International Journal of Wavelets, Multiresolution and Information Processing
© World Scientific Publishing Company

The Linear Time Frequency Analysis Toolbox

Peter L. Søndergaard
Centre for Applied Hearing Research,
Technical University of Denmark,
Building 352, DK-2800 Lyngby, Denmark

Bruno Torrésani
Université de Provence,
LATP, CMI,
39 rue Joliot-Curie, 13453 Marseille Cedex 13, France

Peter Balazs
Acoustics Research Institute,
Austrian Academy of Sciences,
Wohllebengasse 12-14, 1040 Wien, Austria

The Linear Time Frequency Analysis Toolbox is a Matlab/Octave toolbox for computational time-frequency analysis. It is intended both as an educational and computational tool. The toolbox provides the basic Gabor, Wilson and MDCT transform along with routines for constructing windows (filter prototypes) and routines for manipulating coefficients. It also provides a bunch of demo scripts devoted either to demonstrating the main functions of the toolbox, or to exemplify their use in specific signal processing applications. In this paper we describe the used algorithms, their mathematical background as well as some signal processing applications.

1. Introduction

Time-frequency analysis stands as one of the major recent developments in the field of mathematical signal processing. Besides the very natural (and old) idea of providing a localised version of the Fourier transformation that led long ago to the development of the short time Fourier transform (STFT), the theory of Gabor transforms and generalisations has emerged as a new scientific domain that has found applications in many different areas.

Fourier analysis (see for example the first chapter of Ref. 29 for a review) provides expansions for signals as linear combinations of sines and cosines. In the continuous time setting, one writes

$$x(t) = \int_{-\infty}^{\infty} \hat{x}(\nu) e^{2i\pi\nu t}\, d\nu \ ,$$

where i is the imaginary unit ($i^2 = -1$), t is the time variable, ν is the frequency

variable, and the Fourier transform $\hat{x}$ of the signal x reads

$$\hat{x}(\nu) = \int_{-\infty}^{\infty} x(t) e^{-2i\pi\nu t}\, dt \ .$$

Similar expansions may be written in the infinite and finite discrete time settings (see below for more details), as well as higher dimensional situations.

The short time Fourier transform of a signal is essentially a family of Fourier transforms of localised versions of the signal, obtained by multiplying it with shifted copies of a window function. Even though the STFT is a commonly used tool in signal analysis[14,26], it has several shortcomings in a number of applications. Among these, one may mention the fact that the STFT generally represents a vastly overcomplete representation for signals, which is often not suitable (for example for signal coding, or simply in terms of computational load or memory requirements). Another consequence of overcompleteness is the non-uniqueness of the signal expansion as linear combination of localised sine waves; this may be seen as a richness for the STFT, but also introduces extra difficulties (which expansion should be used?).

Gabor analysis encompasses short time Fourier analysis by introducing a suitable mathematical framework within which the role of the window, as well as the sampling of the time-frequency domain are controlled. Gabor analysis provides expansions for signals as (discrete) linear combinations of *Gabor atoms* g_{mn}, defined as shifted and modulated copies of a reference window γ, termed the *synthesis window*, as

$$\gamma_{mn}(t) = e^{2i\pi mbt}\gamma(t - na) \ , \tag{1.1}$$

with t a (discrete or continuous) time variable, and $a, b > 0$ two time and frequency sampling constants. Gabor analysis then provides expansions of the type

$$x = \sum_{m,n} c(m,n)\gamma_{mn} \tag{1.2}$$

and algorithms for the computation of the (generally non-unique) set of coefficients $c(m,n)$ of the expansion, provided that the window γ and the constants a, b have been suitably chosen.

In a dual point of view, Gabor analysis may also be formulated in terms of the Gabor transform, which associates with a signal x the family of its inner products

$$c(m,n) = \langle x, g_{mn} \rangle \tag{1.3}$$

with Gabor atoms g_{mn}, constructed from a single *analysis window* g. In this language, Gabor analysis allows for the inversion of the Gabor transform, i.e. to express the signal x in terms of the coefficients $c(m,n)$, again provided that the window g and the constants a, b have been suitably chosen. The inversion formula then assumes the form (1.2), for some synthesis window γ. Note that Equation (1.2) is in general again non-unique in the sense that different synthesis windows could be used.

In both situations, the Gabor coefficients $c(m, n)$ may be used for various purposes: they can be analysed to provide information on the signal. They can also be modified, which induces a transformation on the signal. Besides analysis and synthesis, signal transformations via coefficients modifications represent the third main aspect of Gabor analysis. The simplest way of modifying Gabor coefficients is to multiply them pointwise with a given mask, or time-frequency transfer function. The time-frequency transfer function takes the form of a doubly labeled sequence $\mathbf{m}(m, n)$, and generates the transformation

$$\mathbb{M}_{\mathbf{m}} x = \sum_{m,n} \mathbf{m}(m, n) c(m, n) \gamma_{m,n} \ . \tag{1.4}$$

It may be shown that the transformation $\mathbb{M}_{\mathbf{m}}$, termed a Gabor multiplier[3,22], is a linear operator. By analogy with the classical linear filters commonly used in signal processing, which are defined as multiplications with a transfer function in the Fourier domain, Gabor multipliers stand as non-stationary, or time-varying linear filters.

It was proved (see Ref. 29 for a review) that classical Gabor analysis may also suffer from some drawbacks, which motivated several authors to propose modified or generalised versions of Gabor analysis. One of these drawbacks is the impossibility of finding smooth windows g and sampling constants a, b such that the corresponding family of Gabor atoms would generate an orthonormal basis of the space of signals (the reader not familiar with the elements of linear algebra relevant for signal processing may want to consult the first chapter of Ref. 56 for an outline). Mainly two constructions were proposed for time-frequency like orthonormal bases: Wilson bases, and MDCT (modified discrete cosine transform) bases. Both are based upon a subtle modification of the construction rule of Gabor functions, which overcome the obstruction that prevents Gabor atoms from forming orthonormal bases. MDCT bases have become extremely popular in practical applications, as they are commonly used in audio coders such as the standards mp3, ogg vorbis and mpeg4 aac see Ref. 60 and references therein).

Such linear time-frequency decomposition methods actually offer a wide variety of expansion methods, which are adapted to various situations. In the signal processing domain, redundant representations (for example Gabor expansions with small values of the product ab) are generally preferred for signal analysis purposes, as they often allow the user to "read" relevant information from the transform, and proceed to further tasks such as detection, parameter estimation, and others. Furthermore, redundant representations turn out to be extremely useful for building time-varying signal filters, as mentioned above. On the contrary, applications such as signal coding and compression prefer to avoid redundancy, and Gabor systems with low redundancy and Wilson or MDCT bases are then preferred. This is also the case for signal denoising, even though very little is known regarding denoising in redundant systems.

The LTFAT toolbox features a number of linear time-frequency transformation tools, pure frequency transforms, and some signal processing tools including examples and test signals.

This toolbox started as the PhD project[49] of the first author. It is planned as an open-source, well-documented and extensive linear time-frequency toolbox for MATLAB/Octave freely available on the net. This paper could serve as a general literature citation when LTFAT is used for academic purposes.

As we deal with algorithms, all signals and matrices are finite-dimensional and periodic. We regard vectors (e.g., discrete signals) $x = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{C}^L$ as periodic functions on $\mathbb{Z}$ (with period L), so $x_{i+k\cdot L} = x_i$ for all $i, k \in \mathbb{Z}$. On this vector space we have a (Euclidean) norm $\|x\|$ which is induced by the scalar product $\langle x, y\rangle = \sum_{i=0}^{L-1} x_i \overline{y}_i$. Every linear operator $A : \mathbb{C}^L \to \mathbb{C}^M$ can be identified with a matrix vector multiplication $A(x) = A \cdot x = \left(\sum_{j=0}^{L-1} a_{i,j} x_j\right)_{i=0..M-1}$, where $A = (a_{i,j})_{M,L}$ is an $M \times L$ matrix. Note that all signals, windows and transforms are considered periodic, as this gives the fastest algorithms and the simplest mathematics, at the expense of the sometimes unnatural periodic boundary condition.

Section 2 introduces the basic tools from discrete Fourier analysis and discrete time/frequency analysis. These methods are mostly intended for teaching and general exploration of the field.

Section 3 introduces the three time/frequency transforms: The Gabor transform, the Wilson transform and the MDCT. These are the main computational methods in the toolbox.

Section 4 introduces a collection of tools that can be used for the construction of windows (filter prototypes) for the transforms.

