JACOBS
UNIVERSITY

Niklas Grip and Götz E. Pfander

Time Varying Narrowband Communications Channels: Analysis and Implementation

Technical Report No. 12
October 2007

School of Engineering and Science

Time Varying Narrowband Communications Channels: Analysis and Implementation[1]

Niklas Grip

Department of Mathematics
Luleå University of Technology
SE-971 87 Luleå
Sweden

Götz E. Pfander

School of Engineering and Science
Jacobs University Bremen
Campus Ring 12
28759 Bremen
Germany

E-Mail: grip@sm.luth.se, g.pfander@jacobs-university.de,
http://math.jacobs-university.de/pfander

Summary

We derive and describe a MATLAB implementation of an efficient numerical algorithm for the analysis of certain classes of Hilbert–Schmidt operators that naturally occur in models of wireless radio and sonar communications channels.

A common short-time model of these channels writes the channel output as a weighted superposition of time- and frequency shifted copies of the transmitted signal, where the weight function is often called the *spreading function* of the channel operator.

It is often believed that a good channel model must allow for spreading functions containing Dirac delta distributions. However, we show that many narrowband finite lifelength channels such as wireless radio communications can be well modelled by smooth and compactly supported spreading functions.

We derive a fast algorithm for computing the matrix representation of such operators with respect to well time-frequency localized Gabor bases, such as pulse shaped OFDM bases. Hereby we use a minimum of approximations, simplifications, and assumptions on the channel. The primary intended target application

[1]The work on this project was supported by the German Research Foundation (DFG) Project PF 450/1-1 as part of the DFG priority program TakeOFDM. During the final preparations of the manuscript, the first mentioned author was supported by the Swedish Research Council, project registration number 2004-3862.

is comparisons of how different parameter settings and pulse shapes affect the diagonalization properties of an OFDM system acting on a given channel.

Finally, we provide and demonstrate the use of a MATLAB implementation that is fast enough for the number of carrier frequencies typically in use today.

A shortened and improved version of the main results in this report is published in the journal paper [GP07], which excludes some discussions, proofs and the software documentation contained here.

Contents

Contents

List of Figures

1 Introduction

Channel–dependent customization is expected to provide considerable performance improvements in time-varying systems such as future generations of wireless communications systems. Consequently, the idea of shaping the transmission pulses in order to minimize the InterCarrier and InterSymbol Interference (ISI and ICI) in Orthogonal Frequency Division Multiplexing (OFDM) communications is an active research area in the applied harmonic analysis and signal processing communities (see [BS01, MSG+05] and references therein). Even though some insights can be gained from careful mathematical modelling and analysis, there remains a need for fast algorithms and implementations aimed at the numerical evaluation of performance improvements through pulse shaping. In this report, we discuss two closely connected topics that we regard of vital importance to fulfill this demand.

1. We review the most important physical properties of wireless channels and show how these lead naturally to a model of the short-time behavior of a channel as an operator H that maps an input signal s to a weighted superposition of time and frequency shifts of s, that is,

$$Hs(\cdot) = \int_{K\times[A,\infty)} S_H(\nu,t)\, \mathrm{e}^{i2\pi\nu(t-t_0)} s(\,\cdot\, - t)\, \mathrm{d}(\nu,t)\,, \qquad K \text{ compact.} \quad (1.1)$$

This model is well-known and the coefficient function S_H is usually called the *spreading function* of H. The model given by (1.1) is mostly used either under the assumption that S_H is square-integrable or that S_H is a tempered distribution. The latter, weaker assumption suggests that the input signal s should be a Schwartz class function and requires the use of distribution theory in the analysis of H, both of which we shall try to avoid. Therefore, we derive (1.1) using some refinements of the standard multipath propagation model of the channel. Our analysis implies that the short-time behaviour in many communications applications can be completely described by a smooth S_H with rapid decay ensuring "essentially compact" support[2]. This model has the big advantage that it allows for *both* Fourier analysis and numerical evaluation of the performance of OFDM procedures without the need of deviating into distribution theory.

2. We employ the channel model described above to derive an efficient algorithm for the numerical evaluation of ISI and ICI in pulse shaped OFDM systems. Although we, for simplicity, justify our channel *model* only for signals $s \in L^2(\mathbb{R})$, we derive our *algorithm* for a straightforward extension to functions $s \in L^2(\mathbb{R}^d)$. We expect this extension to be useful, for example, for combining measurements from several antennas to a large set of samples

[2]With *essentially compact* we mean that the function decays fast enough to assure that in any practical application, the function values outside some "reasonably small" compact set are very small compared to the overall noise level and therefore negligible (see also Section 6.1).

$s(\boldsymbol{x}_n, t_k)$ of s in an (irregular) sampling grid of points $\boldsymbol{x}_n$ in space and t_k in time, as is sometimes done in radio astronomy. Similarly for wireless communications, both on the transmitter and receiver side, *space diversity* is one of the most popular forms of diversity [Rap02]. The main idea there is to use antenna arrays with antennas preferably placed several tens of wavelengths apart for improving the chance to always have good transmission between at least two of the antennas (see, e.g., [Rap02, BAB+01, JYGR74] for a more complete coverage).

We shall now motivate and describe the principles of our discretization in some detail:

For multicarrier modulation systems in general, the aim is the joint diagonalization of a class of possible channel operators in a given environment. That is, we try to find a transmission basis (g_i) and a receiver basis (filters) $(\widetilde{\gamma}_j)$ with the property that all coefficient mappings that correspond to channels in the environment have matrix representations $\mathrm{G}_{i,j} = \langle Hg_i, \widetilde{\gamma}_j \rangle$ that are as close to diagonal as possible, that is, $|\mathrm{G}_{i,j}|$ decays fast with $|i-j|$. In general, an easily computable inverse of this coefficient mapping would allow us to regain the transmitted coefficients (c_i) in the input signal $s = \sum c_i g_i$, and, therefore, the information embedded in these coefficients, from the inner products $\langle Hs, \widetilde{\gamma}_i \rangle$ which are calculated on the receiver side.

In wireline communications, the problem described above has a well accepted solution, namely OFDM (also called Discrete MultiTone or DMT) with cyclic prefix. Here, the transmission basis (g_i) and the receiver basis (γ_i) are so-called Gabor bases, that is, each basis consists of time and frequency shifts of a single prototype function which is often referred to as window function. Diagonalization of the channel operator using Gabor bases with rectangular prototype function is then possible since wired channels are assumed to be time-invariant. This allows us to model such channel operators as convolution operators with complex exponentials $\mathrm{e}^{\mathrm{i}2\pi\omega t}$ as "eigenfunctions". This cyclic prefix procedure applies if the channel has finite lifelength and is explained in more detail in Section 4.1 and with further references in [Gri02]. The superiority of Gabor bases in comparison to Wavelet and Wilson bases for wireline communications is examined in detail in [KPZ02].

Wireless channels are inherently time-varying. The generality of time-varying channel operators and, in particular, the fact that they do not commute in general, implies that joint diagonalization of classes of such channels cannot be achieved as in the general case, so approximate diagonalization becomes our goal. In many cases, for example in mobile telephony, the channel varies only "slowly" with time. Hence, we use the results for time-invariant channels as a starting point and consider in this report only the use of Gabor bases as transmitter and receiver bases.

For such slowly time-varying systems, Matz, Schafhuber, Gröchenig, Hartmann and Hlawatsch conclude that excellent joint time-frequency concentration of the windows g and γ is the most important requirement for low ISI and ICI [MSG+05].

There, it is shown how to compute a γ (or an orthogonalization of the basis (g_i)) that diagonalizes the coefficient mapping in the idealized borderline case when the channel is the identity operator ($\mathrm{G}_{i,j} = \delta_{i,j}$). They show that both γ and the corresponding orthogonalized basis inherit certain polynomial or subexponential time-frequency decay properties from g. They also derive exact and approximate expressions for the ISI and ICI and present an efficient FFT-based modulator and demodulator implementation.

For multicarrier systems with excellent joint time-frequency localization of g and γ, we derive, starting from our channel model, a procedure for the *numerical* computation of the matrix entries $\mathrm{G}_{i,j} = \langle Hg_i, \widetilde{\gamma}_j \rangle$ under a minimum of assumptions, simplifications, or approximations. These properties make our approach different from and complementary to a number of papers that use *discrete* Gabor bases (sometimes under the name BEM or Basis Expansion Model) for time-varying channels and statistical applications, such as [BLM04, BLM05, LM04, LZGk03, MGk02, MGk03b, MGk03a, MLGk05, SA99, TL04].

The paper is organized as follows: The Notation and some mathematical preliminaries are described in *Section 2*. We derive a channel model in *Section 3*, and use it to derive formulas for the matrix elements in *Section 4*. In *Section 5* we describe a MATLAB implementation of these formulas and give suggestions on how to do the necessary parameter and window/pulse shape choices. In *Section 6* we provide typical system-dependent parameters for and demonstrate our software on some example mobile phone communications, satellite communications and underwater sonar communications applications. Finally, our conclusions follow in *Section 7*.

2 Preliminaries

For completeness and easy availability we collect our notation in Section 2.1 and give an overview of the mathematical tools that we shall use. In Section 2.2 we shall discuss the availability of functions that are compactly supported and "essentially bandlimited", in particular, we explain how compactly supported functions can be designed to have subexponential decay. Section 2.3 covers the Gabor system expansions which are used to obtain diagonal dominant coefficient mappings of channel operators. Finally, in Section 2.4 we discuss the Hilbert–Schmidt operator theory and the integral representation of channel operators in terms of system functions such as the spreading function and the time-varying impulse response.

2.1 Notation

We assume the reader to know some basic tools and notation from functional analysis and measure theory, which otherwise can be found in [Fol99, Rud87].

The conjugate of a complex number z is denoted $\overline{z}$. We use boldface font for elements in $\mathbb{R}^d$, write $\mathbb{R}^d_+ \stackrel{\text{def}}{=} (0,\infty)^d \stackrel{\text{def}}{=} \mathbb{R}_+ \times \mathbb{R}_+ \times \cdots \times \mathbb{R}_+$ and $\mathbb{Z}^d_+ \stackrel{\text{def}}{=} \mathbb{Z}^d \cap \mathbb{R}^d_+$. The

Fourier transform of a function f is formally given by $\widehat{f}(\boldsymbol{\xi}) = \int_{\mathbb{R}^d} f(\boldsymbol{t})\mathrm{e}^{-i2\pi\langle\boldsymbol{\xi},\boldsymbol{t}\rangle}\,\mathrm{d}\boldsymbol{t}$ for $\boldsymbol{\xi} \in \mathbb{R}^d$ and $l^2 \stackrel{\text{def}}{=} l^2(\mathbb{Z}^d \times \mathbb{Z}^d)$ is the Hilbert space of sequences $(c_{\boldsymbol{q},\boldsymbol{r}})$ for which the l^2-norm is given by

$$\|(c_{\boldsymbol{q},\boldsymbol{r}})\|_2 \stackrel{\text{def}}{=} \left(\sum_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d} |c_{\boldsymbol{q},\boldsymbol{r}}|^2\right)^{1/2} < \infty.$$

Throughout the paper we use Roman and Greek letters for variables that have a physical interpretation as time or spacial variable and frequency, respectively. For $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}, \boldsymbol{\nu}, \boldsymbol{\omega} \in \mathbb{R}^d$ and $r \in \mathbb{R}$ we use the following shorthand notation:

$$\begin{aligned}
[\boldsymbol{A},\boldsymbol{B}] &\stackrel{\text{def}}{=} [A_1,B_1]\times\cdots\times[A_d,B_d], & \mathbf{1} &\stackrel{\text{def}}{=} \begin{pmatrix}1 & 1 & \cdots & 1\end{pmatrix}^T,\\
\langle\boldsymbol{x},\boldsymbol{y}\rangle &\stackrel{\text{def}}{=} x_1y_1 + x_2y_2 + \cdots + x_dy_d, & \boldsymbol{xy} &\stackrel{\text{def}}{=} \begin{pmatrix}x_1y_1 & \cdots & x_dy_d\end{pmatrix}^T\\
\frac{\boldsymbol{x}}{r} &\stackrel{\text{def}}{=} \begin{pmatrix}\frac{x_1}{r} & \frac{x_2}{r} & \cdots & \frac{x_d}{r}\end{pmatrix}^T, & \boldsymbol{x}^r &\stackrel{\text{def}}{=} \begin{pmatrix}x_1^r & x_2^r & \cdots & x_d^r\end{pmatrix}^T,\\
\frac{\boldsymbol{x}}{\boldsymbol{y}} &\stackrel{\text{def}}{=} \begin{pmatrix}\frac{x_1}{y_1} & \frac{x_2}{y_2} & \cdots & \frac{x_d}{y_d}\end{pmatrix}^T, & |\boldsymbol{x}| &\stackrel{\text{def}}{=} |x_1x_2\cdots x_d|,\\
T_{\boldsymbol{t}}g &\stackrel{\text{def}}{=} g(\cdot-\boldsymbol{t}), & M_{\boldsymbol{\nu}}g &\stackrel{\text{def}}{=} g(\cdot)\mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\nu},\cdot\rangle},\\
I_{\boldsymbol{C},\boldsymbol{B}} &\stackrel{\text{def}}{=} \left[\boldsymbol{C}-\frac{\boldsymbol{B}}{\boldsymbol{2}},\boldsymbol{C}+\frac{\boldsymbol{B}}{\boldsymbol{2}}\right] \quad \text{and} & \operatorname{sinc}_{\boldsymbol{\omega}}(\boldsymbol{x}) &\stackrel{\text{def}}{=} \prod_{j=1}^{d}\frac{\sin(\pi\omega_jx_j)}{\pi x_j}.
\end{aligned}$$

Here, $\operatorname{sinc}_{\boldsymbol{\omega}}$ is extended continuously to $\mathbb{R}^d$ and we shall frequently use that

$$\int_{I_{\boldsymbol{C},\boldsymbol{B}}} \mathrm{e}^{-\mathrm{i}2\pi\langle\boldsymbol{\xi},\boldsymbol{x}\rangle}\,\mathrm{d}\boldsymbol{\xi} = \mathrm{e}^{-\mathrm{i}2\pi\langle\boldsymbol{C},\boldsymbol{x}\rangle}\operatorname{sinc}_{\boldsymbol{B}}(\boldsymbol{x}). \tag{2.1}$$

For $\epsilon > 0$ we define the *ϵ-essential support* of a bounded function $f\colon \mathbb{R}^d \to \mathbb{C}$ to be the closure of the set $\{\boldsymbol{x} : \epsilon \le |f(\boldsymbol{x})| \,/\, \sup_{\boldsymbol{x}} |f(\boldsymbol{x})|\}$. For an almost everywhere defined function f, $\operatorname{supp} f$ denotes the intersection of the supports of all representatives of f (and similarly for ϵ-essential support). For any set I, χ_I is the characteristic function $\chi_I(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in I$ and $\chi_I(\boldsymbol{x}) = 0$ otherwise. The sets of n times, respectively infinitely many, times continuously differentiable functions are denoted $C^{(n)}$ and $C^{(\infty)}$, respectively.

