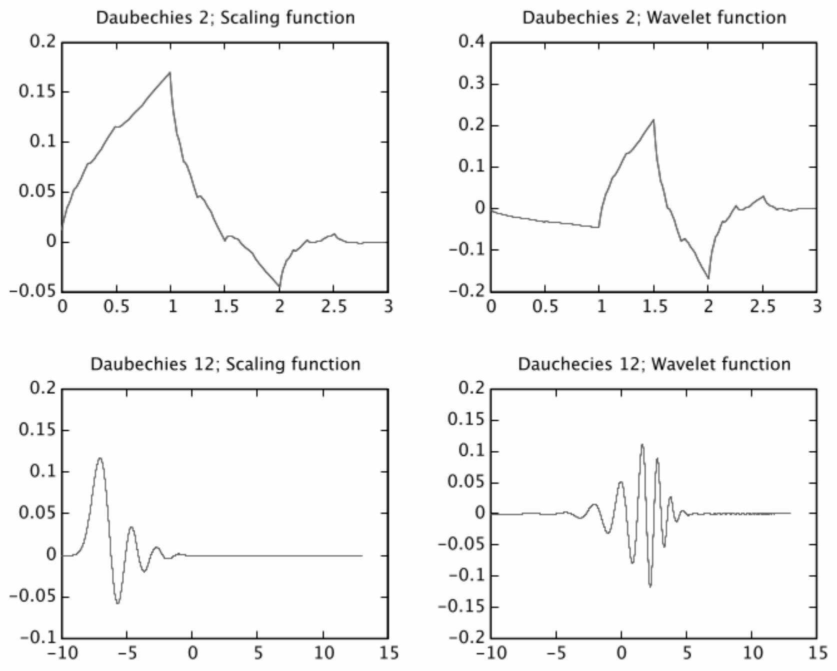

Figure 3.1: Daubechies scaling and wavelet functions

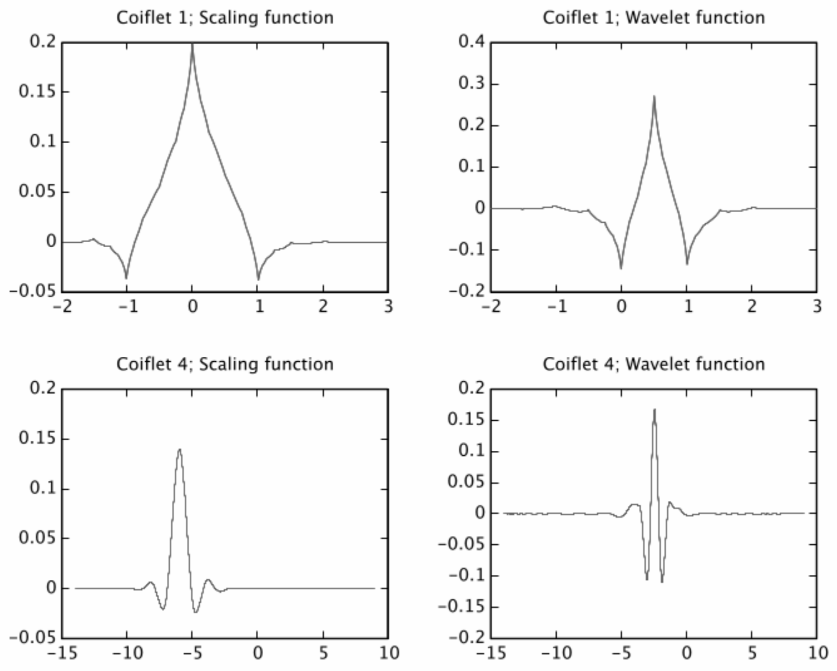

Figure 3.2: Coiflet scaling and wavelet functions

Chapter 4

Frame theory

A *frame* for a finite-dimensional inner product space V is a collection of elements $\{f_k\}_{k=1}^N$ in V that spans V, that is, $\text{span}\{f_k\} = V$. If $\{f_k\}_{k=1}^N$ spans V and is also linear independent, it is called a *Hamel basis* for V. Both a frame and a Hamel 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 $f \in \mathcal{H}$,

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

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 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, the elements f_i of the frame $\{f_k\}_{k=1}^N$ are stored as column vectors of the matrix $\mathbf{S}_N$. Since N vectors can at most span an 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, then it is called a Gabor frame for $\mathbb{C}^L$.

Wavelet frames A discrete dyadic wavelet system is given by

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

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, 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, then `F` is an overcomplete or redundant frame, and if it is equal to 1, then `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 Operators 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* $\mathcal{C}_f : \mathcal{H} \to \mathbb{C}^N$, associated with the frame $\{f_k\}_{k=1}^N$, is given by

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

The analysis operator maps a function $f \in \mathcal{H}$ to the *frame coefficients* $\{\langle f, f_k\rangle\}_{k=1}^N$. The adjoint of $\mathcal{C}_f$ is the so-called *synthesis* or reconstruction operator $\mathcal{R}_f : \mathbb{C}^N \to \mathcal{H}$. The operator $\mathcal{R}_f$, associated with the frame $\{f_k\}_{k=1}^N$, is explicitly given by

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

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

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

The frame operator is invertible if, and only if, the collection $\{f_k\}_{k=1}^N$ forms a frame. Since the frame operator $\mathcal{S}$ is invertible, a function f can be perfectly reconstructed by the formulas

$$\begin{aligned} f = \mathcal{S}^{-1}\mathcal{S}f &= \sum_{k=1}^N \langle f, f_k\rangle \mathcal{S}^{-1} f_k \\ = \mathcal{S}\mathcal{S}^{-1}f &= \sum_{k=1}^N \langle f, \mathcal{S}^{-1} f_k\rangle f_k, \end{aligned}$$

which are called *frame decompositions*. It can be shown that the collection $\{\mathcal{S}^{-1}f_k\}_{k=1}^{N}$ forms a frame, and it is called the *canonical dual frame* of $\{f_k\}$. The existence of the canonical dual frame guarantees that a function f can be perfectly reconstructed from its frame coefficients.

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.

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`. $\Diamond$

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.