Section 5 introduces a collection of signal processing tools and examples that complements the time-frequency methods. These tools and examples demonstrate how time-frequency analysis is useful for a range of signal processing tasks.

Section 6 discusses the algorithms and implementation of the toolbox.

We will link the mathematical background to the algorithms in LTFAT. We will denote the name of a MATLAB function in typewriter style (`functioname`).

2. Basic Fourier Analysis

The toolbox contains some basic Fourier analysis tools intended mostly for teaching. This includes `dft`, a unitary discrete Fourier transform (DFT), its inverse `idft`, periodic convolution `pconv` and involution `involute`.

The DFT $c \in \mathbb{C}^L$ of a signal $f \in \mathbb{C}^L$ is given by

$$c(k) = \frac{1}{\sqrt{L}} \sum_{l=0}^{L-1} f(l) e^{-2\pi i k l / L}, \quad k = 0, \ldots, L-1. \tag{2.1}$$

The inverse transform reads

$$f(l) = \frac{1}{\sqrt{L}} \sum_{k=0}^{L-1} c(k) e^{2\pi i k l/L}, \quad l = 0, \ldots, L-1. \tag{2.2}$$

The invertibility of the DFT originates from the fact that the family of vectors $\epsilon^k \in \mathbb{C}^L$ defined by their components

$$\epsilon_l^k = \frac{1}{\sqrt{L}} e^{2\pi i k l/L}, \quad l = 0 \ldots L-1 \tag{2.3}$$

form an orthonormal basis of $\mathbb{C}^L$.

The standard DFT implementation `fft` that is included in Matlab and Octave is normalised differently. The periodic convolution $h \in \mathbb{C}^L$ of two signals $f, g \in \mathbb{C}^L$ is given by

$$h(l) = \sum_{k=0}^{L-1} f(k) \overline{g(l-k)}, \quad l = 0, \ldots, L-1. \tag{2.4}$$

In addition to the Discrete Fourier transform, the toolbox also contains the classical discrete cosine and sine transforms type I-IV (`dcti, dctii, dctiii` and `dctiv`, and corresponding `dst` functions, see Ref. 48 for a review). These transforms are real valued (real valued input gives real valued output) as opposed to the DFT, and they are therefore sometimes better suited for practical applications on real valued signals. In particular, the DCT is the mathematical transform used in JPEG image compression. For an overview of the statistics of DCT coefficients for natural images, see Ref. 42.

The toolbox contains a list of functions with special behaviour in time and frequency: A periodic chirp `pchirp` that grows linearly in frequency, two families of Hermite functions which are invariant with respect to the DFT, various window functions to be described below and the Shah distribution `shah` (also known as a pulse train).

Remark: The FFT for real signals It is well known that the Fourier transform of a real-valued vector possesses the Hermitean symmetry: if $f \in \mathbb{R}^L$, then $\hat{f}(L-1-n) = \overline{\hat{f}}(n)$ for all $n = 0, \ldots L-1$. Therefore, the computation of half of the values of the Fourier transform is not necessary, which results in interesting savings in terms of memory and computing time. The toolbox includes a version of FFT adapted to real-valued vectors, which only returns the useful values of the Fourier transform (i.e.the first half, which actually corresponds to positive frequencies, due to the periodicity of the DFT). The corresponding function is termed `fftreal`, and the inverse FFT for real-valued vectors (i.e. using only positive frequencies) is `ifftreal`.

3. Time/Frequency transforms

The prototype of time-frequency transform is the *Short Time Fourier Transform* (STFT). The STFT of a signal $f \in \mathbb{C}^L$ is obtained by Fourier transforming copies of

f that are localised by pointwise multiplication with a sliding window $l \to g(l-n)$, i.e. by computing

$$\mathrm{STFT}(m,n) = \sum_{l=0}^{L-1} f(l)\overline{g}(l-n)e^{-2i\pi ml/L} \ , \quad m,n = 0,\dots L \ , \tag{3.1}$$

the bar denoting complex conjugation, and being purely conventional here (for coherence with the next sections). The simplest choice for the window g, namely a rectangular window, turns out to be quite inappropriate in practice because of its poor spectral localisation, and smoother window functions are generally preferred.

The STFT is invertible: a direct calculation yields

$$f(l) = \frac{1}{\|g\|^2} \sum_{m,n=0}^{L-1} \mathrm{STFT}(m,n)g(l-n)e^{2i\pi ml/L} \ , \tag{3.2}$$

where $\|g\|$ is the Euclidean norm of the vector g. However, it is worth mentioning that the STFT of a signal represents highly redundant information, since a signal $f \in \mathbb{C}^L$ is represented by L^2 coefficients $\mathrm{STFT}(m,n)$. Such a redundancy may be useful in a number of applications. However, for large L, the large size of full STFT is often not suitable. One has to subsample the STFT, which leads to Gabor transforms. We describe below the Gabor transform, together with two variants, which have the advantage of yielding orthonormal time-frequency bases, which is impossible if one limits to Gabor. The corresponding transforms are generated from windows, which we also comment on in the corresponding sections. In particular, the windows have to possess some symmetry properties to generate MDCT and Wilson bases.

The STFT is a particular case of the Gabor transform, see next section, and can be calculated in LTFAT by `dgt`.

3.1. *The discrete Gabor transform*

The STFT inversion formula (3.2) may be interpreted as an expansion of the signal f with respect to a (redundant) system of elementary waveforms

$$l \longrightarrow g(l-n)e^{2i\pi ml/L} \ , \qquad m,n = 0,\dots L-1.$$

Subsampling this redundant system leads to the so-called Gabor systems, also called Weyl-Heisenberg systems, or Fourier modulated filter banks[12]:

$$l \longrightarrow g(l-an)e^{2i\pi ml/M} \ , \qquad m = 0,\dots M-1, n = 0,\dots N-1. \tag{3.3}$$

The following parameters describe the size of a discrete Gabor transform:

a : Time separation.

M : Number of frequency bands.

L : Length of signal.

This implicitly assumes that $L = Mb = Na$, which in practice can be accomplished by truncating the signal to the appropriate length, or adding zeroes to the signal[a]. With these parameters, the coefficients $c \in \mathbb{C}^{M \times N}$ computed by the discrete Gabor transform of the signal f are given by:

$$c(m,n) = \sum_{l=0}^{L-1} f(k) e^{-2\pi i m l/M} \overline{g(l-an)}. \tag{3.4}$$

These coefficients are samples of the discrete short-time Fourier transform of the signal. They are complex, even if both window and input signal are real.

Please note that sometimes the terms STFT or Short Time Spectrum are used for both the (full) STFT and the Gabor transform, see e.g. Ref. 1.

Regarding inversion from a discrete Gabor transform, it is convenient to introduce the following notion.

Definition 1. A family of vectors e_j, $j = 0, ..., J-1$ of length L is called a *frame* if constants $0 < A \leq B$ exist such that

$$A \|f\|^2 \leq \sum_{j=0}^{J-1} |\langle f, e_j \rangle|^2 \leq B \|f\|^2, \quad \forall f \in \mathbb{C}^L. \tag{3.5}$$

The constants A and B are called lower and upper frame bounds. If $A = B$, the frame is called *tight*. If $J > L$, the frame is redundant (oversampled).

The redundancy of the frame is the fraction $\frac{J}{L}$. Finite- and infinite dimensional frames are described in Ref. 16.

In the finite dimensional setting we are concerned with, a frame is essentially a family of vectors which is complete in the considered signal space. The right hand side inequality is always fulfilled for a finite set of vectors, however the introduction of frame bounds is still useful, since it allows control of the convergence rate of the inversion algorithm. A frame may be a basis, but interesting frames are often redundant.

A nice and useful feature of frames is the existence of (generally infinitely many) *dual frame(s)*, from which signal expansions may be obtained. More precisely, given a frame $\{e_j, j = 0, ..., J-1\}$, there exists a family $\{\tilde{e}_j, j = 0, ..., J-1\}$ such that for all $f \in \mathbb{C}^L$,

$$f = \sum_{j=0}^{J-1} \langle f, e_j \rangle \tilde{e}_j = \sum_{j=0}^{J-1} \langle f, \tilde{e}_j \rangle e_j.$$

Among the dual frames, the so-called *canonical dual frame* has a specific status, e.g. it gives minimal norm coefficients[16]. The canonical dual frame is constructed as follows.

[a]Such options are implemented in the toolbox.