We denote by $L^p = L^p(\mathbb{R}^d)$ the Banach space of complex-valued measurable functions f with norm

$$\|f\|_p \stackrel{\text{def}}{=} \left(\int_{\mathbb{R}^d} |f(\boldsymbol{x})|^p\,\mathrm{d}\boldsymbol{x}\right)^{1/p} < \infty.$$

$L^2(\mathbb{R}^d)$ is a Hilbert space with inner product $\langle f,g\rangle \stackrel{\text{def}}{=} \int_{\mathbb{R}^d} f(\boldsymbol{x})\overline{g(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x}$. We say that two sequences $(f_{\boldsymbol{n}})$ and $(g_{\boldsymbol{n}})$ of functions are *biorthogonal* if $\langle f_{\boldsymbol{m}}, g_{\boldsymbol{n}}\rangle = 0$ whenever $\boldsymbol{m} \neq \boldsymbol{n}$ and $\langle f_{\boldsymbol{n}}, g_{\boldsymbol{n}}\rangle = 1$ for all $\boldsymbol{n}$. The Wiener amalgam space

$W(A, l^1) = S_0(\mathbb{R}^d)$ (also named the Feichtinger algebra) consists of the set of all continuous $f\colon \mathbb{R}^d \to \mathbb{C}$ for which

$$\sum_{\boldsymbol{n}\in\mathbb{Z}^d} \|(f(\cdot)\psi(\cdot - \boldsymbol{n}))\widehat{\ }\|_1 < \infty$$

for some compactly supported ψ such that $\widehat{\psi} \in L^1(\mathbb{R}^d)$ and $\sum_{n\in\mathbb{Z}^d} \psi(\boldsymbol{x} - \boldsymbol{n}) = 1$. We write S_0' for the space of linear bounded functionals on S_0. S_0 is also a so-called modulation space, described at more depth and with notation $S_0 = M^{1,1} = M^1$ and $S_0' = M^{\infty,\infty} = M^\infty$ in [Grö00, FZ98].

A real-valued, measurable and locally bounded function w on $\mathbb{R}^d$ is said to be a *weight function* if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,

$$w(\boldsymbol{x}) \geq 1 \qquad \text{and} \qquad w(\boldsymbol{x} + \boldsymbol{y}) \leq w(\boldsymbol{x})w(\boldsymbol{y}). \tag{2.2}$$

For weight functions w we define $L^1_w = L^1_w(\mathbb{R}^d)$ to be the family of functions $f \in L^1(\mathbb{R}^d)$ such that

$$\|f\|_{1,w} \stackrel{\text{def}}{=} \|fw\|_1 < \infty.$$

2.2 Frequency localization of compactly supported functions

The Gabor window g in the introduction needs to be compactly supported in a time interval short enough to satisfy typical maximum delay restrictions, such as 25 ms for voice communications. Moreover, its Fourier transform $\widehat{g}$ has to decay fast enough to allow for reasonably high transmission power (which determines the signal-to-noise ratio) without exceeding standard regulations on the allowed power leakage into other frequency bands. In other words, g should have good, joint time-frequency localization, which also is of great importance for achieving low ISI and ICI [MSG$^+$05]. For this reason, we seek to know to what extent compact support of a function can be combined with good decay of its Fourier transform. This classical question was first answered by Beurling [Beu38, Theorem V B] and generalized from functions on $\mathbb{R}$ to functions on locally compact abelian groups, such as $\mathbb{R}^d$, by Domar [Dom56, Theorem 2.11]. Domar's results are explained in much more detail in [RS00, Ch. 6.3 + appendices]. We will describe a partial result that describes the speed of decay of a Fourier transform by specifying for how fast growing weight functions w it belongs to $L^1_w(\mathbb{R})$. For describing the asymptotic decay, we only need to consider continuous w with nondecreasing $w(\xi)$ and $w(-\xi)$ for positive ξ. Suppose for one such w that there is some compactly supported $f \in L^2(\mathbb{R})$ such that $\widehat{f} \in L^1_w(\mathbb{R})$, then for *any* open set $U \in \mathbb{R}$ we can choose a dilation and translation $f(a \cdot -x)$ of f with support in U. Since also $\widehat{f(a \cdot -x)} \in L^1_w(\mathbb{R})$, the following theorem then provides the so-called *logarithmic integral condition* (2.3) on the decay of w:

Theorem 1. *Let w be a continuous weight function such that $w(\xi)$ and $w(-\xi)$ are nondecreasing for positive ξ. Suppose that for every open set $U \subseteq \mathbb{R}$ there is a non-zero function $f \in L^2(\mathbb{R})$ such that* $\operatorname{supp} f \subseteq U$ *and $\widehat{f} \in L^1_w(\mathbb{R})$. Then*

$$\int_{\mathbb{R}} \frac{\log(w(\xi))}{1+\xi^2}\, d\xi < \infty. \tag{2.3}$$

Proof. This theorem was proved in [RS00, Theorem A.1.13] for functions defined on locally compact abelian groups, without our assumption about continuous and nondecreasing $w(\pm\xi)$ and with (2.3) replaced by the conclusion

$$\sum_{n\in\mathbb{Z}} \frac{\log(w(n\nu))}{1+n^2} < \infty \qquad \text{for all} \qquad \nu \in \mathbb{R}. \tag{2.4}$$

However, by our additional assumptions on w, the sum criterion for integrals implies that (2.3) holds if and only if (2.4) holds for $\nu = 1$. This concludes the proof. □

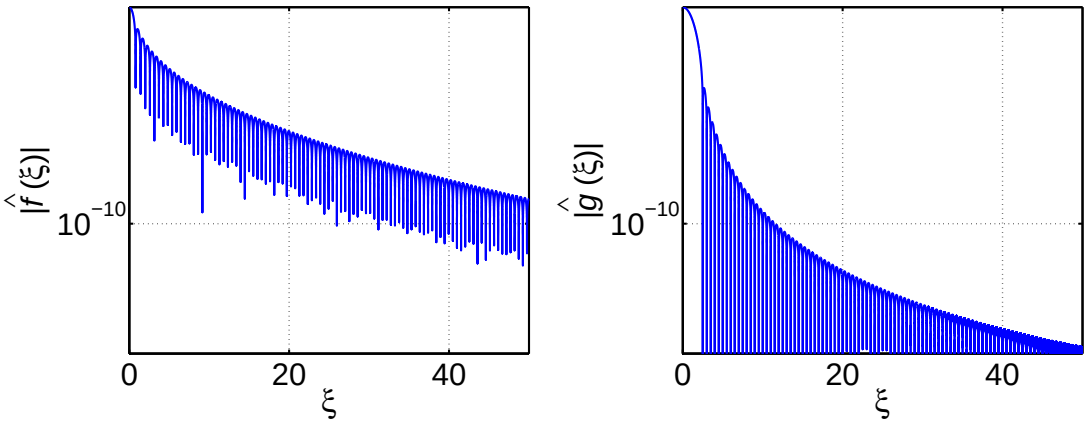
The logarithmic integral condition (2.3) limits the decay of *both* the amplitude and "the area under the tail" of $\widehat{f}$. For example, the Fourier transform of a compactly supported function f cannot be either $\mathcal{O}(\mathrm{e}^{-\alpha|\xi|})$ nor $\widehat{f}(\xi) = \sum_{n\in\mathbb{Z}} \phi(\mathrm{e}^{\alpha|n|}(\xi - n))$ for any $\phi \in C^{(\infty)}$ with support $\operatorname{supp}\phi \subseteq [0,1]$, because in both cases, $\widehat{f} \in L^1_w(\mathbb{R})$ for $w(\xi) = \mathrm{e}^{a|\xi|}$ and $a < \alpha$ but w does not satisfy (2.3). This fact rules out the existence of compactly supported functions f with exponentially decaying $\widehat{f}$. However, Dziubański and Hernández [DH98] have shown how to use a construction by Hörmander [Hör03, Theorem 1.3.5] to construct a compactly supported function f whose Fourier transform is *subexponentially decaying*. That is, they construct f such that for every $0 < \varepsilon < 1$ there exists $C_\varepsilon > 0$ such that

$$\left|\widehat{f}(\xi)\right| \le C_\varepsilon \mathrm{e}^{-|\xi|^{1-\varepsilon}}, \qquad \forall \xi \in \mathbb{R}.$$

From their example and standard techniques such as convolution with a characteristic function, it is then easy to design for any compact set K a compactly supported function f such that $f(x) = 1$ for $x \in K$, and $\widehat{f}$ is subexponentially decaying.

Note however, that subexponential decay is not everything. For example, the function $f(x) = \mathrm{e}^{-\frac{1}{1-x^2}}\chi_{[-1,1]}(x)$ is a compactly supported $C^{(\infty)}$ function, so that $\widehat{f}(\xi) = \mathcal{O}(1+|\xi|)^{-n}$ for all $n \in \mathbb{N}$, whereas the function $g(x) = (1+\cos(\pi x))^4\chi_{[-1,1]}(x)$ has Fourier transform decay $\widehat{g}(\xi) = \mathcal{O}((1+|\xi|)^{-\alpha})$ for $\alpha = 9$ but not for $\alpha > 9$ (see Corollary 2, page 44). However, Figure 1 shows that $\widehat{g}$ decays much faster down to amplitude thresholds such as the power leakage restrictions described above (see also Figure 7, page 36). Thus it can be an important design issue to choose functions and forms of decay that are optimal for a given application.

However, for simplicity and a clear presentation in this report, we shall consistently claim subexponential decay although also other forms of decay are rapid enough for all of our results to hold.



(a) (b)

Figure 1: The Fourier transform decay after normalizing the following positive functions to have integral 1: (a) $f(x) \stackrel{\text{def}}{=} e^{-\frac{1}{1-x^2}}\chi_{[-1,1]}(x)$. (b) $g(x) = (1+\cos(\pi x))^4\chi_{[-1,1]}(x)$.

2.3 Gabor analysis

Here, we give a brief review of some basic Gabor frame theory that is needed to understand the relevance of the coefficient mappings that we introduce in (2.7) below. For a more complete and general coverage of this subject, see, for example, [Chr02, Gri02, Grö00].

A *Gabor* (or *Weyl-Heisenberg*) system with *window* g and *lattice constants* $\boldsymbol{a}$ and $\boldsymbol{b}$ is the sequence $(g_{\boldsymbol{q},\boldsymbol{r}})_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d}$ of translated and modulated functions

$$g_{\boldsymbol{q},\boldsymbol{r}} \stackrel{\text{def}}{=} T_{\boldsymbol{r}\boldsymbol{a}}M_{\boldsymbol{q}\boldsymbol{b}}g = \mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{q}\boldsymbol{b},\boldsymbol{x}-\boldsymbol{r}\boldsymbol{a}\rangle}g(\boldsymbol{x}-\boldsymbol{r}\boldsymbol{a}).$$

The corresponding *synthesis* or *reconstruction operator*

$$R_g\colon l^2 \to L^2, \qquad R_g c \stackrel{\text{def}}{=} \sum_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d} c_{\boldsymbol{q},\boldsymbol{r}}g_{\boldsymbol{q},\boldsymbol{r}}$$

is defined with convergence in the L^2-norm if and only if its adjoint, the so-called *analysis operator*

$$R_g^*\colon L^2 \to l^2, \qquad R_g^* f = \left(\langle f, g_{\boldsymbol{q},\boldsymbol{r}}\rangle\right)_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d},$$

is bounded, i.e., if and only if $\sum_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d}|\langle f, g_{\boldsymbol{q},\boldsymbol{r}}\rangle|^2 \le B\,\|f\|_2^2$ for some $B\in\mathbb{R}_+$ and all $f\in L^2(\mathbb{R}^d)$ [Gri02, p. 14].

We call $(g_{\boldsymbol{q},\boldsymbol{r}})_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d}$ a Gabor *frame* for $L^2(\mathbb{R}^d)$ if there are *frame bounds* $A, B \in \mathbb{R}_+$ such that for all $f\in L^2(\mathbb{R}^d)$,

$$A\,\|f\|_2^2 \le \left\|R_g^* f\right\|_2^2 \le B\,\|f\|_2^2. \tag{2.5}$$

It follows from (2.5) that the *frame operator* $S_g \stackrel{\text{def}}{=} R_g R_g^*$ is invertible. We call a frame with elements $\widetilde{g}_{\boldsymbol{q},\boldsymbol{r}} \stackrel{\text{def}}{=} T_{\boldsymbol{ra}} M_{\boldsymbol{qb}} \widetilde{g}$ a *dual Gabor frame* if for every $f \in L^2$,

$$f = \sum_{\boldsymbol{q},\boldsymbol{r} \in \mathbb{Z}^d} \langle f, \widetilde{g}_{\boldsymbol{q},\boldsymbol{r}} \rangle \, g_{\boldsymbol{q},\boldsymbol{r}} = \sum_{\boldsymbol{q},\boldsymbol{r} \in \mathbb{Z}^d} \langle f, g_{\boldsymbol{q},\boldsymbol{r}} \rangle \, \widetilde{g}_{\boldsymbol{q},\boldsymbol{r}} \tag{2.6}$$

with L^2-norm convergence of both series. There may exist (infinitely) many different dual windows $\widetilde{g}$ for g. However, we shall always consider the *canonical dual window*, which is the minimum L^2–norm dual window [Jan98, p. 51]. The dual frame has frame bounds A^{-1}, B^{-1} and the coefficients in (2.6) are not unique in l^2, but they are the unique minimum l^2-norm coefficients. It follows also from (2.5) and (2.6) that $R_{\widetilde{g}}^*$ picks coefficients from $C_{\widetilde{g}} \stackrel{\text{def}}{=} R_{\widetilde{g}}^*(L^2(\mathbb{R}^d)) \subseteq l^2$ and that R_g is a bounded invertible mapping of $C_{\widetilde{g}}$ onto L^2 with bounded inverse $R_g^{-1} = R_{\widetilde{g}}^*$. By this isomorphism and the usual definition of operator norms we can use two Gabor frames $(g_{\boldsymbol{q},\boldsymbol{r}})$ and $(\gamma_{\boldsymbol{q},\boldsymbol{r}})$ (possibly with different lattice constants) to obtain an isomorphism of the family of linear bounded operators $H \colon L^2(\mathbb{R}^d) \to L^2(\mathbb{R}^d)$ with the coefficient mappings $G = R_\gamma^* H R_g$, as illustrated in the following commutative diagram.

$$\begin{array}{ccc} L^2(\mathbb{R}^d) & \xrightarrow{H} & L^2(\mathbb{R}^d) \\ \uparrow{\scriptstyle R_g} & & \downarrow{\scriptstyle R_\gamma^*} \\ C_{\widetilde{g}} & \xrightarrow{G = R_\gamma^* H R_g} & C_\gamma \end{array} \tag{2.7}$$

We will provide an explicit expression for G in Section 4.

The frame $(g_{\boldsymbol{q},\boldsymbol{r}})_{\boldsymbol{q},\boldsymbol{r} \in \mathbb{Z}^d}$ is called a *Riesz basis* if $C_{\widetilde{g}} = l^2$. Then, the coefficients in (2.6) are truly unique and, as a consequence, $(g_{\boldsymbol{q},\boldsymbol{r}})$ and $(\widetilde{g}_{\boldsymbol{q},\boldsymbol{r}})$ are biorthogonal.