Consider the linear transformation $S : \mathbb{C}^L \to \mathbb{C}^L$, called the *frame operator*, defined by

$$(Sf)(l) = \sum_{j=0}^{J-1} \langle f, e_j \rangle e_j(l) \ , \qquad l = 0, \dots L-1 \ . \tag{3.6}$$

It can be shown that S is invertible. The canonical dual frame is then defined by

$$\tilde{e}_j^o = S^{-1} e_j \ , \qquad j = 0, \dots J-1 \ . \tag{3.7}$$

In the particular case of Gabor frames, the canonical dual frame of a Gabor frame is still a Gabor frame, and is therefore generated using the *canonical dual window*

$$g^d = S^{-1} g \ . \tag{3.8}$$

Another window of special importance is the *canonical tight window*

$$g^t = S^{-1/2} g. \tag{3.9}$$

This window generates a tight Gabor frame and gives perfect reconstruction if it is used for both analysis and synthesis, much like in the situation of an orthonormal basis. Given an analysis window g, and lattice constants a and b that generate a Gabor frame, a vector h is an admissible dual window if for all $f \in \mathbb{C}^L$,

$$f = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \langle f, g_{mn} \rangle h_{mn} \ . \tag{3.10}$$

To obtain perfect reconstruction with well-behaved windows, it is necessary that the Gabor system be redundant, so the Gabor system is a frame, and not a basis. A Gabor frame is redundant if $a < M$. For detailed information about Gabor systems, see the books 29, 23, 24.

The discrete Gabor transform is implemented in the function `dgt`. Besides the signal and the parameter choices, `dgt` takes as one of its arguments the window. See below for more details on window generation. The inverse Gabor transform is implemented in `idgt`.

Remark: the DGT for real signals Like the FFT, the discrete Gabor transform of real valued vectors possesses the Hermitean symmetry, *if the window is itself real-valued*,

$$c(M-1-m, n) = \overline{c(m,n)} \ , \quad n = 0, \dots N-1, \ m = 0, \dots M-1$$

so that again only half of the DGT coefficients are necessary to characterise the whole family. The toolbox implements the function `dgtreal` (resp. `idgtreal`) that account for that particular symmetry, and return (resp. take as input variable) only the relevant half of coefficients.

3.2. *The discrete Wilson transform*

Wilson bases were proposed as substitutes for Gabor frames, because of the impossibility of constructing Gabor systems that would be simultaneously generated using well-behaved windows, and bases of the considered signal spaces.

A Wilson basis is formed by taking linear combinations of appropriate basis functions from a Gabor frame with redundancy 2, see Ref. 11. Essentially Gabor atoms of positive and negative frequencies are combined, with suitable fine tuning of their phases. This remarkable construction turns a tight Gabor frame into an real, orthonormal basis, or turns a non-tight Gabor frame into a Riesz basis (corresponding to a bi-orthogonal filter bank). In Ref. 36 this system is described as a "linear phase cosine modulated maximally decimated filter bank with perfect reconstruction". Is is currently an open question whether there exist Wilson bases obtained from a Gabor frame with another redundancy than 2. A partial answer has been given in Ref. 62.

The coefficients $w \in \mathbb{C}^{2M \times \frac{L}{2M}}$ computed by the Discrete Wilson Transform, `dwilt`, of the signal $f \in \mathbb{C}^L$ are given by

If $m = 0$:

$$w(0,n) = \sum_{l=0}^{L-1} f(l) g(l - 2na). \tag{3.11}$$

If m is odd and less than M:

$$w(m,n) = \sqrt{2} \sum_{l=0}^{L-1} f(l) \sin(\pi \frac{m}{M} l) g(l - 2na), \tag{3.12}$$

$$w(m+M,n) = \sqrt{2} \sum_{l=0}^{L-1} f(l) \cos(\pi \frac{m}{M} l) g(l - (2n+1)a). \tag{3.13}$$

If m is even and less than M:

$$w(m,n) = \sqrt{2} \sum_{l=0}^{L-1} f(l) \cos(\pi \frac{m}{M} l) g(l - 2na), \tag{3.14}$$

$$w(m+M,n) = \sqrt{2} \sum_{l=0}^{L-1} f(l) \sin(\pi \frac{m}{M} l) g(l - (2n+1)a). \tag{3.15}$$

If $m = M$ and M is even:

$$w(M,n) = \sum_{l=0}^{L-1} f(l)(-1)^l g(l - 2na) \tag{3.16}$$

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

$$w(M,n) = \sum_{k=0}^{L-1} f(l)(-1)^l g(l - (2n+1)a). \tag{3.17}$$

Some notes on this: $w(0,n)$ and $w(M,n)$ only have half the bandwidth of the other filters, as cosine and sine include two complex frequencies, one positive and one negative. This implies that an M-channel Wilson filter bank will split a signal into $M+1$ frequency bands, with the highest and lowest frequency band having half the bandwidth.

If a Wilson basis is considered as an $2M$-channel filter bank, then roughly half of the filters ($M-1$ if M is even, otherwise $M+1$) are time-shifted by M samples.

The Wilson transform is implemented in the function `dwilt`. Besides the signal and the parameter choices, `dwilt` takes as one of its arguments the window, which has to be whole point even (see below for more details on window generation). The inverse Wilson transform is implemented in `idwilt`.

3.3. *The modified discrete cosine transform (MDCT)*

The MDCT (modified discrete cosine transform) is another substitute for the non-existent well localised Gabor bases that has become extremely popular recently for its numerous applications, in audio coding for instance[39,45,46]. The main idea is again to provide a local version for trigonometric bases, using smooth windows rather than rectangular ones.

The coefficients $c \in \mathbb{C}^{M\times N}$ computed by the MDCT of $f \in \mathbb{C}^L$ are given by:

For $m+n$ even:

$$w(m,n) = \sqrt{2}\sum_{l=0}^{L-1} f(l)\cos\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)l+\frac{\pi}{4}\right)g(l-na). \tag{3.18}$$

For $m+n$ odd:

$$w(m,n) = \sqrt{2}\sum_{l=0}^{L-1} f(l)\sin\left(\frac{\pi}{M}\left(m+\frac{1}{2}\right)l+\frac{\pi}{4}\right)g(l-na). \tag{3.19}$$

Notice that this definition of the MDCT is not the most common one: the common definition of the MDCT require the window to have the HPE symmetry (see Sec. 4), whereas this definition uses WPE windows just as the Gabor and Wilson transforms.

The LTFAT form of MDCT transform is implemented in the function `wmdct`. The inverse transform is implemented in `iwmdct`.

4. Window design

The toolbox can use both FIR (finite impulse response) and IIR (infinite impulse response) windows. In this finite setting, the meaning of IIR and FIR is the following: FIR windows are shorter than the signal to be analysed, while IIR windows will have the same length.

The intent of the toolbox is to stay close to the underlying mathematics, and we have therefore adopted a different layout of windows than what it usually done

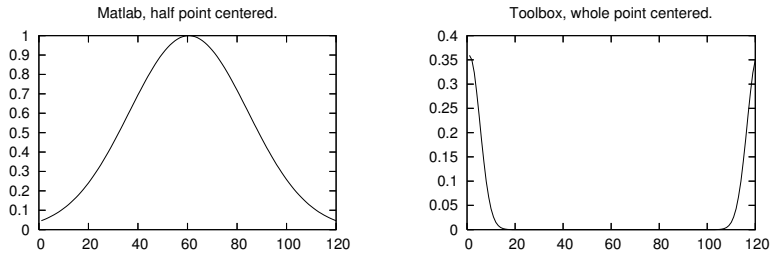

Figure 4.1. The figure to the left shows the output of the Matlab command `gausswin(120)` while the figure to the right show the output of the LTFAT command `pgauss(120)`.

in Matlab/Octave. This difference is visible in FIGURE 4.1, which shows a Gaussian window computed by the Matlab/Octave `gausswin` and the LTFAT command `pgauss`. The command `gausswin` produces an FIR window made from a truncated Gaussian function that is centered in between the two middle points in the vector. The command `pgauss` produces a long window which is the Gaussian window centered around the first element of the vector.

Centering the window around the first element makes it look unnatural when plotting, however it has the benefit that the Gaussian window is invariant with respect to the DFT, and that the window generate a zero delay filter.

The default symmetry in Matlab is the *half point even* (HPE) symmetry. If g is an HPE window of length L, then

$$g(l) = \overline{g(L-1-l)} \tag{4.1}$$

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

The windows in LTFAT are *whole point even* (WPE) meaning that for a window g of length L then

$$g(l) = \overline{g(-l)} = \overline{g(L-l)} \tag{4.2}$$

for $l = 0, \ldots, L-1$. This implies that $g(0)$ must always be real, and so must $g\left(\frac{L}{2}+1\right)$ if L is even. A signal g is WPE if and only if the DFT of g is real valued. In signal processing, such a window is said to have *zero phase*.