2.4 Hilbert–Schmidt operators

The mathematical framework for the use of Hilbert–Schmidt operators acting on functions defined on locally compact abelian groups has been developed in great generality in harmonic and functional analysis [FK98]. For the basic theory, see, for example, [Con00, RS80] or [Fol95, Appendix 2], where the following classic definition is used.

Definition 1 (Hilbert–Schmidt operator)**.** Let $\mathcal{H}$ be a separable Hilbert space. A bounded operator $H \colon \mathcal{H} \to \mathcal{H}$ is called a *Hilbert–Schmidt operator* if $\sum_{n \in \mathbb{Z}} \|He_n\|_{\mathcal{H}}^2 < \infty$ for some orthogonal basis $(e_n)_{n \in \mathbb{Z}}$ of $\mathcal{H}$. The Hilbert–Schmidt norm of H is given by

$$\|H\|_{\text{H.S.}} = \left(\sum_{n \in \mathbb{Z}} \|He_n\|_{\mathcal{H}}^2 \right)^{\frac{1}{2}}.$$

This norm is independent of the choice of the orthonormal basis $(e_n)_{n\in\mathbb{Z}}$ [Con00, RS80] and can be calculated in a number of ways, one of which is given below.

Proposition 1 ([RS80, Theorem VI.23])**.** *Let $\{M, \mu\}$ be a measure space and $\mathcal{H} = L^2(M,\, d\mu)$. Then a linear bounded operator $H\colon \mathcal{H} \to \mathcal{H}$ is Hilbert–Schmidt if and only if there is a function*

$$\kappa \in L^2(M \times M, \mu \otimes \mu)$$

such that

$$(Hf)(x) = \int \kappa(x,y) f(y)\, d\mu(y) \quad \text{a.e. } x\,. \tag{2.8}$$

Moreover,

$$\|H\|^2_{\text{H.S.}} = \int |\kappa(x,y)|^2 \; d\mu\otimes\mu(x,y).$$

Within this report, we shall only consider Hilbert–Schmidt operators on the Hilbert space $\mathcal{H} = L^2(\mathbb{R}^d)$. There are many similar versions of Proposition 1, some of which can be obtained by applying partial unitary Fourier transforms to the function κ and replacing (2.8) with corresponding mappings relating f or $\widehat{f}$ to either Hf or $\widehat{Hf}$ as done in (2.11) below. For example, application of the leftmost unitary Fourier transform in (2.11a) gives the following corollary:

Corollary 1. *A linear bounded operator $H\colon L^2(\mathbb{R}^d) \to L^2(\mathbb{R}^d)$ is Hilbert–Schmidt if and only if there is a function $S_H \in L^2(\mathbb{R}^d \times \mathbb{R}^d)$ such that for all $s \in L^2(\mathbb{R}^d)$,*

$$(Hs)(\boldsymbol{t_0}) = \int_{\mathbb{R}^d\times\mathbb{R}^d} S_H(\boldsymbol{\nu},\boldsymbol{t}) s(\boldsymbol{t_0}-\boldsymbol{t}) \mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\nu},\boldsymbol{x}-\boldsymbol{t}\rangle}\, d(\boldsymbol{t},\boldsymbol{\nu}). \tag{2.9}$$

The integral in (2.9) is defined in a weak sense. In fact, for $s, g \in L^2(\mathbb{R}^d)$, we have $g(\cdot)s(\cdot - \boldsymbol{t}) \in L^1$, so that the *short-time Fourier transform* $\mathcal{V}_g g$ of g with window s is well-defined as the function

$$\mathcal{V}_s g(\boldsymbol{\nu},\boldsymbol{t}) \stackrel{\text{def}}{=} \int_{\mathbb{R}^d} g(\boldsymbol{t_0})\overline{s(\boldsymbol{t_0}-\boldsymbol{t})}\mathrm{e}^{-\mathrm{i}2\pi\langle\boldsymbol{\nu},\boldsymbol{t_0}-\boldsymbol{t}\rangle}\, \mathrm{d}\boldsymbol{t_0} \tag{2.10}$$

on $L^2(\mathbb{R}^d \times \mathbb{R}^d)$. Hence, Hs is defined to be the unique $L^2(\mathbb{R}^d)$-function with

$$\langle Hs, g\rangle_{L^2(\mathbb{R}^d)} = \langle S_H, \mathcal{V}_s g\rangle_{L^2(\mathbb{R}^d\times\mathbb{R}^d)}\,.$$

Many similarly obtained system functions are known under a rich plethora of different names in the literature, ranging back to a first systematic study by Zadeh and Bello [Zad50, Bel63, Bel64] (see also [Ric03] for an overview). The integral representations of importance in this text describe H in terms of the *spreading function* S_H, the *kernel* κ_H, the *time-varying impulse response* h, the

Kohn-Nirenberg symbol σ_H and the *bifrequency function* B_H. These system functions are related via the following partial Fourier transforms:

$$\begin{array}{ccc} \kappa_H(\boldsymbol{t_0}, \boldsymbol{t_0} - \boldsymbol{t}) = h(\boldsymbol{t_0}, \boldsymbol{t}) & \overset{\mathcal{F}_{\boldsymbol{t}\to\boldsymbol{\xi}}}{\longmapsto} & \sigma_H(\boldsymbol{t_0}, \boldsymbol{\xi}) \\ \Big\downarrow \mathcal{F}_{\boldsymbol{t_0}\to\boldsymbol{\nu}} & & \Big\downarrow \mathcal{F}_{\boldsymbol{t_0}\to\boldsymbol{\nu}} \\ \mathrm{e}^{-\mathrm{i}2\pi\langle\boldsymbol{\nu},\boldsymbol{t}\rangle} S_H(\boldsymbol{\nu}, \boldsymbol{t}) & \overset{\mathcal{F}_{\boldsymbol{t}\to\boldsymbol{\xi}}}{\longmapsto} & B_H(\boldsymbol{\nu}, \boldsymbol{\xi}) \end{array} \tag{2.11a}$$

For κ_H being smooth and compactly supported, we apply the Fubini–Tonelli theorem, (2.11a) and Plancherel's theorem to (2.8) to get

$$\begin{aligned} (Hs)(\boldsymbol{t_0}) &= \int \kappa_H(\boldsymbol{t_0}, \boldsymbol{t}) s(\boldsymbol{t}) \,\mathrm{d}\mu(\boldsymbol{t}) \\ &= \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} S_H(\boldsymbol{\nu}, \boldsymbol{t}) \mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\nu}, \boldsymbol{x}-\boldsymbol{t}\rangle} d\boldsymbol{\nu} s(\boldsymbol{t_0} - \boldsymbol{t}) \,\mathrm{d}\boldsymbol{t} && (2.11b) \\ &= \int_{\mathbb{R}^d} h(\boldsymbol{t_0}, \boldsymbol{t}) s(\boldsymbol{t_0} - \boldsymbol{t}) \,\mathrm{d}\boldsymbol{t} && (2.11c) \\ &= \int_{\mathbb{R}^d} \sigma_H(\boldsymbol{t_0}, \boldsymbol{\xi}) \widehat{s}(\boldsymbol{\xi}) \mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{t_0}, \boldsymbol{\xi}\rangle} \,\mathrm{d}\boldsymbol{\xi} && (2.11d) \\ &= \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} B_H(\boldsymbol{\nu} - \boldsymbol{\xi}, \boldsymbol{\xi}) \widehat{s}(\boldsymbol{\xi}) \,\mathrm{d}\boldsymbol{\xi} \mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{t_0}, \boldsymbol{\nu}\rangle} \,\mathrm{d}\boldsymbol{\nu}. && (2.11e) \end{aligned}$$

Note that the validity above extends to general Hilbert–Schmidt operators via a density argument. Certainly, the convergence of the integrals is considered in the L^2 sense as generally done in L^2-Fourier analysis. In this case, the equalities above hold for almost every $\boldsymbol{t_0}$.

It follows naturally from (2.11) to view $h(\boldsymbol{t_0}, \boldsymbol{t})$ as the impulse response at $\boldsymbol{t_0}$ to an impulse at $\boldsymbol{t_0} - \boldsymbol{t}$ and to view $\sigma_H(\boldsymbol{t_0}, \boldsymbol{\xi})$ as the frequency response at $\boldsymbol{t_0}$ to a complex exponential with frequency $\boldsymbol{\xi}$.

A Hilbert–Schmidt operator H is usually called *underspread* if its spreading function is contained in a rectangle with area less than one and *overspread* otherwise. Underspread operators have the important property that they are *identifiable* [KP06, PW06], which means that the operator H can be computed from its response to a selected single input function. The most well-known example of identifiability is the fact that linear time-invariant channels are completely characterized by their action on a Dirac delta distribution, that is, by their impulse response.

3 The channel model

An important and uniting property for radio and sonar communications in air and water, respectively, is *multipath propagation*, which means that due to reflections on different structures in the environment, the transmitted signal reaches the

receiver via a possibly infinite number of different wave propagation paths, as illustrated in Figure 2 (see, for example, [Rap02]).

In Sections 3.1–3.2, we examine the multipath propagation model at some depth under the standard assumptions that the electric field component at the receiver is the superposition of the contributions from all signal paths leading there, and that the action of the channel on a transmitted signal is the superposition of the action on all complex exponentials in a Fourier expansion of the signal. For this we use a standard and straightforward linear extension ($H(u + \mathrm{i}v) = Hu + \mathrm{i}Hv$) of H to complex valued functions. Initially, we do also allow the channel to be of infinite lifelength, which is necessary for our class of modelled channels to include, for example, the identity operator, which has a Dirac delta distribution spreading function.

For communications applications, however, only finite lifelength channels are important. We show in Section 3.3 that this subclass of channels can be completely described by very smooth spreading functions with "essentially compact" support.

3.1 Single path frequency response

Most wireless communication channels change their characteristics slowly compared to the rate at which transmission symbols are sent. Significant changes either require a long time-period to evolve, or they are caused by abrupt changes in the environment, for example, when a mobile telephone user drives into a tunnel. The standard countermeasure is to regularly make new estimates of the channel. In OFDM based methods this is usually done by sending pilot symbols, pilot tones or scattered pilots [GHS+01]. For a more general treatment, see [GHS+01, KP06, LPW05, MMH+02, PW06].

Thus, from now on we shall only consider the *short-time behaviour* of the channel during time intervals I that are short enough to assume a fixed collection

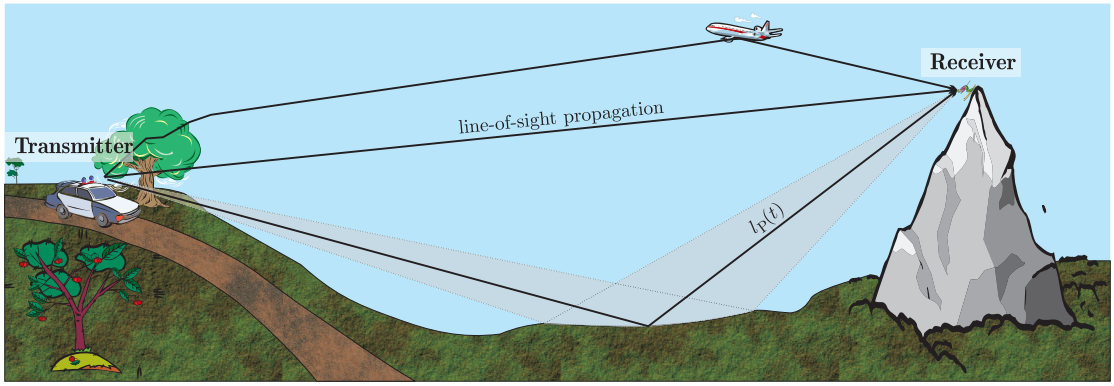


Figure 2: The transmitted signal reaches the receiver along a continuum of different signal paths. Each path P has time-varying length $l_P(t)$.

of signal paths with the length $l_{\mathrm{P}}(t)$ of path P being a linear function of the time. That is, we assume the length and prolongation-speed of each path to be such that for some $T_0 \in I$ and all $t \in I$,

$$l_{\mathrm{P}}(t) = L_{\mathrm{P}} + V_{\mathrm{P}} \cdot (t - T_0) \qquad \text{with} \qquad |V_{\mathrm{P}} \cdot (t - T_0)| \ll L_{\mathrm{P}}. \tag{3.1}$$

Physical constraints on the speed of antenna and reflecting object movements give some upper bound $V_{\max}$ for $|V_{\mathrm{P}}|$. We will assume $V_{\max}$ to be smaller than the *wave propagation speed* V_{w}, so that

$$V_{\mathrm{w}} > V_{\max} \geq |V_{\mathrm{P}}| \quad \text{for all paths P.}$$

Hence, if a simple harmonic $\mathrm{e}^{\mathrm{i}2\pi\xi t}$ is sent along the path P without any attenuation or perturbations, then the received signal would be

$$\mathrm{e}^{\mathrm{i}2\pi\xi\left(t-\frac{l_{\mathrm{P}}(t)}{V_{\mathrm{w}}}\right)} = \mathrm{e}^{\mathrm{i}2\pi\xi\left(\left(1-\frac{V_{\mathrm{P}}}{V_{\mathrm{w}}}\right)t-\frac{L_{\mathrm{P}}-V_{\mathrm{P}}T_0}{V_{\mathrm{w}}}\right)} = \mathrm{e}^{\mathrm{i}2\pi\xi\left(1-\frac{V_{\mathrm{P}}}{V_{\mathrm{w}}}\right)\left(t-\frac{L_{\mathrm{P}}-V_{\mathrm{P}}T_0}{V_{\mathrm{w}}-V_{\mathrm{P}}}\right)} = T_{t_{\mathrm{P}}} M_{\nu_{\mathrm{P}}\xi} \mathrm{e}^{\mathrm{i}2\pi\xi(\cdot)} \tag{3.2a}$$

where the time and frequency shifts t_{P} and $\nu_{\mathrm{P}}\xi$ satisfy

$$t_{\mathrm{P}} = \frac{L_{\mathrm{P}} - V_{\mathrm{P}}T_0}{V_{\mathrm{w}} - V_{\mathrm{P}}} \quad \text{and} \quad \nu_{\mathrm{P}} = -\frac{V_{\mathrm{P}}}{V_{\mathrm{w}}} \in \left[-\frac{V_{\max}}{V_{\mathrm{w}}}, \frac{V_{\max}}{V_{\mathrm{w}}}\right] \subset (-1, 1). \tag{3.2b}$$

This mapping from $(V_{\mathrm{P}}, L_{\mathrm{P}})$ to $(\nu_{\mathrm{P}}, t_{\mathrm{P}})$ is invertible with inverse

$$V_{\mathrm{P}} = -\nu_{\mathrm{P}} V_{\mathrm{w}} \qquad \text{and} \qquad L_{\mathrm{P}} = V_{\mathrm{w}} \left(t_{\mathrm{P}} \left(1 + \nu_{\mathrm{P}}\right) - \nu_{\mathrm{P}} T_0\right). \tag{3.2c}$$

By (3.2b), (3.1) and (3.2c), there is a compact set $K \subset (-1, 1)$ and some $A \in \mathbb{R}$ such that $(\nu_{\mathrm{P}}, t_{\mathrm{P}}) \in K \times [A, \infty)$ for all paths P.

Now fix some arbitrary $(\nu, t) \in K \times [A, \infty)$ and some path P with frequency response parameters $(\nu_{\mathrm{P}}, t_{\mathrm{P}}) = (\nu, t)$ in (3.2a). The channel operator action on a complex exponential $s_\xi(t_0) = \mathrm{e}^{\mathrm{i}2\pi\xi t_0}$ consists of the following components:

1. A multiplication by a transmitter amplitude gain $G_{\mathrm{T}}(\mathrm{P})$.

   If we identify the path with the angular direction in which it leaves the transmitter, then we can integrate (or sum) over all P and note that for energy conservation reasons the total power gain $\int |G_{\mathrm{T}}(\mathrm{P})|^2 \, \mathrm{dP}$ must be finite.

2. The time-frequency shift by $(\nu_{\mathrm{P}}\xi, t_{\mathrm{P}})$ that is given in (3.2a).

3. Attenuation with a factor[3] $A_\xi(\mathrm{P}) \in \mathbb{R}$ that for free space transmission has size $\mathcal{O}(L_{\mathrm{P}}^{-2})$ for large L_{P} [Rap02, Reu74].

   However, the decay is usually much faster and exponential decay $\mathcal{O}(\mathrm{e}^{-a_\xi L_{\mathrm{P}}})$ can be argued for if we assume some fixed minimum attenuation every time

[3]We allow $A_\xi(\mathrm{P})$ to be negative to include potential sign-changes caused by reflections.

a signal is reflected [Str06]. Even for radio signals propagating through the atmosphere without reflections (*line-of-sight propagation*, see Figure 2), frequency selective absorption causes exponential decay with faster decay for higher frequencies [Reu74, Section 2.1.7]. From this and (3.2c) we get that for some $a_\xi, C > 0$,

$$|A_\xi(\mathrm{P})| \leq C\mathrm{e}^{-a_\xi L_\mathrm{P}} \leq C_\xi \mathrm{e}^{-\alpha t_\mathrm{P}} \chi_{[A,\infty)}(t_\mathrm{P}) \tag{3.3}$$

with $C_\xi = \sup_{|\nu|<1} C\mathrm{e}^{a_\xi V_\mathrm{w} \nu T_0} < C\mathrm{e}^{a_\xi V_\mathrm{w}|T_0|}$ and $\alpha = \inf_\xi \inf_{|\nu|<1} a_\xi V_\mathrm{w}(1-\nu) > 0$.

4. Multiplication by a receiver amplitude gain $G_\mathrm{R}(\mathrm{P})$, which for any kind of practical use must also satisfy that $\int |G_\mathrm{R}(\mathrm{P})|^2 \,\mathrm{dP} < \infty$.

Altogether, the above steps add up to the following single path frequency response:

$$\begin{aligned} s_\xi(\cdot) \overset{\text{def}}{=} \mathrm{e}^{\mathrm{i}2\pi\xi(\cdot)} &\xrightarrow{\text{Transmitter}} G_\mathrm{T}(\mathrm{P}) s_\xi \\ &\xrightarrow{\text{TF-shift (3.2a)}} G_\mathrm{T}(\mathrm{P}) T_{t_\mathrm{P}} M_{\nu_\mathrm{P}\xi} s_\xi \\ &\xrightarrow{\text{Attenuation}} G_\mathrm{T}(\mathrm{P}) A_\xi(\mathrm{P}) T_{t_\mathrm{P}} M_{\nu_\mathrm{P}\xi} s_\xi \\ &\xrightarrow{\text{Receiver}} G_\mathrm{T}(\mathrm{P}) A_\xi(\mathrm{P}) G_\mathrm{R}(\mathrm{P}) T_{t_\mathrm{P}} M_{\nu_\mathrm{P}\xi} s_\xi \end{aligned} \tag{3.4}$$

Now set

$$B_\xi(\mathrm{P}) \overset{\text{def}}{=} A_\xi(\mathrm{P}) \mathrm{e}^{\alpha_\xi t_\mathrm{P}}$$

for all paths P, so that by (3.3), $|B_\xi(\mathrm{P})| \leq C_\xi$ for all P. Further, we set $P_{\nu,t} = \{\mathrm{P} : (\nu_\mathrm{P}, t_\mathrm{P}) = (\nu, t)\}$.

As usual for electromagnetic waves, we expect the electric field component measured at the receiver to be the superposition of the electric field components received from the different paths P (and similarly for sonar waves), or written as a formal integration[4]

$$(Hs_\xi)(t_0) = \int_{K\times[A,\infty)} \left(\int_{P_{\nu,t}} G_\mathrm{T}(\mathrm{P}) B_\xi(\mathrm{P}) G_\mathrm{R}(\mathrm{P}) \,\mathrm{dP} \right) \mathrm{e}^{-\alpha_\xi t} (T_t M_{\nu\xi} s_\xi)(t_0) \,\mathrm{d}(\nu, t). \tag{3.5}$$

If we denote the inner integral

$$r_\xi(\nu, t) \overset{\text{def}}{=} \int_{P_{\nu,t}} G_\mathrm{T}(\mathrm{P}) B_\xi(\mathrm{P}) G_\mathrm{R}(\mathrm{P}) \,\mathrm{dP}, \tag{3.6a}$$

then it follows from the Hölder inequality, the bound $|B_\xi(\mathrm{P})| \leq C_\xi$ and items 1 and 4 above that

$$\begin{aligned} \int_{K\times[A,\infty)} |r_\xi(\nu,t)| \,\mathrm{d}(\nu,t) \leq &\int_{\cup_{(\nu,t)\in K\times[A,\infty)} P_{(\nu,t)}} |G_\mathrm{T}(\mathrm{P}) B_\xi(\mathrm{P}) G_\mathrm{R}(\mathrm{P})| \,\mathrm{dP} \\ \leq &C_\xi \cdot \|G_\mathrm{T}\|_2 \cdot \|G_\mathrm{R}\|_2 < \infty. \end{aligned} \tag{3.6b}$$

[4]Here again, $\int \cdots \mathrm{dP}$ is shorthand notation for the integration over the different angles in a polar coordinate system.

Both for the actual transmitted real-valued signals and for our linear extension of H to complex-valued signals, the gain and attenuation factors are all real-valued. We shall however, without any extra effort, allow for complex-valued r in our mathematical model. Moreover, for inclusion of some important idealized borderline cases such as r being a Dirac delta distribution, and for avoiding some computational distribution theory technicalities, we choose to model the integrals $\rho_\xi(U \times V) \stackrel{\text{def}}{=} \int_{U\times V} r_\xi(\nu,t)\,\mathrm{d}(\nu,t)$ over sets $U \times V \subseteq K \times [A,\infty)$ to be a complex Borel measure ρ_ξ with finite total variation, that is

$$\rho_\xi(U \times V) = \int_{U\times V} \mathrm{d}\rho_\xi(\nu,t) \qquad \text{with} \qquad |\rho_\xi|\,(K \times [A,\infty)) < \infty. \tag{3.7a}$$

Thus, in this mathematical model, (3.5) takes the form

$$(Hs_\xi)(t_0) = \int_{K\times[A,\infty)} \mathrm{e}^{-\alpha_\xi t}(T_t M_{\nu\xi} s_\xi)(t_0)\,\mathrm{d}\rho_\xi(\nu,t). \tag{3.7b}$$

Note that this model includes, for example, Dirac measures and thus also the identity operator.

3.2 Narrowband signals

We shall call the transmitted signal s *narrowband* if $\widehat{s}$ is well-localized enough to justify the approximations

$$\nu\xi \approx \nu\xi_0, \qquad \mathrm{e}^{-\alpha_\xi t} \approx \mathrm{e}^{-\alpha_{\xi_0} t} \qquad \text{and} \qquad \rho_\xi \approx \rho_{\xi_0} \tag{3.8}$$

in the computations leading to (3.9b) below. We will primarily assume this narrowband assumption to hold for the same ξ_0 in the entire transmission frequency band. In the remark on page 29, we will show that this assumption holds true for some radio communications examples and discuss a refined model with different ξ_0 for different basis functions that is necessary in underwater sonar communications.

Suppose now that the physical channel has the property that its action on a signal s is the superposition of its action on each complex exponential in a Fourier expansion of s, that is,

$$Hs(\cdot) = H\int_{\mathbb{R}} \widehat{s}(\xi)\mathrm{e}^{\mathrm{i}2\pi\xi(\cdot)}\,\mathrm{d}\xi = \int_{\mathbb{R}} \widehat{s}(\xi) H\mathrm{e}^{\mathrm{i}2\pi\xi(\cdot)}\,\mathrm{d}\xi. \tag{3.9a}$$

Then, at least for bandlimited and thus continuous narrowband L^1-signals s, we

can apply (3.7b), (3.8) and the Fubini–Tonelli theorem to obtain

$$\begin{aligned}
Hs(t_0) &= \int_{\mathbb{R}} \widehat{s}(\xi) \int_{K\times[A,\infty)} \mathrm{e}^{-\alpha_\xi t}\mathrm{e}^{\mathrm{i}2\pi\nu\xi(t_0-t)}\mathrm{e}^{\mathrm{i}2\pi\xi(t_0-t)}\,\mathrm{d}\rho_\xi(\nu,t)\,\mathrm{d}\xi \\
&\approx \int_{\mathbb{R}} \widehat{s}(\xi) \int_{K\times[A,\infty)} \mathrm{e}^{-\alpha_{\xi_0} t}\mathrm{e}^{\mathrm{i}2\pi\nu\xi_0(t_0-t)}\mathrm{e}^{\mathrm{i}2\pi\xi(t_0-t)}\,\mathrm{d}\rho_{\xi_0}(\nu,t)\,\mathrm{d}\xi \\
&= \int_{K\times[A,\infty)} \mathrm{e}^{-\alpha_{\xi_0} t}\mathrm{e}^{\mathrm{i}2\pi\nu\xi_0(t_0-t)} \int_{\mathbb{R}} \widehat{s}(\xi)\mathrm{e}^{\mathrm{i}2\pi\xi(t_0-t)}\,\mathrm{d}\xi\,\mathrm{d}\rho_{\xi_0}(\nu,t) \\
&= \int_{K\times[A,\infty)} \mathrm{e}^{-\alpha_{\xi_0} t}\mathrm{e}^{\mathrm{i}2\pi\nu\xi_0(t_0-t)} s(t_0-t)\,\mathrm{d}\rho_{\xi_0}(\nu,t).
\end{aligned}$$

We use the last expression as definition of our mathematical model

$$Hs(t_0) \stackrel{\text{def}}{=} \int_{K\times[A,\infty)} \mathrm{e}^{-\alpha_{\xi_0} t}(T_t M_{\nu\xi_0} s)(t_0)\,\mathrm{d}\rho_{\xi_0}(\nu,t). \tag{3.9b}$$

If $s \in L^2(\mathbb{R})$, then by (3.9b) and the Minkowski integral inequality

$$\begin{aligned}
\|Hs\|_2 &= \left\| \int_{K\times[A,\infty)} \mathrm{e}^{-\alpha_{\xi_0} t}(T_t M_{\nu\xi_0} s)(\cdot)\,\mathrm{d}\rho_{\xi_0}(\nu,t) \right\|_2 \\
&\le \int_{K\times[A,\infty)} \mathrm{e}^{-\alpha_{\xi_0} t}\,\|T_t M_{\nu\xi_0} s\|_2\,\mathrm{d}\,|\rho_{\xi_0}|\,(\nu,t) \\
&\le |\rho_{\xi_0}|\,(K\times[A,\infty))\mathrm{e}^{-\alpha_{\xi_0} A}\,\|s\|_2\,.
\end{aligned}$$

Hence equation (3.9b) defines a bounded linear mapping $H\colon L^2(\mathbb{R}) \to L^2(\mathbb{R})$. If, in addition, ρ_{ξ_0} is absolutely continuous with respect to the Lebesgue measure, then we can write $\mathrm{d}\rho_{\xi_0}(\nu,t) = r_{\xi_0}(\nu,t)\,\mathrm{d}(\nu,t)$ where r_{ξ_0} is the function in (3.6), which equals the inner integral in our physical model (3.5). In practice, r_{ξ_0} will be bounded and thus also in $L^2(\mathbb{R})$. Application of this to (3.9b) and the substitution $\nu' = \nu\xi_0$ gives

$$\begin{gathered}
Hs(\cdot) = \int_{K'\times[A,\infty)} S_H(\nu',t)(T_t M_{\nu'} s)(\cdot)\,\mathrm{d}(\nu',t), \qquad K' \subseteq (-\xi_0,\xi_0) \text{ compact}, \\
S_H(\nu',t) \stackrel{\text{def}}{=} \frac{1}{\xi_0} r_{\xi_0}\left(\frac{\nu'}{\xi_0},t\right)\mathrm{e}^{-\alpha_{\xi_0} t}\chi_{K'\times[A,\infty)}(\nu',t) \qquad \text{and} \qquad r_{\xi_0} \in L^1 \cap L^2(\mathbb{R}\times\mathbb{R}).
\end{gathered} \tag{3.9c}$$

Alternatively we can use (3.9b) as the definition of a larger space of operators on a smaller space of functions by allowing a larger subclass of the complex Borel measures, such as, the class of all complex Borel measures ρ_{ξ_0} for which the mapping

$$\varphi \mapsto \int_{K\times[A,\infty)} \varphi(\nu,t)\mathrm{e}^{-\alpha_{\xi_0} t}\,\mathrm{d}\rho_{\xi_0}(\nu,t)$$

defines a linear bounded functional S_H on $S_0(K \times [A, \infty))$. Then for all functions s for which the mapping

$$(\nu\xi_0, t) \mapsto (T_t M_{\nu\xi_0} s)(t_0) \in S_0(K' \times [A, \infty)) \qquad \text{for all } t_0, \tag{3.10}$$

is well-defined, it follows that (3.9b) is pointwise well-defined for all t_0. Consequently (3.9b) can be interpreted as the application of a functional $S_H \in S_0'$ to the test function (3.10), or, with the usual formal integral notation,

$$Hs(\cdot) = \int_{K' \times [A,\infty)} S_H(\nu', t)(T_t M_{\nu'} s)(\cdot)\, \mathrm{d}(\nu', t), \qquad S_H \in S_0'(K' \times [A, \infty)).$$

Since the space $S_0'(\mathbb{R} \times \mathbb{R})$ includes Dirac delta distributions, this model includes important idealized borderline cases such as the following:

Line-of-sight transmission: $S_H = a\delta_{\nu_0,t_0}$, a Dirac distribution at (ν_0, t_0) representing a time- and Doppler-shift with attenuation a.

Time-invariant systems: $h(x, t) = h(0, t)$ and $S_H(\nu, t) = h(t)\delta_0(\nu)$.