All the other window functions in the toolbox work similarly to `pgauss`. They return WPE windows centered around the first element of the vector. A routine `middlepad` is included to cut or extend these kind of windows.

4.1. *FIR windows*

While every signal (and therefore every window) has finite length in a finite-dimensional setting, we use the term *finite impulse response* (FIR) for windows, whose length is shorter than the signal length L. This is important in practical situations where long signals have to be processed, and windows as long as the signal

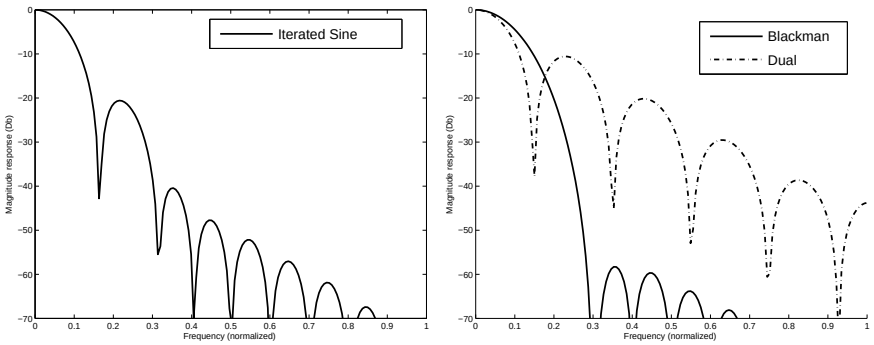

Figure 4.2. The left figure shows the magnitude response of an iterated sine window, used in the Ogg-Vorbis audio codex. This window generates a tight Gabor frame or an orthonormal Wilson/MDCT basis. The figure on the right shows the magnitude response of the Blackman window and its dual window. Together, these windows generate a pair of dual Gabor frames, or a pair of bi-orthogonal Wilson/MDCT bases. The magnitude response plots have been created with the `magresp` command.

would incur a large computational load or a long processing delay. The utility function `long2fir` converts a long window into a shorter one by discarding unnecessary values. `fir2long` converts short windows to full length windows.

Finite impulse response (FIR) windows can be generated by the routines `firwin` and `firkaiser`. The routine `firwin` generates the classical Hann, Hamming, Blackman and Nuttall windows, and in addition it also returns windows that generate tight Gabor frames or orthonormal Wilson/MDCT bases. These include the sine window and the iterated sine window[57]. The routine `firkaiser` generates the Kaiser-Bessel window (see Ref. 43 for definitions). The magnitude response of some windows can be seen on FIGURE 4.2.

4.2. *IIR windows*

IIR stands for *infinite impulse response*. Again this meaning of this term has to be adapted in the finite-dimensional setting. We refer to IIR windows, if their length is equal to signal length L. Note that this includes any window, and not just windows with a small number of poles in the z-domain.

The toolbox contains routines to generate Gaussian (`pgauss`), Sech (`psech`) and Gauss-Hermite (`pherm`) windows. These windows are all sampled and periodised version of their continuous counterparts. They are invariant with respect to a discrete Fourier transform, just as their continuous counterparts are invariant with respect to the Fourier transform.

The finite, discrete Gaussian $\varphi_w \in \mathbb{C}^L$ computed by `pgauss` is given by:

$$\varphi_w(l) = \left(\frac{wL}{2}\right)^{-1/4} \sum_{k\in\mathbb{Z}} e^{-\pi\left(\frac{(l+c_t)}{\sqrt{L}} - k\sqrt{L}\right)^2/w}, \tag{4.3}$$

where the parameter $w > 0$ controls the ratio of the time-support and the frequency support of the Gaussian, and c_t is a centering parameter. Setting $c_t = 0$ generates a whole point centered function and setting $c_t = 1/2$ generates a half point centered function, as most Matlab filtering routines.

The routines `psech` and `pherm` work entirely similar to `pgauss` taking the same parameter w as input.

The routine `pbspline` can generate several different classes of discrete, fractional splines. These splines are described in detail in Ref. 50. It may also simply be used to generate sampling of the classical B-splines.

4.3. *Dual / tight windows*

The canonical dual window (3.8) of a Gabor frame may be easily calculated by `gabdual`, and the canonical tight window (3.9) by `gabtight`. For Wilson the Riesz dual window is computed by `wildual` and an orthonormal window may be generated by `wilorth`. For MDCT bases the same window algorithms as for Wilson can be used.

To judge the quality of a given Gabor system, two methods are supplied. One possibility is to consider the frame bounds (A and B from (3.5)) of a Gabor frame, these can be calculated by `gabframebounds`. The ratio $\frac{B}{A}$ of the frame bounds play the exact same role as the condition number of a matrix.

Another method is to consider the reconstruction quality of a pair of windows g, γ. The function `gabdualnorm` will return the maximal reconstruction error of any signal when using the pair g, γ as windows for analysis and synthesis[8,32]. In particular for two windows that generate dual Gabor systems, the numerical precision is returned.

5. Some examples of time-frequency signal processing using LTFAT

The toolbox includes special support for some simple signal processing tasks. This has been done to make the toolbox more self-contained, so that is it possible to teach and demonstrate aspects of time/frequency signal processing using only the tools available in the toolbox.

The *phase vocoder*[25] is a well-known signal processing algorithm. It is an analysis / synthesis system, allowing modification of the coefficients. There are two ways to interpret the phase vocoder[18], either as filter bank or a Gabor transform. Thus, it can be implemented using `dgt`, followed by an estimation of the instantaneous frequency using the temporal derivative of the phase, compare Section 5.1. A typical application of the phase vocoder is time stretching. For phase vocoder techniques the parameter a is called the *hop size* and M the *FFT length*.

5.1. *Signal analysis, visualisation and estimation*

The first step of most signal processing problems is generally signal analysis. Time-frequency signal analysis generally requires visualisation tools. The toolbox includes a few plotting routines, including among others

- A spectrogram plot `sgram`. This is an easy-to-use function, that displays a spectrogram of the input function. It is based on a suitable Gabor frame. The spectrogram displays the squared amplitude of the Gabor transform, i.e. $|c(m,n)|^2$. A simple example is provided in `demo_sgram`, see Figure 5.1.

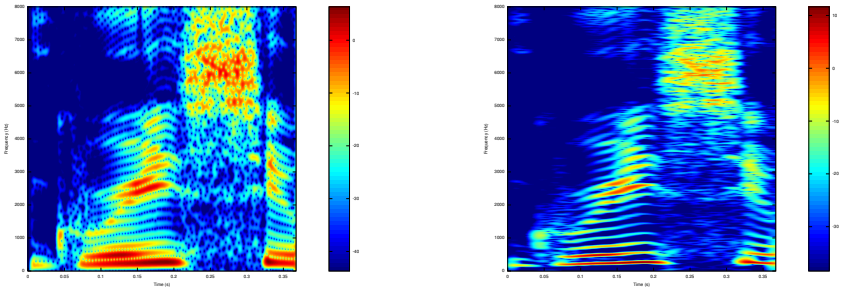

Figure 5.1. The figure on the left shows a spectogram of a speech signal, a female speaker pronouncing the work "greasy". The spectrogram has been generated with an equal resolution in time and frequency. The right figure shows a spectrogram of the same signal, but with a time resolution of 20 ms. This is the output of `demo_sgram`.

- A phase plot, illustrated in `demo_phaseplot`, see Figure 5.2. In the spectrogram the information about the complex angle of the coefficients $c(m,n)$, i.e. the *phase*, is lost. The phase may sometimes reveal features not visible on the spectrogram[14].
  Notice that the phase of the `stft` as defined in (3.1) is generally not interpretable, as it is not time-covariant: the phase of a shifted copy of the signal is not equal to the shifted copy of the original signal's phase. The *phase-locked* version, in which covariance has been restored, is generally preferred. For example, in the phase-locked version, constant phase lines converge to the locations of singularities, as can be seen in Figure 5.2. The phase-locked version STFT_L of the stft is linked to the classical one by the expression

$$\mathrm{STFT}_L(m,n) = \sum_{l=0}^{L-1} f(l)\overline{g}(l-n)e^{-2i\pi m(l-n)/L} =$$

$$= e^{2i\pi mn/L}\mathrm{STFT}(m,n)\ , \quad m,n = 0,\dots L\ .$$

Phase locking (resp. phase unlocking, i.e. the inverse transformation) of Gabor transform is implemented in `phaselock` (resp. `phaseunlock`).