Moreover, S_0' excludes derivatives of Dirac distributions, which can be used to avoid complex-valued Hs with no physical meaning [Ric03, Sec. 3.1.1]. Further, S_0 is the smallest Banach space of test functions with some useful properties like invariance under time-frequency shifts [Grö00, p.253], thus allowing for time-frequency analysis on its dual S_0' which is, in that particular sense, the largest possible Banach space of tempered distributions that is useful for time-frequency analysis. One more motivation for considering spreading functions in S_0' is that Hilbert–Schmidt operators are compact, hence, they exclude invertible operators, such as the example $S_H = a\delta_{\nu_0,t_0}$ above, and small perturbations of invertible operators, which are useful in the theory of radar identification and in some mobile communication applications. For results using a Banach space setup, see for example [PW06, Str06].

Hence it may come as a surprise that we will show in Section 3.3 that the Hilbert–Schmidt model (3.9c) is a natural choice for wireless communications channels. This also justifies our use of this model in the remaining paper.

3.3 Finite lifelength channels

We shall show that wireless communications channels can be modelled well by well-localized $C^{(\infty)}$-spreading functions. This fact allows for a minimal use of distribution theory in our analysis and adds simplicity to proofs, results and software development both in this report and for mathematical and numerical analysis of wireless communications channels in general.

In the following, we will assume that the bifrequency function

$$B_H(\nu, \cdot) \text{ is compactly supported.} \tag{3.11}$$

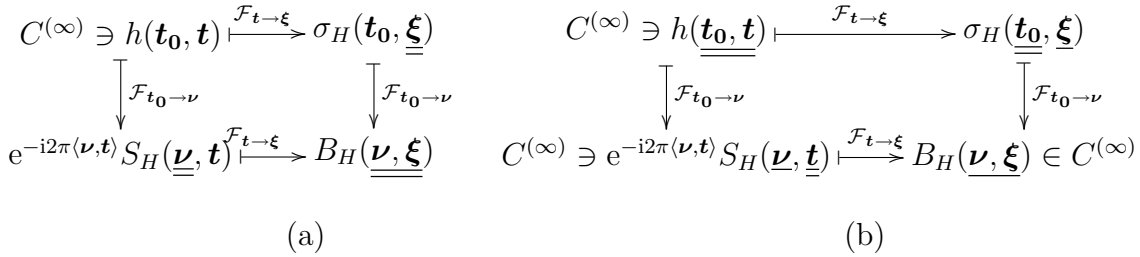


Figure 3: (a) Bandlimiting properties of the physical channel provides us with compactly supported B_H. Thus $h \in C^{(\infty)}$, but S_H may be a tempered distribution, for example, if $h(\boldsymbol{t_0}, \boldsymbol{t}) = h(\boldsymbol{0}, \boldsymbol{t})$. (b) Our restriction to *finite lifelength* channels gives a well localized $S_H \in C^{(\infty)}$. In (a) and (b) we denote with underlining subexponential decay and compact support.

This is not strictly true in general, but for communications applications we only need to model the channel response to a family of signals s that all are bandlimited to some common frequency band Ω_1. Hence it is clear from (2.11e) that Hs will only depend on the restriction of B_H to $\mathbb{R} \times \Omega_1$, so that we are free to set B_H equal to zero outside $\mathbb{R} \times \Omega_2$ for an arbitrary and compact set $\Omega_2 \supseteq \Omega_1$. Moreover, recall from (3.9c) that $S_H(\cdot, t)$ has compact support as well. Hence, combining this with (2.11a) and (3.11), we see that the distribution $B_H(\nu, \xi)$ is compactly supported. Consequently, since the bifrequency function satisfies $B_H = \widehat{h}$, we have that $h \in C^{(\infty)}(\mathbb{R}^2)$. We summarize a straightforward generalization of these properties to functions on $\mathbb{R}^d$ in Figure 3 (a).

Since we are modelling the short time input-output relationships of a channel, any useful communications system must be constructed to be independent of the properties of h outside some compact set K_h. Thus, we are free to multiply h with a compactly supported function $w \in C^{(\infty)}(\mathbb{R}^{2d})$ such that $w = 1$ on K_h and $\widehat{w}$ is subexponentially decaying (as described in Section 2.2). It is easy to check that it follows from this and from the compact support of $\widehat{h}$ that also the convolution $\widehat{wh} = \widehat{w} * \widehat{h}$ is subexponentially decaying. Now let H_1 be the Hilbert–Schmidt operator with time-varying impulse response $wh \in C^{(\infty)}$. From the fact that the space of Schwartz functions is invariant under partial Fourier transforms, it follows that also $S_{H_1} \in C^{(\infty)}$. This gives an operator with system function properties that we summarize in the multivariate case in Figure 3 (b). We will also assume that w is chosen to be "wide and smooth enough" so that the smooth cut-off of $S_H(\boldsymbol{\nu}, \cdot)$ only deletes a very small-amplitude and negligible part of its exponential tail, and so that also the "blurring out" of the compact support of $S_H(\cdot, \boldsymbol{t})$ to subexponential decay has a rather small impact on the shape of S_H, which therefore can be expected to resemble those given in the figures of Section 6.

The following sections are devoted to Hilbert–Schmidt operators having exactly

the properties that are summarized in Figure 3 (b). All results apply directly to the communication channels described in this section as long as the narrowband assumption of Section 3.2 holds for the entire frequency band of the transmitted signals. We describe in a remark on page 29 how refined versions of our results can be applied to wideband signals as long as the attenuation factor A_ξ of (3.3) is roughly frequency independent within the transmission frequency band.

REMARK. The exponential decay of $S_H(\nu,\cdot)$ only affects the just mentioned shape of S_{H_1}. In general, the reasoning in this and the following sections hold also with exponential and subexponential decays replaced by other forms of rapid decay.

4 Discretization of the channel model

In this section we derive finite sum formulas for the computation of the matrix representation of the coefficient mapping G in (2.7) for classes of multivariate Hilbert–Schmidt operators H that satisfy the properties summarized in Figure 3 (b).

For the series expansions in Gabor frames $(g_{\boldsymbol{q},\boldsymbol{r}})$ and $(\gamma_{\boldsymbol{q}',\boldsymbol{r}'})$ in Section 2.3, H maps any L^2-function $s = \sum_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d} \langle s, \widetilde{g}_{\boldsymbol{q},\boldsymbol{r}}\rangle\, g_{\boldsymbol{q},\boldsymbol{r}}$ to

$$\begin{aligned} Hs &= \sum_{\boldsymbol{q}',\boldsymbol{r}'\in\mathbb{Z}^d} \langle Hs, \gamma_{\boldsymbol{q}',\boldsymbol{r}'}\rangle\, \widetilde{\gamma}_{\boldsymbol{q}',\boldsymbol{r}'} = \sum_{\boldsymbol{q}',\boldsymbol{r}'\in\mathbb{Z}^d} \left\langle H \sum_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d} \langle s, \widetilde{g}_{\boldsymbol{q},\boldsymbol{r}}\rangle\, g_{\boldsymbol{q},\boldsymbol{r}}\,, \gamma_{\boldsymbol{q}',\boldsymbol{r}'} \right\rangle \widetilde{\gamma}_{\boldsymbol{q}',\boldsymbol{r}'} \\ &= \sum_{\boldsymbol{q}',\boldsymbol{r}'\in\mathbb{Z}^d} \left(\sum_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d} \langle s, \widetilde{g}_{\boldsymbol{q},\boldsymbol{r}}\rangle\, \langle Hg_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q}',\boldsymbol{r}'}\rangle \right) \widetilde{\gamma}_{\boldsymbol{q}',\boldsymbol{r}'}, \end{aligned} \tag{4.1}$$

where convergence in $L^2(\mathbb{R}^d)$ follows from the continuity of H and of the inner product. Thus the coefficient mapping G in (2.7) is

$$G\colon\ (\langle s, \widetilde{g}_{\boldsymbol{q},\boldsymbol{r}}\rangle)_{\boldsymbol{q},\boldsymbol{r}} \mapsto \left(\sum_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d} \langle s, \widetilde{g}_{\boldsymbol{q},\boldsymbol{r}}\rangle\, \langle Hg_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q}',\boldsymbol{r}'}\rangle \right)_{\boldsymbol{q}',\boldsymbol{r}'}$$

with biinfinite matrix representation

$$\mathrm{G}_{\boldsymbol{q}',\boldsymbol{r}';\boldsymbol{q},\boldsymbol{r}} = \langle Hg_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q}',\boldsymbol{r}'}\rangle\,, \tag{4.2}$$

and with $2d$-dimensional indices $(\boldsymbol{q}',\boldsymbol{r}')$ and $(\boldsymbol{q},\boldsymbol{r})$ for rows and columns respectively.

For communications applications with Q carrier frequencies, at least Q samples of every received symbol are needed in the receiver. Thus a hasty and naive approach to computing the matrix elements could start with a $Q \times Q$ matrix representation of H for computing the samples of $Hg_{q,r}$. If every R neighbouring symbols have overlapping ϵ-essential support, then we need to compute $(RQ)^2$

matrix elements $\langle Hg_{q,r}, \gamma_{q',r'}\rangle$, which, with this approach, would require $R^2 \cdot \mathcal{O}(Q^5)$ arithmetic operations with Q typically being at least of the size 256–1024 in radio communications, and with $R = 4$ for $\epsilon = 10^{-6}$ and the optimally well-localized Gaussian windows that we shall use for our example applications in Section 6 (see Figure 12). This is a quite demanding task, so there is a clear need for the more efficient formulas and algorithms that we shall derive in the following sections.

We begin in Section 4.1 with motivating the use of well time-frequency localized Gabor bases. Then, in Section 4.2 we introduce the tools and notation that we find most suitable for discretization of Hilbert–Schmidt operators with the properties summarized in Figure 3 (b). Finally, in Section 4.3 we use these tools for deriving more efficient formulas for computing the matrix elements under a minimum of further assumptions, simplifications or approximations.

REMARK. If $(g_{\boldsymbol{q},\boldsymbol{r}})$ is a frame for its closed linear span but not a Riesz basis, then the coefficient subspace $C_{\widetilde{g}}$ of (2.7) will be a proper subspace of l^2. Thus an orthogonal projection to $C_{\widetilde{g}}$ in the receiver would add some stability against noise and perturbations, but at the cost of an additional requirement on the transmitter to only use coefficient sequences in $C_{\widetilde{g}}$.

For systems in use today, it is typical to instead use Gabor Riesz bases and arbitrary coefficient sequences with coefficients chosen from some standard coefficient constellation, such as quadrature amplitude modulation (QAM), with maximum coefficient amplitude given by transmission power regulations.

4.1 Gabor bases for near-diagonalization

Although (4.1) holds for any pair of frames $(g_{\boldsymbol{i}})$ and $(\gamma_{\boldsymbol{j}})$, Gabor Riesz bases are traditionally used for communications applications. Gabor systems give good diagonalization of G, especially if all basis elements are narrowband (see below and [KPZ02]).

We call an operator H *time-invariant* if it commutes with the translation operator $T_{\boldsymbol{t_0}}$ for any $\boldsymbol{t_0}$, that is, if $T_{\boldsymbol{t_0}}H = HT_{\boldsymbol{t_0}}$. Consequently, the impulse response h of H satisfy $h(\boldsymbol{t_0}, \boldsymbol{t}) = h(\boldsymbol{0}, \boldsymbol{t})$ and (2.11c) becomes a convolution $h(\boldsymbol{0}, \cdot) * s$. Then the family of complex exponentials $\mathrm{e}^{i2\pi\langle\boldsymbol{\xi},\cdot\rangle}$ are "eigenfunctions" in the sense that for inputs of the form $s(\cdot) = \mathrm{e}^{i2\pi\langle\boldsymbol{\xi},\cdot\rangle}\chi_{[\boldsymbol{0},\boldsymbol{L}]}(\cdot)$, there is some complex scalar $\lambda_{\boldsymbol{\xi}}$ such that if $\operatorname{supp} h = [\boldsymbol{0}, \boldsymbol{L_h}]$, then $Hs = \lambda_{\boldsymbol{\xi}} s$ in the interval $[\boldsymbol{L_h}, \boldsymbol{L}]$,

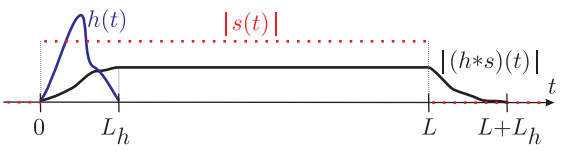


Figure 4: Convolution of a compactly supported complex exponential $s(t) = \chi_{[0,L]}(t)\mathrm{e}^{i2\pi\xi t}$ with a function $h = h\chi_{[0,L_h]}$ will on the interval $[L_h, L]$ equal the same complex exponential multiplied with a complex scalar.

as illustrated in Figure 4. Thus G is easily diagonalized by using Gabor windows $g = \chi_{[\mathbf{0},\boldsymbol{L}]}$, $\gamma = \chi_{[\boldsymbol{L_h},\boldsymbol{L}]}$ and lattice constants such that the resulting Gabor systems $(g_{\boldsymbol{k},\boldsymbol{l}})$ and $(\gamma_{\boldsymbol{k},\boldsymbol{l}})$ are biorthogonal bases for their respective span. This trick is used in wireline communications, where the smaller support of γ is obtained by removing a guard interval (often called *cyclic prefix*) from g. See, for example, [Gri02, Section 2.3] for more details and further references.

In *wireless* communications, we can at most hope for approximate diagonalization of the channel operator due to its time varying nature. In general, two different time-varying operators do not commute, so both cannot be diagonalized with the same choice of bases. Thus, diagonalization is usually only possible in the following sense: Typically, $(Hg_{\boldsymbol{q},\boldsymbol{r}})$ is a finite and linearly independent sequence, and thus a Riesz basis with some dual basis $\left(\widetilde{Hg_{\boldsymbol{q},\boldsymbol{r}}}\right)$. In fact, since the computations in (4.1) are not restricted to Gabor frames, we can set $\gamma_{\boldsymbol{q}',\boldsymbol{r}'} = \widetilde{Hg_{\boldsymbol{q},\boldsymbol{r}}}$, which gives true diagonalization of (4.2), but in general, $\widetilde{Hg_{\boldsymbol{q},\boldsymbol{r}}}$ will not be a Gabor frame or have any other simple structure that enables efficient computation of all $\widetilde{Hg_{\boldsymbol{q},\boldsymbol{r}}}$ and all the inner products $\left\langle Hg_{\boldsymbol{q},\boldsymbol{r}}, \widetilde{Hg_{\boldsymbol{q}',\boldsymbol{r}'}} \right\rangle$.

Thus, for computational complexity to meet practical restrictions we will use "almost dual" Gabor bases $(g_{\boldsymbol{q},\boldsymbol{r}})$ and $(\gamma_{\boldsymbol{q}',\boldsymbol{r}'})$, such as the Gabor bases proposed in [MSG+05]. We are primarily interested in bases that are good candidates for providing low intersymbol and interchannel interference (ISI and ICI). As proposed in [MSG+05], we expect excellent joint time-frequency concentration of g and γ to be the most important requirement for achieving that goal.