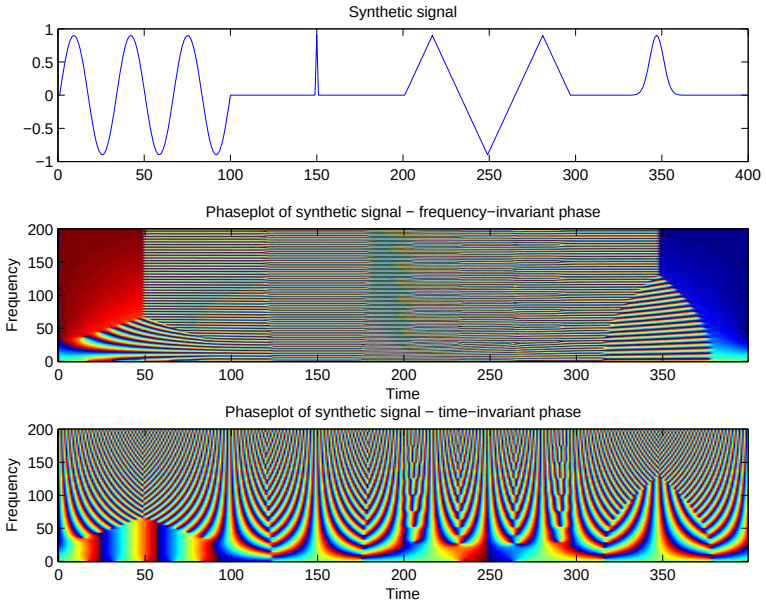

Figure 5.2. This figure is part of the output from `demo_phaseplot`. (TOP): The synthetic signal analyzed. (MIDDLE): The non-phaselocked phase values. (BOTTOM): The phaselocked phase.

Visualisation is generally the first step in a signal processing task. The visualised time-frequency transform can then be processed in various ways, for example for signal or parameter estimation. The toolbox also provides useful alternative representations, including time-frequency phase gradients, which are of interest in some signal processing applications, and reassigned Gabor transforms. The `gabphasegrad` function provides numerical estimates of the time and frequency derivatives of the Gabor transform phase, or more precisely the difference between the latter and the intrinsic derivatives at the given time-frequency point. These estimates are useful for example for computing reassigned spectrograms. Reassignment was proposed as a way to enhance the time-frequency resolution of the usual spectrogram, see Ref. 2 for a review. An example (which can be generated using the script `demo_reassign`) is provided in Figure 5.3.

5.2. *Time variant filtering*

Time-invariant filter are systems, where the frequency spectrum is multiplied by a fixed function, called the *transfer function*[43], also defined as the Fourier transform of

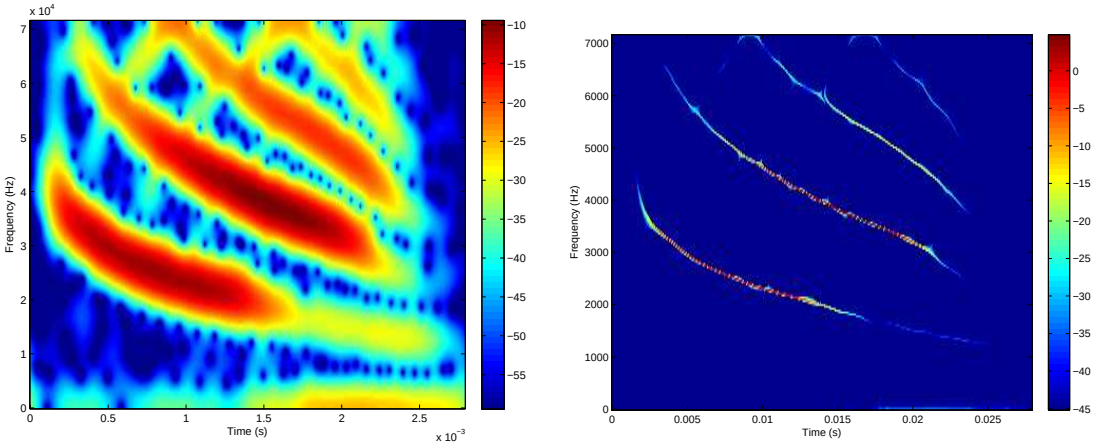

Figure 5.3. Gabor transform modulus (left) and reassigned Gabor transform modulus (right) of a bat sonar signal.

the filter's impulse response. Using the Fourier transform to calculate the spectrum, such an operator can also be called a *Fourier multiplier*. This technique has been used for many years and finds a wide range of applications in all areas of signal processing, for example in audio applications[52].

A generalisation of this technique is the so called *time-variant filtering*[59], which has become more important recently. In time variant filters, the impulse response varies with time, and the Fourier transform is not adequate any more. Localised versions of spectral analysis can be used to characterise and synthesise such filters. If the STFT is used in its sampled version, the Gabor transform, one possibility to construct a time variant filter is the usage of *Gabor multipliers*[22], as in Equation (1.4)[b]. The latter consists of Gabor transform, followed by pointwise multiplication with a fixed time-frequency vector (called the symbol) and then by inverse Gabor transform. These operators have been already used for quite some time implicitly in engineering applications and recently have been used in signal-processing applications as time-variant filters called *Gabor filters*[40]. Recent applications can be found for example in the fields of source separation[63], system estimation[38] or psychoacoustical modelling[9]. The blind source separation application relies on the assumption that several observations of the signal are available that are linear combinations of unknown sources with unknown coefficients, and that the effective supports of the time-frequency representations of the sources do not overlap. In such a situation, the supports can be estimated, and the different sources recovered by applying an

[b]Such operators can be defined for any kind of frames[3], including also Wilson or MDCT bases. However redundancy allows one to describe larger classes of operators. These operators are naturally connected to the concept of weighted frames[5]. If the symbol is shifted to the involved frames, the multiplier corresponds to the frame operator of the weighted frames.

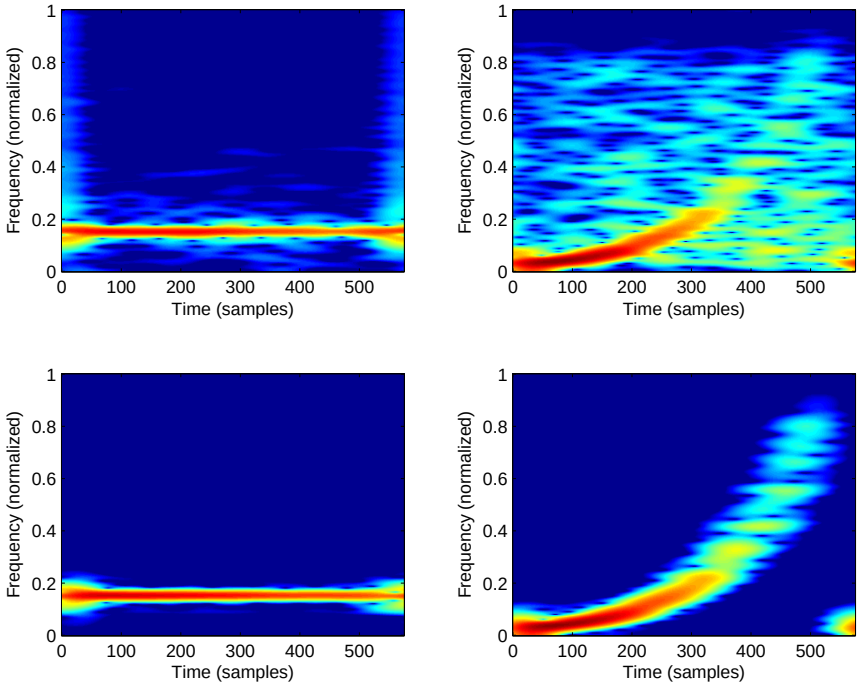

Figure 5.4. Comparison of the spectrogram of the output signal of a slowly time-varying system (TOP) and the the best approximation by a Gabor multiplier (BOTTOM) applied on the same signals, on a sinusoid (LEFT) and an exponential sweep (RIGHT) . This is a part of the output of `demo_gabmulappr`.

appropriate time-frequency multiplier to the signal. The symbol of the latter may be either binary (in the so-called DUET approach), or more elaborate.

The toolbox contains a routine `gabmul` for easy application of Gabor multipliers, whose analysis and synthesis windows, lattice constants and symbol can be specified by the user. The script `demo_gabmul` shows the effect of Gabor multipliers on a Gaussian white noise, with various choices of the multiplier's symbol.