We will strive to customize decisions like the choice of discretization methods for channels with the system function properties summarized in Figure 3 (b) and for windows g and γ that are bandlimited and decay "fast enough" to be represented by a finite (and reasonably small) number of Nyquist frequency samples. This is the topic of sections 4.2 and 4.3.

4.2 Discretization tools

Recall first from Section 3.3 and Figure 3 (b) that it is justified to work with spreading functions that can be truncated to compact support with negligible truncation errors.

Then Proposition 2 below shows that the functions $Hg_{\boldsymbol{q},\boldsymbol{r}}$ and $\widehat{Hg_{\boldsymbol{q},\boldsymbol{r}}}$ are well localized around the lattice points of a lattice with the same lattice constants as the lattice of the Gabor basis $(g_{\boldsymbol{q},\boldsymbol{r}})$. This suggests to choose $(\gamma_{\boldsymbol{q}',\boldsymbol{r}'})$ to be a Gabor basis with the same lattice constants as $(g_{\boldsymbol{q},\boldsymbol{r}})$. For this scenario, propositions 3–5 in Section 4.3 provide us with formulas that allow an efficient computation of the matrix elements $\langle Hg_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q}',\boldsymbol{r}'} \rangle$.

Proposition 2. *Suppose that H is a Hilbert–Schmidt operator on $L^2(\mathbb{R}^d)$,*

$$\operatorname{supp} S_H \subseteq I_{\boldsymbol{\omega}_c,\boldsymbol{\omega}} \times [\boldsymbol{a},\boldsymbol{b}] \qquad \textit{and} \qquad g \in L^2(\mathbb{R}^d). \tag{4.3}$$

(a) If $\operatorname{supp} f \subseteq [\boldsymbol{A}, \boldsymbol{B}]$, *then* $\operatorname{supp} Hf \subseteq [\boldsymbol{A} + \boldsymbol{a}, \boldsymbol{B} + \boldsymbol{b}]$.

(b) If $\operatorname{supp} \widehat{f} \subseteq I_{\boldsymbol{C}_f, \boldsymbol{B_f}}$, *then* $\operatorname{supp} \widehat{Hg} \subseteq I_{\boldsymbol{C}_f + \boldsymbol{\omega}_c, \boldsymbol{B_f} + \boldsymbol{\omega}}$.

Proof. (a) follows immediately from (2.11b). *(b)*: For $S_H \in L^2(\mathbb{R}^d \times \mathbb{R}^d)$ and $f \in L^2(\mathbb{R}^d)$, a repeated use of the Hölder inequality gives that

$$\begin{aligned} |(Hf)(\boldsymbol{t_0})|^2 &\le \left(\int_{\mathbb{R}^d} \int_{\mathbb{R}^d} |S_H(\boldsymbol{\nu}, \boldsymbol{t})| \, \mathrm{d}\boldsymbol{\nu} \, |f(\boldsymbol{t_0} - \boldsymbol{t})| \, \mathrm{d}\boldsymbol{t} \right)^2 \\ &\le \|f\|_{L^2(\mathbb{R}^d)}^2 \int_{\mathbb{R}^d} \left(\int_{I_{\boldsymbol{\omega}_c, \boldsymbol{\omega}}} 1 \cdot |S_H(\boldsymbol{\nu}, \boldsymbol{t})| \, \mathrm{d}\boldsymbol{\nu} \right)^2 \mathrm{d}\boldsymbol{t} \\ &\le \|f\|_{L^2(\mathbb{R}^d)}^2 \, |\boldsymbol{\omega}| \cdot \|S_H\|_{L^2(\mathbb{R}^d \times \mathbb{R}^d)}^2 < \infty. \end{aligned}$$

Similarly, for *continuous* f *with compact support*, Fubini–Tonelli gives that

$$\begin{aligned} \widehat{Hf}(\boldsymbol{\xi}) &= \int_{\mathbb{R}^d} \int_{[\boldsymbol{\alpha}, \boldsymbol{\beta}]} \int_{I_{\boldsymbol{\omega}_c, \boldsymbol{\omega}}} S_H(\boldsymbol{\nu}, \boldsymbol{t}) \mathrm{e}^{\mathrm{i}2\pi \langle \boldsymbol{\nu}, \boldsymbol{x} - \boldsymbol{t} \rangle} \, \mathrm{d}\boldsymbol{\nu} \, f(\boldsymbol{t_0} - \boldsymbol{t}) \, \mathrm{d}\boldsymbol{t} \, \mathrm{e}^{-\mathrm{i}2\pi \langle \boldsymbol{x}, \boldsymbol{\xi} \rangle} \, \mathrm{d}\boldsymbol{t_0} \\ &= \int_{I_{\boldsymbol{\omega}_c, \boldsymbol{\omega}}} \int_{[\boldsymbol{\alpha}, \boldsymbol{\beta}]} \int_{\mathbb{R}^d} f(\boldsymbol{t_0} - \boldsymbol{t}) \mathrm{e}^{-\mathrm{i}2\pi \langle \boldsymbol{\xi} - \boldsymbol{\nu}, \boldsymbol{x} - \boldsymbol{t} \rangle} \, \mathrm{d}\boldsymbol{t_0} \, S_H(\boldsymbol{\nu}, \boldsymbol{t}) \mathrm{e}^{-\mathrm{i}2\pi \langle \boldsymbol{t}, \boldsymbol{\xi} \rangle} \, \mathrm{d}\boldsymbol{t} \, \mathrm{d}\boldsymbol{\nu} \\ &= \int_{I_{\boldsymbol{\omega}_c, \boldsymbol{\omega}}} \int_{[\boldsymbol{\alpha}, \boldsymbol{\beta}]} \widehat{f}(\boldsymbol{\xi} - \boldsymbol{\nu}) S_H(\boldsymbol{\nu}, \boldsymbol{t}) \mathrm{e}^{-\mathrm{i}2\pi \langle \boldsymbol{t}, \boldsymbol{\xi} \rangle} \, \mathrm{d}\boldsymbol{t} \, \mathrm{d}\boldsymbol{\nu} \\ &= \int_{I_{\boldsymbol{\omega}_c, \boldsymbol{\omega}}} \widehat{f}(\boldsymbol{\xi} - \boldsymbol{\nu}) \widehat{S_H(\boldsymbol{\nu}, \cdot)}(\boldsymbol{\xi}) \, \mathrm{d}\boldsymbol{\nu} = \int_{I_{\boldsymbol{\omega}_c, \boldsymbol{\omega}}} \widehat{f}(\boldsymbol{\xi} - \boldsymbol{\nu}) B_H(\boldsymbol{\nu}, \boldsymbol{\xi}) \, \mathrm{d}\boldsymbol{\nu}. \end{aligned} \tag{4.4}$$

By the Hölder inequality, the L^2-norm squared of the righthand side equals

$$\int_{\mathbb{R}^d} \left| \int_{I_{\boldsymbol{\omega}_c, \boldsymbol{\omega}}} \widehat{f}(\boldsymbol{\xi} - \boldsymbol{\nu}) B_H(\boldsymbol{\nu}, \boldsymbol{\xi}) \, \mathrm{d}\boldsymbol{\nu} \right|^2 \mathrm{d}\boldsymbol{\xi} \le \left\| \widehat{f} \right\|_2^2 \int_{\mathbb{R}^d} \int_{I_{\boldsymbol{\omega}_c, \boldsymbol{\omega}}} |B_H(\boldsymbol{\nu}, \boldsymbol{\xi})|^2 \, \mathrm{d}\boldsymbol{\nu} \, \mathrm{d}\boldsymbol{\xi}.$$

Since H is bounded and $B_H \in L^2(\mathbb{R}^d \times \mathbb{R}^d)$, both the lefthand and the righthand side of (4.4) depend continuously on f, so that (4.4) holds for all $f \in L^2(\mathbb{R}^d)$ by standard density arguments. Thus (4.3) implies that if $\operatorname{supp} \widehat{f} \subseteq I_{\boldsymbol{C}_f, \boldsymbol{B_f}}$, then $\operatorname{supp} \widehat{Hf} \subseteq I_{\boldsymbol{C}_f + \boldsymbol{\omega}_c, \boldsymbol{B_f} + \boldsymbol{\omega}}$. □

We will repeatedly apply the following special case of the Poisson Summation formula [Grö00, Kat04] to the functions $g_{\boldsymbol{q}, \boldsymbol{r}}$, $Hg_{\boldsymbol{q}, \boldsymbol{r}}$ and $\gamma_{\boldsymbol{q}', \boldsymbol{r}'}$:

Theorem 2 (Sampling theorems)**.** *Suppose that* $f \in L^2(\mathbb{R}^d)$ *has a Fourier transform with support* $\operatorname{supp} \widehat{f} \subseteq I_{\boldsymbol{C_f}, \boldsymbol{B_f}}$. *Then* $\widehat{f} \in L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d)$, f *is continuous (modulo modifications on a set of measure 0) and for all* $\boldsymbol{\xi} \in \widehat{\mathbb{R}^d}$,

$$\widehat{f}(\boldsymbol{\xi}) = |\boldsymbol{T_f}| \, \chi_{I_{\boldsymbol{C_f}, \boldsymbol{B_f}}}(\boldsymbol{\xi}) \sum_{\boldsymbol{k} \in \mathbb{Z}^d} f(\boldsymbol{k}\boldsymbol{T_f}) \mathrm{e}^{-\mathrm{i}2\pi \langle \boldsymbol{\xi}, \boldsymbol{k}\boldsymbol{T_f} \rangle}, \qquad \boldsymbol{T_f} = \frac{\boldsymbol{1}}{\boldsymbol{B_f}}. \tag{4.5a}$$

Equivalently (via (2.1)*),* (4.5a) *is also known as the* Nyquist sampling theorem

$$f(\boldsymbol{t}) = |\boldsymbol{T_f}| \sum_{\boldsymbol{k}\in\mathbb{Z}^d} f(\boldsymbol{kT_f}) \mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{C_f}, \boldsymbol{t}-\boldsymbol{kT_f}\rangle} \mathrm{sinc}_{\boldsymbol{B_f}}(\boldsymbol{t}-\boldsymbol{kT_f}). \tag{4.5b}$$

For $\boldsymbol{a} \in \mathbb{Z}^d$, some $\boldsymbol{b}, \boldsymbol{\Omega} \in \mathbb{R}^d_+$ and f equal to $g_{\boldsymbol{q},\boldsymbol{r}}$ or $\gamma_{\boldsymbol{q}',\boldsymbol{r}'}$, we will consider time-frequency shifts $f_{\boldsymbol{q},\boldsymbol{r}} \stackrel{\text{def}}{=} T_{\boldsymbol{raT_g}} M_{\boldsymbol{qb\Omega}} f$, for which

$$\boldsymbol{T_g} \stackrel{\text{def}}{=} \frac{1}{\boldsymbol{\Omega}}, \qquad \operatorname{supp} \widehat{f_{\boldsymbol{q},\boldsymbol{r}}}, \subseteq I_{\boldsymbol{C_f}+\boldsymbol{qb\Omega},\boldsymbol{B_f}}, \tag{4.6a}$$

and the Nyquist frequency samples are

$$f_{\boldsymbol{q},\boldsymbol{r}}(\boldsymbol{kT_f}) = f(\boldsymbol{kT_f} - \boldsymbol{raT_g}) \mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{kT_f}-\boldsymbol{raT_g}, \boldsymbol{qb\Omega}\rangle}. \tag{4.6b}$$

If $\boldsymbol{T_g} = \boldsymbol{T_f}$, then (4.6b) becomes

$$f_{\boldsymbol{q},\boldsymbol{r}}(\boldsymbol{kT_g}) = f((\boldsymbol{k}-\boldsymbol{ra})\boldsymbol{T_g}) \mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{k}-\boldsymbol{ra}, \boldsymbol{qb}\rangle}.$$

If $\boldsymbol{T_g} \neq \boldsymbol{T_f}$, then we can instead compute the samples $f(\boldsymbol{kT_f} - \boldsymbol{raT_g})$ by applying (4.5b), which will be a finite sum formula for those f that we will consider.

Proposition 2, Theorem 2, (4.6a) and (4.6b) are the basic tools that we will use to compute the matrix elements $\langle Hg_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q}',\boldsymbol{r}'}\rangle$. We will apply these to bandlimited g and γ that have only a finite number of nonzero samples in the Nyquist reconstruction formula (4.5b). These samples should be chosen so that g and γ decay fast enough to allow truncation to compact support with both the maximum and the L^2-norm of the truncation error being less than some ϵ well below the overall noise level of the application at hand. In Figure 7 on page 36, we show by example how such window functions can be constructed. For the scenario that we summarized in Figure 3 (b), we can, in practice, regard S_H to be compactly supported and conclude from Proposition 2 that $Hg_{\boldsymbol{q},\boldsymbol{r}}$ and $\widehat{Hg_{\boldsymbol{q},\boldsymbol{r}}}$ inherits the good localization properties of $g_{\boldsymbol{q},\boldsymbol{r}}$, $\widehat{g_{\boldsymbol{q},\boldsymbol{r}}}$ and S_H. Hence, with negligible truncation errors, we can assume also $Hg_{\boldsymbol{q},\boldsymbol{r}}$ to be bandlimited, to be fully described by a finite number of Nyquist frequency samples and to have ϵ-essential time-frequency support given by Proposition 2. (In Section 5 we choose $\epsilon = 10^{-6}$.) Throughout the paper, we shall use the following notation for these compact supports and finite index sets.

$$\operatorname{supp} \widehat{g} \subseteq I_{\boldsymbol{\Omega}_{\mathrm{c}},\boldsymbol{\Omega}}, \qquad \boldsymbol{T_g} \stackrel{\text{def}}{=} \tfrac{1}{\boldsymbol{\Omega}}, \qquad \boldsymbol{T_\gamma} \stackrel{\text{def}}{=} \tfrac{1}{\boldsymbol{\Omega}+\boldsymbol{\omega}}, \tag{4.7a}$$

$$\operatorname{supp} S_H \subseteq I_{\boldsymbol{\omega}_{\mathrm{c}},\boldsymbol{\omega}} \times I_{\boldsymbol{C},\boldsymbol{L}}, \qquad \operatorname{supp} \widehat{Hg} \subseteq \operatorname{supp} \widehat{\gamma} \subseteq I_{\boldsymbol{\Omega}_{\mathrm{c}}+\boldsymbol{\omega}_{\mathrm{c}},\boldsymbol{\Omega}+\boldsymbol{\omega}}, \tag{4.7b}$$

$$\mathcal{K}, \mathcal{M} \subset \mathbb{Z}^d, \qquad |\mathcal{K}| < \infty, \qquad |\mathcal{M}| < \infty \qquad \text{and} \tag{4.7c}$$

$$g(\boldsymbol{mT_g}) = \gamma(\boldsymbol{kT_\gamma}) = (Hg)(\boldsymbol{kT_\gamma}) = 0 \quad \text{for} \quad \boldsymbol{k} \in \mathbb{Z}^d \setminus \mathcal{K} \text{ and } \boldsymbol{m} \in \mathbb{Z}^d \setminus \mathcal{M}. \tag{4.7d}$$

The corresponding supports and index sets for $g_{\boldsymbol{q},\boldsymbol{r}}$, $Hg_{\boldsymbol{q},\boldsymbol{r}}$ and $\gamma_{\boldsymbol{q},\boldsymbol{r}}$ are easily computed from (4.6) above.