While any linear transformation can be written as a time-variant filter, Gabor multipliers only describe a subclass of these. However, Gabor multipliers can also yield good quality approximations for many time-varying filter, provided parameters are suitably chosen. The function `gabmulappr` provides optimal Gabor multiplier approximations for any linear operator (represented by its matrix), using the algorithm presented in Ref. 20[c]. The use of this function is exemplified in `demo_gabmulappr`.

Gabor Multipliers can only approximate operators well with localised *spreading*

[c]Together with a reference code[4] which can be used for any type of frame.

function[20], i.e. containing only small time and frequency shifts. For filters, i.e. linear time-variant system, this means that only filters with a small support of the impulse response can be well approximated by Gabor Multipliers[7]. Such an approach is used in some software systems to efficiently implement linear time-variant systems.

For time-variant systems this fact is displayed in Figure 5.4. A slowly time-varying system and its best approximation by a Gabor multiplier are applied on a sinusoid and an exponential sweep[d]. In the figure it can be seen that the 'local behaviour' is caught by the multiplier, but large time-frequency shifts are not represented.

5.3. *Denoising*

Denoising, or noise reduction, is one of the classical tasks in signal processing (see Ref 61 and references therein). Denoising is often performed by time-invariant filtering (as for example the Wiener filter), which is often implemented by pointwise multiplication in the Fourier domain. When the noise is non-stationary, denoising can also be realised through time-varying filtering, using time-frequency multipliers.

However, the emergence of time-frequency tools has made it possible to develop efficient methods for performing denoising using thresholding strategies[19]. The rationale is the following. If a signal of interest is represented by a sparse expansion with respect to a given basis, noise is generally not. Therefore, the signal's energy is concentrated in a small number of coefficients, while the noise's energy is spread on all coefficients. Thresholding the coefficients of the noisy signal allows one to get rid of most of the noise contribution.

Coefficient thresholding is usually performed in two different ways. Denote by c_k the coefficients of the expansion of a signal with respect to a given system. Hard thresholding H_τ simply sets to zero all coefficients values c_k whose modulus is lower than a given threshold τ:

$$H_\tau c_k = \begin{cases} c_k & \text{if } |c_k| \geq \tau \\ 0 & \text{otherwise.} \end{cases}$$

Soft thresholding S_τ sets to zero all coefficients values c_k whose modulus is lower than a given threshold τ and decreases the modulus of the other by τ:

$$S_\tau c_k = \begin{cases} \operatorname{sgn}(c_k)(|c_k| - \tau) & \text{if } |c_k| \geq \tau \\ 0 & \text{otherwise.} \end{cases}$$

The toolbox includes tools for hard and soft coefficient thresholding, both implemented in the `thresh` code.

It is known that local Fourier type bases allow a sparse representation of most audio signals. An example of audio signal denoising using hard and soft thresholding of MDCT coefficients is provided in the function `demo_audiodenoise`. Corresponding numerical results are shown in FIGURE 5.5. In that example, the noise was an

[d]Exponential sweeps are often used for system estimation in audio applications[38,37].

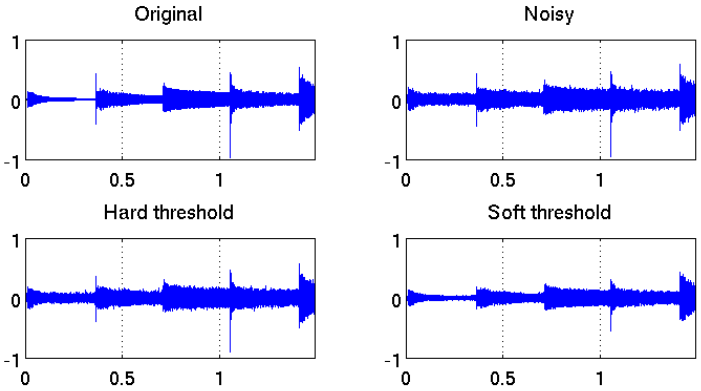

Figure 5.5. Denoising of glockenspiel audio signal using Hard and Soft thresholding of MDCT coefficients. Input SNR: 6.0 dB; Output SNR (Hard): 7.22 dB; Output SNR (Soft): 13.30 dB

additive white Gaussian noise, and the threshold has been chosen in such a way that the number of significant MDCT coefficients after denoising be approximately the same as the number of significant MDCT coefficients of the original signal. In practical situations, the threshold has to be estimated from data, which makes the denoising problem harder.

Coefficient thresholding has introduced itself naturally into the context of variational formulations of denoising. For example, given an orthonormal basis $\{\varphi_k\}$, the variational denoising problem (termed the lasso problem)

$$\min_{\alpha} \left[\frac{1}{2} \left\| x - \sum_k \alpha_k \varphi_k \right\|_2^2 + \lambda \|\alpha\|_1 \right]$$

is solved by $\alpha_k = S_\lambda \langle x, \varphi_k \rangle$. This is not true any more when $\{\varphi_k\}$ is a more general frame, but dedicated iterative algorithms (based upon soft thresholding[17]) can yield the solution. Such algorithms are implemented in the function `gablasso` for Gabor frames. The script `demo_gablasso` provides an illustration of the use of the function. The toolbox also includes other related functions, such as `wmdctlayerdecomp`, which solves the same ℓ_1-penalised regression problem in a union of two MDCT bases instead of a Gabor frame (using block coordinate relaxation algorithms[55,35]) and variants exploiting mixed norms regularisations[34] (see `gabglasso`, `demo_gabglasso`, `wmdctglayerdecomp`, and `demo_layerdecomp`). A nice outcome of the layer decomposition is that it allows separate estimation of the components with respect to the two bases. An example of such a decomposition is provided in `demo_layerdecomp`, in which a synthetic signal is created as the sum of two sparse expansions with respect to two MDCT bases, and the two components are recovered up to a small error. An example can be found in FIGURE 5.6.

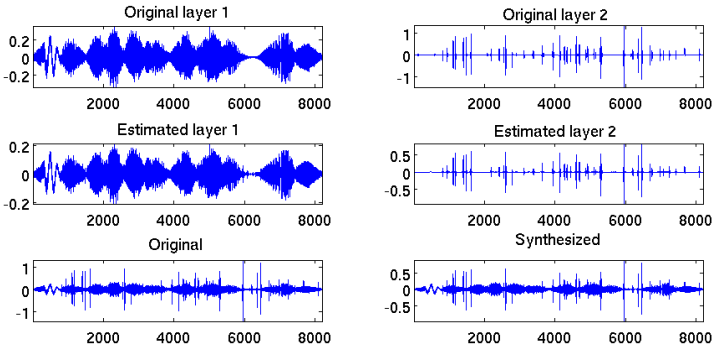

Figure 5.6. Layer decomposition: decomposition of a synthetic signal as a sum of two components (layers) that are sparsely represented in two different MDCT bases.

5.4. *Signal compression*

Signal compression (see Ref. 33 for an introduction), aims at representing a signal with maximal fidelity, using minimal storage. Transform coding is one of the most popular signal compression techniques. Transform coding starts by expanding the signal on a given basis. Then only the most significant coefficients of the expansion are retained, which provide an approximation of the signal. Finally, the retained coefficients are quantised, i.e. represented using a finite (small) number of bits.

MDCT bases form the main building block for transform coding of audio signals. After an MDCT expansion, the significant coefficient selection is generally done by keeping the MDCT coefficients corresponding to lower frequencies, i.e. coefficients $w(m,n)$ with values of m lower than some reference value M_1. These retained coefficients are then quantised, after a suitable weighting. Such schemes are called linear transform coding schemes.

An alternative, called non-linear transform coding, follows a different rule for the selection of significant coefficients. Instead of retaining low frequency coefficients, the coefficients with largest magnitude (after frequency dependent weighting if needed) are retained. The approximation error in such approaches may be proved to be always lower than the approximation error of the corresponding linear transform coding scheme (for the same number of retained coefficients).

An example of audio signal linear and non-linear approximation is provided in the function `demo_audiocompress`. Corresponding results are shown in FIGURE 5.7, where are plotted the values of SNR (Signal to Noise ratio)

$$\mathrm{SNR} = 20\log_{10}\left(\frac{\sigma_x}{\sigma_e}\right)$$

as a function of the number of retained coefficients, σ_x being the signal's standard deviation, and σ_e being the standard deviation of the (linear or non-linear)

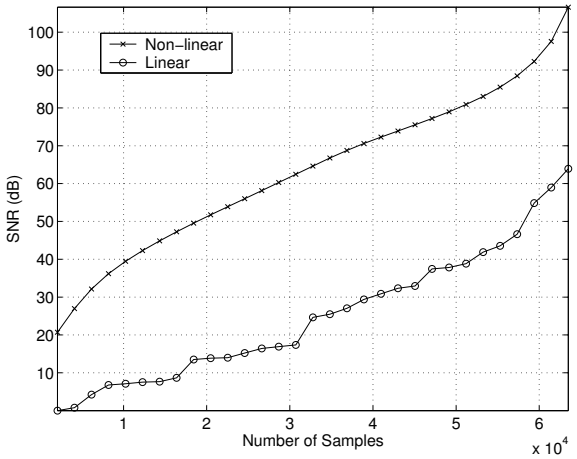

Figure 5.7. Output signal to noise ratio for linear and non-linear approximation of glockenspiel audio signal using MDCT coefficients, as a function of the number of MDCT coefficients retained in the approximation.

approximation error.