4.3 Computing the channel matrix

For spreading functions and window functions having the properties, supports and index sets given in (4.7) and Figure 3 (b), the following proposition provides us with a finite sum formula for computing the matrix elements $\langle Hg_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q}',\boldsymbol{r}'}\rangle$ from samples of $Hg_{\boldsymbol{q},\boldsymbol{r}}$ and $\gamma_{\boldsymbol{q}',\boldsymbol{r}'}$:

Proposition 3. *Let $u, v \in L^2(\mathbb{R}^d)$ be functions with compact, overlapping frequency supports satisfying*

$$\operatorname{supp}\widehat{u} \subseteq I_{\boldsymbol{C_u},\boldsymbol{B}}, \quad \operatorname{supp}\widehat{v} \subseteq I_{\boldsymbol{C_v},\boldsymbol{B}} \quad \textit{and} \quad I_{\boldsymbol{C_{uv}},\boldsymbol{B_{uv}}} \stackrel{\text{def}}{=} I_{\boldsymbol{C_u},\boldsymbol{B}} \cap I_{\boldsymbol{C_v},\boldsymbol{B}} \neq \emptyset.$$

For $\boldsymbol{T} = \frac{1}{\boldsymbol{B}}$ and $\boldsymbol{k} \in \mathbb{Z}^d$, suppose that $u(\boldsymbol{kT}) \neq 0$ and $v(\boldsymbol{kT}) \neq 0$ only for $\boldsymbol{k}$ in some finite index sets $\mathcal{I}_u$ and $\mathcal{I}_v$, respectively. Denote with v_{bpf} the inverse Fourier transform of the restriction (bandpass filtering) of $\widehat{v}$ to the support of $\widehat{u}$, that is,

$$\widehat{v_{\text{bpf}}}(\boldsymbol{\xi}) \stackrel{\text{def}}{=} \widehat{v}(\boldsymbol{\xi})\chi_{I_{\boldsymbol{C_{uv}},\boldsymbol{B_{uv}}}}(\boldsymbol{\xi}).$$

Then

$$\langle u, v\rangle_{L^2(\mathbb{R}^d)} = |\boldsymbol{T}| \sum_{\boldsymbol{k}\in\mathcal{I}_u} u(\boldsymbol{kT})\overline{v_{\text{bpf}}(\boldsymbol{kT})}. \tag{4.8a}$$

Further, $v_{\text{bpf}} = v$ if $\boldsymbol{C_u} = \boldsymbol{C_v}$ and, otherwise,

$$v_{\text{bpf}}(\boldsymbol{kT}) = |\boldsymbol{T}| \sum_{\boldsymbol{k'}\in\mathcal{I}_v} v(\boldsymbol{k'T})\mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{C_{uv}},(\boldsymbol{k}-\boldsymbol{k'})\boldsymbol{T}\rangle}\operatorname{sinc}_{\boldsymbol{B_{uv}}}((\boldsymbol{k}-\boldsymbol{k'})\boldsymbol{T}). \tag{4.8b}$$

Proof. (4.8a) follows from the Plancherel theorem and Parseval's identity:

$$\langle u, v\rangle_{L^2(\mathbb{R}^d)} = \langle \widehat{u}, \widehat{v}\rangle_{L^2(\mathbb{R}^d)} = \langle \widehat{u}, \widehat{v_{\text{bpf}}}\rangle_{L^2(I_{\boldsymbol{C_v},\boldsymbol{B}})} = |\boldsymbol{T}| \sum_{\boldsymbol{k}\in\mathcal{I}_u} u(\boldsymbol{kT})\overline{v_{\text{bpf}}(\boldsymbol{kT})}.$$

Moreover, the sampling theorem (4.5a) and (2.1) implies that

$$\begin{aligned} v_{\text{bpf}}(\boldsymbol{kT}) &= \int_{\mathbb{R}^d} \widehat{v}(\boldsymbol{\xi})\chi_{I_{\boldsymbol{C_{uv}},\boldsymbol{B_{uv}}}}(\boldsymbol{\xi})\mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\xi},\boldsymbol{kT}\rangle}\,\mathrm{d}\boldsymbol{\xi} \\ &= |\boldsymbol{T}| \sum_{\boldsymbol{k'}\in\mathcal{I}_v} v(\boldsymbol{k'T}) \int_{I_{\boldsymbol{C_{uv}},\boldsymbol{B_{uv}}}} \mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\xi},(\boldsymbol{k}-\boldsymbol{k'})\boldsymbol{T}\rangle}\,\mathrm{d}\boldsymbol{\xi} \\ &= |\boldsymbol{T}| \sum_{\boldsymbol{k'}\in\mathcal{I}_v} v(\boldsymbol{k'T})\mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{C_{uv}},(\boldsymbol{k}-\boldsymbol{k'})\boldsymbol{T}\rangle}\operatorname{sinc}_{\boldsymbol{B_{uv}}}((\boldsymbol{k}-\boldsymbol{k'})\boldsymbol{T}). \end{aligned}$$

□

Using (4.7) and (4.8a), we get, for example, that

$$\langle Hg, \gamma\rangle_{L^2(\mathbb{R}^d)} = |\boldsymbol{T_\gamma}| \sum_{\boldsymbol{k}\in\mathcal{K}} (Hg)(\boldsymbol{kT_\gamma})\overline{\gamma(\boldsymbol{kT_\gamma})}$$

and it remains only to derive efficient formulas for the computation of the samples $(Hg)(\boldsymbol{kT_\gamma})$. For this, we derive the finite sum formula (4.12) below with nonzero terms only for $\boldsymbol{m} \in \mathcal{M}$ when $f = g$ (and similarly for $f = g_{\boldsymbol{q},\boldsymbol{r}}$)[5]. It is clear that the following proof holds for arbitrary samples $Hf(\boldsymbol{t_0})$, but by setting $\boldsymbol{t_0} = \boldsymbol{kT_\gamma}$ we get our results in exactly the form that we shall need later on.

Proposition 4. *Suppose that* $\boldsymbol{C_f}, \boldsymbol{\Omega} \in \mathbb{R}^d$, $f \in L^2(\mathbb{R}^d)$, $\operatorname{supp} \widehat{f} \subseteq I_{\boldsymbol{C_f},\boldsymbol{\Omega}}$ *and*

$$(f(\boldsymbol{mT_f}))_{\boldsymbol{m}\in\mathbb{Z}^d} \in l^1(\mathbb{Z}^d), \qquad \text{with} \qquad \boldsymbol{T_f} \stackrel{\text{def}}{=} \frac{\boldsymbol{1}}{\boldsymbol{\Omega}}. \tag{4.9}$$

Let H be a Hilbert–Schmidt operator with spreading function

$$S_H \in C^{(\infty)} \cap L^2(\mathbb{R}^d) \quad \text{and} \quad \operatorname{supp} S_H \subseteq I_{\boldsymbol{\omega}_c,\boldsymbol{\omega}} \times I_{\boldsymbol{C},\boldsymbol{L}}. \tag{4.10a}$$

Define $S_H^{\boldsymbol{C_f},\boldsymbol{\Omega}}$ to be the convolution

$$S_H^{\boldsymbol{C_f},\boldsymbol{\Omega}}(\boldsymbol{\nu},\boldsymbol{t_0}) \stackrel{\text{def}}{=} \left(S_H(\boldsymbol{\nu},\cdot) * (\mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{C_f}+\boldsymbol{\nu},\cdot\rangle}\operatorname{sinc}_{\boldsymbol{\Omega}}(\cdot))\right)(\boldsymbol{t_0}). \tag{4.10b}$$

Then for all $\boldsymbol{k} \in \mathbb{Z}^d$ and $\boldsymbol{T_\gamma} \in \mathbb{R}_+^d$, we have

$$\begin{aligned}
&(Hf)(\boldsymbol{kT_\gamma}) = \\
&= |\boldsymbol{T_f}| \int_{\mathbb{R}^d} \sum_{\boldsymbol{r}\in\mathbb{Z}^d} S_H^{\boldsymbol{C_f},\boldsymbol{\Omega}}(\boldsymbol{rT_f} + \boldsymbol{k}(\boldsymbol{T_\gamma} - \boldsymbol{T_f}), \boldsymbol{\nu})(T_{\boldsymbol{rT_f}} M_{\boldsymbol{\nu}} f)(\boldsymbol{kT_f})\, d\boldsymbol{\nu} \qquad (4.11)\\
&= |\boldsymbol{T_f}| \sum_{\boldsymbol{m}\in\mathbb{Z}^d} f(\boldsymbol{mT_f}) \left(S_H^{\boldsymbol{C_f},\boldsymbol{\Omega}}(\cdot, \boldsymbol{kT_\gamma} - \boldsymbol{mT_f})\right)\widehat{\ }(-\boldsymbol{mT_f}). \qquad (4.12)
\end{aligned}$$

Proof. From (2.11b) and the Fubini–Tonelli theorem,

$$\begin{aligned}
(Hf)(\boldsymbol{t_0}) &= \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} S_H(\boldsymbol{\nu},\boldsymbol{t}) f(\boldsymbol{t_0} - \boldsymbol{t}) \mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\nu},\boldsymbol{t_0}-\boldsymbol{t}\rangle}\, \mathrm{d}\boldsymbol{t}\, \mathrm{d}\boldsymbol{\nu} \\
&= \int_{\mathbb{R}^d} \left(S_H(\boldsymbol{\nu},\cdot) * (f(\cdot)\mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\nu},\cdot\rangle})\right)(\boldsymbol{t_0})\, \mathrm{d}\boldsymbol{\nu} \stackrel{\text{def}}{=} \int_{\mathbb{R}^d} f_{\boldsymbol{\nu}}(\boldsymbol{t_0})\, \mathrm{d}\boldsymbol{\nu}, \qquad (4.13)
\end{aligned}$$

where the Nyquist sampling theorem (4.5b) and then (2.1) gives that

$$\begin{aligned}
f_{\boldsymbol{\nu}}(\boldsymbol{t_0}) &= |\boldsymbol{T_f}| \sum_{\boldsymbol{m}\in\mathbb{Z}^d} f(\boldsymbol{mT_f}) \left(S_H(\boldsymbol{\nu},\cdot) * (\mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{C_f},\cdot-\boldsymbol{mT_f}\rangle}\operatorname{sinc}_{\boldsymbol{\Omega}}(\cdot - \boldsymbol{mT_f})\mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\nu},\cdot\rangle})\right)(\boldsymbol{t_0}) \\
&= |\boldsymbol{T_f}| \sum_{\boldsymbol{m}\in\mathbb{Z}^d} f(\boldsymbol{mT_f})\mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\nu},\boldsymbol{mT_f}\rangle} \left(S_H(\boldsymbol{\nu},\cdot) * (\mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{C_f}+\boldsymbol{\nu},\cdot\rangle}\operatorname{sinc}_{\boldsymbol{\Omega}}(\cdot))\right)(\boldsymbol{t_0} - \boldsymbol{mT_f}).
\end{aligned}$$

[5] In (4.12) it is necessary for f to provide the finite summation index, since there can be a finite number of nonzero $S_H^{\boldsymbol{C_f},\Omega}(\cdot, \boldsymbol{kT_\gamma} - \boldsymbol{mT_g})$ and $(Hf)(\boldsymbol{kT_\gamma})$ (without alias effects due to undersampling) only if $\frac{\boldsymbol{T_\gamma}}{\boldsymbol{T_g}} \in \mathbb{Z}^d$ and $\boldsymbol{T_\gamma} = \frac{\boldsymbol{1}}{\boldsymbol{\Omega}+\boldsymbol{\omega}}$, that is, only if $\boldsymbol{\omega} = \boldsymbol{0}$.

Insertion of this in (4.13) gives

$$\begin{aligned}(Hf)(\boldsymbol{kT_\gamma}) &= \\ &= |\boldsymbol{T_f}| \int_{\mathbb{R}^d} \sum_{\boldsymbol{m}\in\mathbb{Z}^d} f(\boldsymbol{mT_f}) \mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{\nu}, \boldsymbol{mT_f}\rangle} S_H^{\boldsymbol{C_f},\boldsymbol{\Omega}}(\boldsymbol{\nu}, \boldsymbol{kT_\gamma} - \boldsymbol{mT_f})\,\mathrm{d}\boldsymbol{\nu} \qquad (4.14)\\ &= |\boldsymbol{T_f}| \int_{\mathbb{R}^d} \sum_{\boldsymbol{r}\in\mathbb{Z}^d} f((\boldsymbol{k}-\boldsymbol{r})\boldsymbol{T_f}) \mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{\nu}, (\boldsymbol{k}-\boldsymbol{r})\boldsymbol{T_f}\rangle} S_H^{\boldsymbol{C_f},\boldsymbol{\Omega}}(\boldsymbol{\nu}, \boldsymbol{rT_f} + \boldsymbol{k}(\boldsymbol{T_\gamma}-\boldsymbol{T_f}))\,\mathrm{d}\boldsymbol{\nu}.\end{aligned}$$

This proves (4.11). By (4.10a) and (4.10b), $S_H^{\boldsymbol{C_f},\boldsymbol{\Omega}}$ is bounded. Hence (4.9), (4.14) and the Fubini–Tonelli theorem provide

$$\begin{aligned}(Hf)(\boldsymbol{kT_\gamma}) &= |\boldsymbol{T_f}| \sum_{\boldsymbol{m}\in\mathbb{Z}^d} f(\boldsymbol{mT_f}) \int_{I_{\boldsymbol{\omega}_\mathrm{c},\boldsymbol{\omega}}} \mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{\nu}, \boldsymbol{mT_f}\rangle} S_H^{\boldsymbol{C_f},\boldsymbol{\Omega}}(\boldsymbol{\nu}, \boldsymbol{kT_\gamma} - \boldsymbol{mT_f})\,\mathrm{d}\boldsymbol{\nu} \\ &= |\boldsymbol{T_f}| \sum_{\boldsymbol{m}\in\mathbb{Z}^d} f(\boldsymbol{mT_f}) \Big(S_H^{\boldsymbol{C_f},\boldsymbol{\Omega}}(\cdot, \boldsymbol{kT_\gamma} - \boldsymbol{mT_f})\Big)\widehat{\ }(-\boldsymbol{mT_f}),\end{aligned}$$

which proves (4.12) and concludes the proof. □

Next, we shall derive the finite sum formula (4.16) for computing the samples of $(S_H^{\boldsymbol{\Omega_c}+\boldsymbol{qb\Omega},\boldsymbol{\Omega}}(\cdot,\boldsymbol{t}))\widehat{\ }$ that are needed in (4.12) to compute $(Hg_{\boldsymbol{q},\boldsymbol{r}})(\boldsymbol{kT_\gamma})$ for a Gabor system

$$g_{\boldsymbol{q},\boldsymbol{r}} \stackrel{\text{def}}{=} T_{\boldsymbol{raT_g}} M_{\boldsymbol{qb\Omega}}\, g, \qquad (\boldsymbol{q},\boldsymbol{r}) \in \mathcal{Q}\times\mathcal{R}$$

for a finite index set $\mathcal{Q}\times\mathcal{R} \subseteq \mathbb{Z}^{2d}$. Equations (4.10b), (4.7a) and (4.7b) imply that $S_H^{\boldsymbol{\Omega_c}+\boldsymbol{qb\Omega},\boldsymbol{\Omega}}(\boldsymbol{\nu},\cdot)$ is obtained from bandpass filtering $S_H(\boldsymbol{\nu},\cdot)$ to the frequency