However, it is worth mentioning that in non-linear coding schemes, reconstruction from retained coefficients is only possible if their "addresses" (i.e. corresponding values of (m, n)), termed significance maps, are stored as well, which represents a significant amount of side information. Efficient strategies for representing significance maps are then necessary.

5.5. *OFDM transmission*

Orthogonal Frequency Division Multiplexing is a widely used technique for transmitting digital signals. The idea is to generate a series of pulses that are mutually orthogonal. The pulses carry the information, and the orthogonality ensures that the method is robust against noise, delay, dispersion etc.

A non-redundant Gabor system (a Gabor Riesz sequence) with time-shift $a = K$ and number of channels $M = N$ is a good model of an OFDM system (non-redundancy means that $K >= L$). Furthermore, such a system is related to the corresponding redundant Gabor frame with a time-shift of $a = N$ and number of channels $M = K$. If two windows g and γ are canonical dual with respect to the redundant Gabor system, then they are also dual windows for the non-redundant system (for its closed linear span). Using this property, the standard routines for computing tight and dual windows can be used for the construction of orthogonal and bi-orthogonal windows for OFDM. The toolbox includes an example `demo_ofdm` that demonstrates the various aspects of OFDM transmission using Gabor frames. For more information on using Gabor and Wilson systems for OFDM, see Ref. 54, 10.

6. Implementation

The toolbox uses two different algorithms for computing a DGT, depending on the type of window.

If an FIR window is used a weighted overlap-add based approach will be used. The algorithms are based on an simple application of the FFT in each timestep. This gives a computational complexity of $\mathcal{O}(NM \log M + Nd)$, where M is the number of channels, N is the number of time shifts, and d is the length of the window.

The long window variants are used when the window has the same length as the signal. A thorough analysis of the computational complexity of the main algorithms in LTFAT is presented in Ref. 51 and is based on previous work on matrix factorisations of Gabor operators done in Ref. 47, 53. The long window algorithm requires all data to be available before computing the transform. This makes it unsuitable for streaming data through a DSP, but it poses no problem in Matlab. The long window algorithm has complexity $\mathcal{O}(NM \log M + Lq)$, where L is the length of the signal, MN is the total number of coefficients and q appears in the redundancy $\frac{q}{p}$ written as an irreducible fraction (i.e. for a Gabor frame with redundancy 1.25, $q = 5$. For a Wilson basis $q = 2$ always).

The DGT algorithms (for either FIR or full length windows) are also used to calculate the canonical dual and tight windows, frame bounds and the dual norm measure. If the number of frequency bins M is larger than the size of an FIR window ('painless non-orthogonal case'), the canonical dual window can be calculated very efficiently, as the frame operator is diagonal and can therefore be easily inverted. Otherwise, canonical dual windows of FIR windows are calculated by using the full-length window algorithms.

Wilson and MDCT bases are also handled by the DGT algorithm by the use of pre/post processing stages. This is very convenient, as it makes it possible to speed up the whole toolbox by optimising a single algorithm. For this reason, a C-implementation of the DGT algorithms is included in the toolbox, as well as interfaces for Octave and Matlab (Mex) to use this library. The library links to the efficient FFT implementation (FFTW[27]) included in both Octave and Matlab, and to the BlAS and LAPACK libraries for necessary linear algebra routines.

The instruction `ltfatmex` compiles the C code and generates corresponding `mex` files.

The discrete sine and cosine transforms are computed by the classic algorithms published in Ref. 58.

7. Perspectives and Outlook

In the following section we will describe some of the possible topics and features that the toolbox could include in future versions.

7.1. *Less structured time-frequency representations*

The time-frequency representations currently in the toolbox are generated from a single window, a fixed hop-size and a fixed number of channels. By allowing more general choices for these three parameters, it is possible to create frames and bases that can be better suited for specific applications.

As an example, if the window is allowed to depend on the channel, it is possible to create *uniform filter banks*, as described in Ref. 13. Such systems can be tailored to provide models of the human auditory system, where the frequency resolution (the width in frequency of each window) depends in a non-linear way on the centre frequency of the channels. Another interesting class of system is the *non-stationary Gabor frames*[31,6] , where the window, hop-size and number of channels may change with time. Such systems can be used for efficient models of music because the resolution can be adapted to be narrow in time for onsets and transient sounds and to be wide in time for harmonic sounds.

7.2. *Wavelets*

The biggest goal for future development of the toolbox is to include a self-contained implementation of various Wavelet transforms. A possible solution will be to integrate other existing open-source Wavelet toolboxes[30], and in this way ensure that they will continue to be developed and maintained.

7.3. *Structural improvements*

We would like to expand the processing capabilities of the toolbox by including block-processing methods. These are routines that allow standard transforms like the DGT to be applied to blocks of a signal in such a way that the output blocks can be assembled to generate the results. This way of processing is commonly used in systems that continuously generate and process data, like real-time applications.

Another computational improvement of the toolbox will be to implement parallel processing capabilities for the most demanding methods. The target for optimisation will be the standard computer architecture where 2-8 tightly integrated processors share a common memory subsystem.

7.4. *Various minor improvements*

In addition to extending the toolbox with more time-frequency transforms, a number of smaller additions are also considered. These includes the *Discrete Fractional Fourier transform*[44], more support for Hermite functions[41], detection of ridges in signals[15], better support for construction of non-canonical dual Gabor windows, Gabor analysis on non-rectangular grids[21], efficient computation of the S_0 norm and non-linear spectrogram reconstruction methods[28].

Acknowledgements

The authors would like to thank the people that made contributions to the toolbox: Nina Engelputzeder, Hans Feichtinger, Thomas Hrycak, Florent Jaillet, A.J.E.M. Janssen, Norbert Kaiblinger, Matthieu Kowalski, Ewa Matusiak, Piotr Majdak, Thomas Strohmer and Tobias Werther.

Bibliography

1. J. Allen. Short term spectral analysis, synthesis, and modification by discrete Fourier transform. *IEEE Trans. on Acoustics, Speech and Signal Processing*, ASSP-25(3):235–238, 1977.
2. F. Auger and P. Flandrin. Improving the readability of time-frequency and time-scale representations by the reassignment method. *IEEE Trans. Signal Process.*, 43(5):1068–1089, 1995.
3. P. Balazs. Basic definition and properties of Bessel multipliers. *Journal of Mathematical Analysis and Applications*, 325(1):571–585, January 2007.
4. P. Balazs. Hilbert-Schmidt operators and frames - classification, best approximation by multipliers and algorithms. *International Journal of Wavelets, Multiresolution and Information Processing*, 6(2):315 – 330, March 2008.
5. P. Balazs, J. Antoine, and A. Grybos. Weighted and Controlled Frames: Mutual Relationship and First Numerical Properties. *Int. J. Wavelets Multi.*, 8(1):109, 2010.
6. P. Balazs, M. Dörfler, N. Holighaus, F. Jaillet, and G. Velasco. Theory, Implementation and Application of Nonstationary Gabor Frames. *Journal of Computational and Applied Mathematics*, 236(6):1481–1496, 2011.
7. P. Balazs and N. Engelputzeder. Representation of linear time invariant filters by gabor multipliers. *in preparation*, -:–, 2010.
8. P. Balazs, H. G. Feichtinger, M. Hampejs, and G. Kracher. Double preconditioning for Gabor frames. *IEEE Trans. Signal Process.*, 54(12):4597–4610, December 2006.
9. P. Balazs, B. Laback, G. Eckel, and W. A. Deutsch. Time-frequency sparsity by removing perceptually irrelevant components using a simple model of simultaneous masking. *IEEE Transactions on Audio, Speech and Language Processing*, 18(1):34–49, 2010.
10. H. Bölcskei. Orthogonal frequency division multiplexing based on offset QAM. In Feichtinger and Strohmer [24], chapter 12, pages 321–352.
11. H. Bölcskei, H. G. Feichtinger, K. Gröchenig, and F. Hlawatsch. Discrete-time Wilson expansions. In *Proc. IEEE-SP 1996 Int. Sympos. Time-Frequency Time-Scale Analysis*, june 1996.
12. H. Bölcskei, F. Hlawatsch, and H. G. Feichtinger. Equivalence of DFT filter banks and Gabor expansions. In *SPIE 95, Wavelet Applications in Signal and Image Processing III*, volume 2569, part I, San Diego, july 1995.
13. H. Bölcskei, F. Hlawatsch, and H. G. Feichtinger. Frame-theoretic analysis of oversampled filter banks. *Signal Processing, IEEE Transactions on*, 46(12):3256–3268, 2002.
14. R. Carmona, W. Hwang, and B. Torrésani. *Practical Time-Frequency Analysis: continuous wavelet and Gabor transforms, with an implementation in S*, volume 9 of *Wavelet Analysis and its Applications*. Academic Press, San Diego, 1998.
15. R. Carmona, W. Hwang, and B. Torrésani. Multiridge detection and time-frequency reconstruction. *IEEE Trans. Signal Process.*, 47:480–492, 1999.
16. O. Christensen. *An Introduction to Frames and Riesz Bases*. Birkhäuser, 2003.
17. I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied*

Mathematics, 57(11):1413 – 1457, August 2004.
18. M. Dolson. The phase vocoder: a tutorial. *Computer Musical Journal*, 10(4):11–27, 1986.
19. D. L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41:613–627, May 1995.
20. M. Dörfler and B. Torrésani. Representation of operators in the time-frequency domain and generalized Gabor multipliers. *J. Fourier Anal. Appl.*, 16(2):261–293, April 2010.
21. H. Feichtinger, M. Hazewinkel, N. Kaiblinger, E. Matusiak, and M. Neuhauser. Metaplectic Operators on $\mathbb{C}^n$. *The Quarterly Journal of Mathematics*, 59(1):15, 2008.
22. H. G. Feichtinger and K. Nowak. A first survey of Gabor multipliers. In Feichtinger and Strohmer [24], chapter 5, pages 99–128.
23. H. G. Feichtinger and T. Strohmer, editors. *Gabor Analysis and Algorithms*. Birkhäuser, Boston, 1998.
24. H. G. Feichtinger and T. Strohmer, editors. *Advances in Gabor Analysis*. Birkhäuser, 2003.
25. J. Flanagan, D. Meinhart, R. Golden, and M. Sondhi. Phase Vocoder. *The Journal of the Acoustical Society of America*, 38:939, 1965.
26. P. Flandrin. *Time-frequency/time-scale analysis*, volume 10 of *Wavelet Analysis and its Applications*. Academic Press Inc., San Diego, CA, 1999. With a preface by Yves Meyer, Translated from the French by Joachim Stöckler.
27. M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".
28. D. Griffin and J. Lim. Signal estimation from modified short-time Fourier transform. 32(2):236–243, 1984.
29. K. Gröchenig. *Foundations of Time-Frequency Analysis*. Birkhäuser, 2001.
30. L. Jacques, A. Coron, P. Vandergheynst, and A. Rivoldini. The YAWTb toolbox : Yet Another Wavelet Toolbox, 2007. http://rhea.tele.ucl.ac.be/yawtb.
31. F. Jaillet, P. Balazs, M. Dörfler, and N. Engelputzeder. Nonstationary Gabor frames. In *SAMPTA'09, Marseille, May 18-22*, 2009.
32. A. J. E. M. Janssen and P. L. Søndergaard. Iterative algorithms to approximate canonical Gabor windows: Computational aspects. *J. Fourier Anal. Appl.*, 13:211–241, 2007.
33. N. S. Jayant and P. Noll. *Digital coding of waveforms*. Prentice-Hall, 1984.
34. M. Kowalski. Sparse regression using mixed norms. *Applied and Computational Harmonic Analysis*, 27(3):303–324, 2009.
35. M. Kowalski and B. Torrésani. Sparsity and persistence: mixed norms provide simple signal models with dependent coefficients. *Signal, Image and Video Processing*, 3(3):251–264, September 2009. doi: 10.1007/s11760-008-0076-1.
36. Y.-P. Lin and P. Vaidyanathan. Linear phase cosine modulated maximally decimated filter banks with perfectreconstruction. *IEEE Trans. Signal Process.*, 43(11):2525–2539, 1995.
37. P. Majdak, P. Balazs, W. Kreuzer, and M. Dörfler. A time-frequency method for increasing the signal-to-noise ratio in system identification with exponential sweeps. In *Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, 2011.
38. P. Majdak, P. Balazs, and B. Laback. Multiple exponential sweep method for fast measurement of head related transfer functions. *Journal of the Audio Engineering Society*, 55(7/8):623–637, July/August 2007.
39. H. S. Malvar. *Signal Processing with Lapped Transforms*. Artech House Publishers, 1992.

40. G. Matz and F. Hlawatsch. *Linear Time-Frequency Filters: On-line Algorithms and Applications*, chapter 6 in 'Application in Time-Frequency Signal Processing', pages 205–271. eds. A. Papandreou-Suppappola, Boca Raton (FL): CRC Press, 2002.
41. D. Mugler, S. Clary, and Y. Wu. Discrete Hermite expansion of digital signals: applications to ECG signals. *Digital Signal Processing Workshop, 2002 and the 2nd Signal Processing Education Workshop. Proceedings of 2002 IEEE 10th*, pages 262–267, 2002.
42. S. Nadarajah. GG DCT Coefficient Models. *Int. J. Wavelets Multi.*, 8(05):793–812, 2010.
43. A. V. Oppenheim and R. W. Schafer. *Discrete-time signal processing.* Prentice Hall, Englewood Cliffs, NJ, 1989.
44. H. M. Ozaktas, Z. Zalevsky, and M. A. Kutay. *The Fractional Fourier Transform.* John Wiley and Sons, 2001.
45. J. P. Princen and A. B. Bradley. Analysis/synthesis filter bank design based on time domain aliasing cancellation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-34(5):1153–1161, 1986.
46. J. P. Princen, A. W. Johnson, and A. B. Bradley. Subband/transform coding using filter bank designs based on time domain aliasing cancellation. *Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2161–2164, 1987.
47. P. Prinz. Calculating the dual Gabor window for general sampling sets. *IEEE Trans. Signal Process.*, 44(8):2078–2082, 1996.
48. K. Rao and P. Yip. *Discrete Cosine Transform, Algorithms, Advantages, Applications.* Academic Press, 1990.
49. P. L. Søndergaard. *Finite Discrete Gabor Analysis.* PhD thesis, 2007.
50. P. L. Søndergaard. Symmetric, discrete fractional splines and Gabor systems. *preprint*, 2008.
51. P. L. Søndergaard. Efficient Algorithms for the Discrete Gabor Transform with a long FIR window. *J. Fourier Anal. Appl.*, 18(3):456–470, 2012.
52. J. Strawn (ed.). *Digital Audio Signal Processing - An Anthology.* William Kaufmann, Inc., 1985.
53. T. Strohmer. Numerical algorithms for discrete Gabor expansions. In Feichtinger and Strohmer [23], chapter 8, pages 267–294.
54. T. Strohmer. Approximation of dual Gabor frames, window decay, and wireless co mmunications. *Appl. Comput. Harmon. Anal.*, 11(2):243–262, 2001.
55. P. Tseng. Dual coordinate ascent methods for non-stricly convex minimization. *Mathematical Programming*, 59:231–247, 1993.
56. M. Vetterli and J. Kovačević. *Wavelets and Subband Coding.* Signal Processing Series. Prentice Hall, Englewood Cliffs, NJ, 1995.
57. E. Wesfreid and M. Wickerhauser. Adapted local trigonometric transforms and speech processing. *IEEE Trans. Signal Process.*, 41(12):3596–3600, 1993.
58. M. V. Wickerhauser. *Adapted wavelet analysis from theory to software.* Wellesley-Cambridge Press, Wellesley, MA, 1994.
59. B. Widrow and S. D. Stearns. *Adaptive Signal Processing.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.
60. Wikipedia. List of audio codecs, 2007.
61. Wikipedia. Noise reduction, 2010.
62. P. Wojdłło. Wilson System for Triple Redundancy. *Int. J. Wavelets Multi.*, 9(01):151–167, 2011.
63. O. Yilmaz and S. Rickard. Blind separation of speech mixtures via time-frequency masking. *IEEE Transactions on Signal Processing*, 52(7):1830–1847, July 2004.