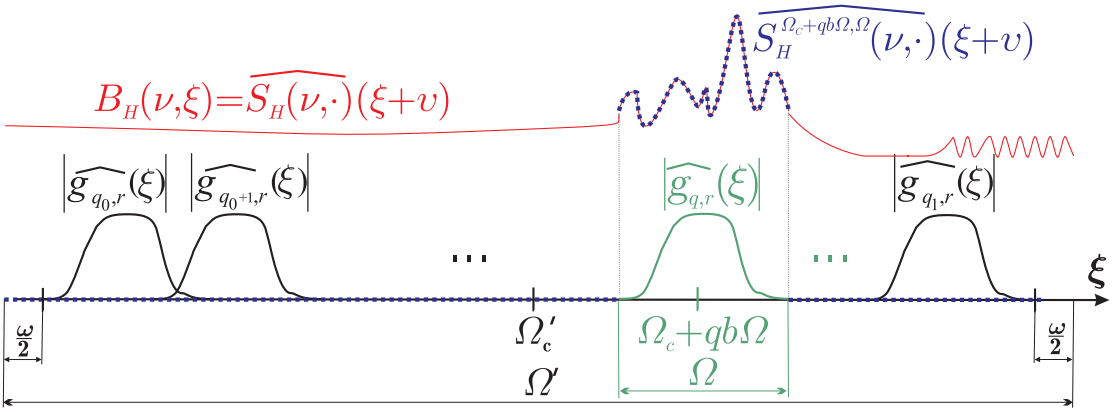


Figure 5: For Gabor frames $g_{\boldsymbol{q},\boldsymbol{r}} \stackrel{\text{def}}{=} T_{\boldsymbol{raT_g}} M_{\boldsymbol{qb\Omega}} g$ with $\boldsymbol{q} \in \mathcal{Q} \stackrel{\text{def}}{=} [\boldsymbol{q_0}, \boldsymbol{q_1}] \cap \mathbb{Z}^d$, the function $|\widehat{g_{\boldsymbol{q},\boldsymbol{r}}}|$ depends only on $\boldsymbol{q}$. Proposition 4 shows how to compute all $Hg_{\boldsymbol{q},\boldsymbol{r}}$ only from the restriction of $\widehat{S_H(\boldsymbol{\nu},\cdot)}$ to any interval $I_{\boldsymbol{\Omega_c'},\boldsymbol{\Omega'}}$ containing the supports of all $\widehat{g_{\boldsymbol{q},\boldsymbol{r}}}(\cdot-\boldsymbol{\nu})$ for all $\boldsymbol{\nu} \in I_{\boldsymbol{\omega}_\mathrm{c},\boldsymbol{\omega}}$, as is illustrated here for minimal Ω' and $d=1$.

band

$$\operatorname{supp} \widehat{g_{\boldsymbol{q},\boldsymbol{r}}}(\cdot - \boldsymbol{\nu}) = I_{\boldsymbol{\Omega}_\mathrm{c}+\boldsymbol{q}b\boldsymbol{\Omega}+\boldsymbol{\nu},\boldsymbol{\Omega}}, \qquad \text{where} \qquad \boldsymbol{\nu} \in \operatorname{supp} S_H^{\boldsymbol{\Omega}_\mathrm{c}+\boldsymbol{q}b\boldsymbol{\Omega},\boldsymbol{\Omega}}(\cdot,\boldsymbol{t}) \subseteq I_{\boldsymbol{\omega}_\mathrm{c},\boldsymbol{\omega}}.$$

Hence, we can compute all $S_H^{\boldsymbol{\Omega}_\mathrm{c}+\boldsymbol{q}b\boldsymbol{\Omega},\boldsymbol{\Omega}}$ from the restriction of $\widehat{S_H(\boldsymbol{\nu},\cdot)}$ to the smallest interval $I_{\boldsymbol{\Omega}'_\mathrm{c},\boldsymbol{\Omega}'}$ containing the union of the supports of all $\widehat{g_{\boldsymbol{q},\boldsymbol{r}}}(\cdot - \boldsymbol{\nu})$ for all $\boldsymbol{\nu} \in I_{\boldsymbol{\omega}_\mathrm{c},\boldsymbol{\omega}}$, as illustrated for $d = 1$ in Figure 5. This means that if $\mathcal{Q} = [\boldsymbol{q_0}, \boldsymbol{q_1}] \cap \mathbb{Z}^d$, then

$$\boldsymbol{\Omega}'_\mathbf{c} \stackrel{\text{def}}{=} \boldsymbol{\Omega}_\mathbf{c} + \frac{\boldsymbol{q_0} + \boldsymbol{q_1}}{2} b\boldsymbol{\Omega} + \boldsymbol{\omega}_\mathrm{c} \qquad \text{and} \qquad \boldsymbol{\Omega}' \stackrel{\text{def}}{=} (\boldsymbol{q_1} - \boldsymbol{q_0})b\boldsymbol{\Omega} + \boldsymbol{\Omega} + \boldsymbol{\omega}. \quad (4.15a)$$

This observation allows for a two-step discretization based on the system function properties summarized in Figure 3 (b), namely:

First, recall from Figure 3 (b) that h has compact support which is contained in some "box" $I_{\boldsymbol{C_0},\boldsymbol{L_0}} \times I_{\boldsymbol{C},\boldsymbol{L}}$, where $I_{\boldsymbol{C},\boldsymbol{L}}$ contains the support of $S_H(\boldsymbol{\nu},\cdot)$ for all $\boldsymbol{\nu}$. Recall also that due to the smooth truncation in Section 3.3, $h(\cdot,\boldsymbol{t})$ is supported in an interval $I_{\boldsymbol{C_0},\boldsymbol{L_0}}$, which is large enough not to affect the channel output in the time interval under consideration. This together with the impulse response integral representation (2.11c) of H gives that $\operatorname{supp} h(\cdot,\boldsymbol{t}) \subseteq I_{\boldsymbol{C_0},\boldsymbol{L_0}}$ must be large enough to contain the support of $Hg_{\boldsymbol{q},\boldsymbol{r}}$ for all $r \in \mathcal{R}$. Using element-wise addition of sets, this condition has the form

$$\operatorname{supp} g + \mathcal{R}\boldsymbol{a}\boldsymbol{T_g} + I_{\boldsymbol{C},\boldsymbol{L}} \subseteq I_{\boldsymbol{C_0},\boldsymbol{L_0}}. \quad (4.15b)$$

It follows that

$$\operatorname{supp} \widehat{S_H(\cdot,\boldsymbol{t})}(\boldsymbol{t_0}) = \operatorname{supp} h(\boldsymbol{t}-\boldsymbol{t_0},\boldsymbol{t}) = \left\{(\boldsymbol{t_0},\boldsymbol{t}) \in \mathbb{R}^{2d} : \boldsymbol{t} \in I_{\boldsymbol{C},\boldsymbol{L}} \text{ and } \boldsymbol{t_0} \in I_{\boldsymbol{t}-\boldsymbol{C_0},\boldsymbol{L_0}}\right\}.$$

Due to this compact support set and the subexponential decay of $S_H(\cdot,\boldsymbol{t})$, we can apply and truncate the Nyquist sampling theorem (4.5b) to

$$\begin{aligned} S_H(\boldsymbol{\nu},\boldsymbol{t}) &= |\boldsymbol{\omega_0}| \sum_{\boldsymbol{n}\in\mathcal{N}} S_H(\boldsymbol{n\omega_0},\boldsymbol{t}) \mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{t}-\boldsymbol{C_0},\boldsymbol{\nu}-\boldsymbol{n\omega_0}\rangle} \operatorname{sinc}_{\boldsymbol{L_0}}(\boldsymbol{\nu} - \boldsymbol{n\omega_0}), \\ &|\mathcal{N}| < \infty \qquad \text{and} \qquad \boldsymbol{\omega}_0 \stackrel{\text{def}}{=} \frac{1}{\boldsymbol{L_0}}. \end{aligned} \quad (4.15c)$$

Second, we shall use that only the restriction of $\widehat{S_H(\boldsymbol{\nu},\cdot)} = B_H(\boldsymbol{\nu},\cdot - \boldsymbol{\nu})$ to $I_{\boldsymbol{\Omega}'_\mathrm{c},\boldsymbol{\Omega}'}$ is of importance for our computations. Since $S_H(\boldsymbol{\nu},\cdot)$ has compact support, we know from Section 2.2 that there is a smooth truncation of $\widehat{S_H(\boldsymbol{\nu},\cdot)}$ to a $C^{(\infty)}$ function $\widehat{S^{\boldsymbol{\Omega}''}_{\boldsymbol{\Omega}''_\mathrm{c}}(\boldsymbol{\nu},\cdot)}$ such that

$$\widehat{S^{\boldsymbol{\Omega}''}_{\boldsymbol{\Omega}''_\mathrm{c}}(\boldsymbol{\nu},\boldsymbol{\cdot})} = \widehat{S_H(\boldsymbol{\nu},\boldsymbol{\cdot})} \text{ in } I_{\boldsymbol{\Omega}'_\mathrm{c},\boldsymbol{\Omega}'}, \qquad \operatorname{supp} \widehat{S^{\boldsymbol{\Omega}''}_{\boldsymbol{\Omega}''_\mathrm{c}}(\boldsymbol{\nu},\cdot)} \subseteq I_{\boldsymbol{\Omega}''_\mathrm{c},\boldsymbol{\Omega}''} \quad (4.15d)$$

with $I_{\boldsymbol{\Omega}''_\mathrm{c},\boldsymbol{\Omega}''}$ compact and such that

$$S^{\boldsymbol{\Omega}''}_{\boldsymbol{\Omega}''_\mathrm{c}}(\boldsymbol{\nu},\cdot) \text{ decays subexponentially.} \quad (4.15e)$$

Hence we can apply the sampling theorem (4.5a) to $\widehat{S^{\Omega''}_{\Omega''_{\mathbf{c}}}(\boldsymbol{\nu},\cdot)}$ and truncate it with no or negligible error to a finite sum

$$\begin{aligned}\widehat{S^{\Omega''}_{\Omega''_{\mathbf{c}}}(\boldsymbol{\nu},\cdot)}(\boldsymbol{\xi}) &= |\boldsymbol{T''}|\,\chi_{I_{\Omega''_{\mathbf{c}},\Omega''}}(\boldsymbol{\xi})\sum_{\boldsymbol{p}\in\mathcal{P}} S^{\Omega''}_{\Omega''_{\mathbf{c}}}(\boldsymbol{\nu},\boldsymbol{pT''})\mathrm{e}^{-\mathrm{i}2\pi\langle\boldsymbol{\xi},\boldsymbol{pT''}\rangle},\\ |\mathcal{P}|<&\infty \qquad \text{and} \qquad \boldsymbol{T''}\overset{\text{def}}{=}\frac{\mathbf{1}}{\boldsymbol{\Omega''}}.\end{aligned} \tag{4.15f}$$

We are now ready to state our final proposition.

Proposition 5. *For the functions and supports defined in* (4.7), (4.10b) *and* (4.15), *set* $\boldsymbol{\Omega_{c,q}}\overset{\text{def}}{=}\boldsymbol{\Omega_c}+\boldsymbol{qb\Omega}$ *for all* $\boldsymbol{q}\in\mathcal{Q}$. *Then*

$$\begin{aligned}\widehat{S_H^{\boldsymbol{\Omega_{c,q}},\boldsymbol{\Omega}}}(\cdot,\boldsymbol{t})(\boldsymbol{t_0}) = |\boldsymbol{\omega_0 T''}|\,\chi_{I_{\boldsymbol{C_0},\boldsymbol{L_0}}}(\boldsymbol{t}-\boldsymbol{t_0})\sum_{\boldsymbol{p}\in\mathcal{P}}\mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\Omega_{c,q}},\boldsymbol{t}-\boldsymbol{pT''}\rangle}\mathrm{sinc}_{\boldsymbol{\Omega}}(\boldsymbol{t}-\boldsymbol{pT''})\times\\ \times\sum_{\boldsymbol{n}\in\mathcal{N}} S^{\Omega''}_{\Omega''_{\mathbf{c}}}(\boldsymbol{n\omega_0},\boldsymbol{pT''})\mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{t}-\boldsymbol{t_0}-\boldsymbol{pT''},\boldsymbol{n\omega_0}\rangle}.\end{aligned} \tag{4.16}$$

Proof. Note first that if $\boldsymbol{\xi}\in I_{\boldsymbol{\Omega_{c,q}}+\boldsymbol{\nu},\boldsymbol{\Omega}}$, then for all $\boldsymbol{n\omega_0}\in I_{\boldsymbol{\omega}_{\mathrm{c}},\boldsymbol{\omega}}$,

$$\boldsymbol{\xi}-(\boldsymbol{\nu}-\boldsymbol{n\omega_0})\in I_{\boldsymbol{\Omega_{c,q}}+\boldsymbol{n\omega_0},\boldsymbol{\Omega}}\subseteq I_{\boldsymbol{\Omega}'_{\mathbf{c}},\boldsymbol{\Omega}'},$$

so that (4.15d) implies that

$$\widehat{S_H(\boldsymbol{\nu},\cdot)}(\boldsymbol{\xi}-(\boldsymbol{\nu}-\boldsymbol{n\omega_0}))\chi_{I_{\boldsymbol{\Omega_{c,q}}+\boldsymbol{\nu},\boldsymbol{\Omega}}}(\boldsymbol{\xi}) = \widehat{S^{\Omega''}_{\Omega''_{\mathbf{c}}}(\boldsymbol{\nu},\cdot)}(\boldsymbol{\xi}-(\boldsymbol{\nu}-\boldsymbol{n\omega_0}))\chi_{I_{\boldsymbol{\Omega_{c,q}}+\boldsymbol{\nu},\boldsymbol{\Omega}}}(\boldsymbol{\xi}).$$

Hence, it follows from (4.15), the definition (4.10b) and (2.1) that for all $\boldsymbol{\nu}\in I_{\boldsymbol{\omega}_{\mathrm{c}},\boldsymbol{\omega}}$,

$$\widehat{S_H^{\boldsymbol{\Omega_{c,q}},\boldsymbol{\Omega}}}(\boldsymbol{\nu},\cdot)(\boldsymbol{\xi}) = \widehat{S_H(\boldsymbol{\nu},\cdot)}(\boldsymbol{\xi})\chi_{I_{\boldsymbol{\Omega_{c,q}}+\boldsymbol{\nu},\boldsymbol{\Omega}}}(\boldsymbol{\xi})$$

$$\begin{aligned}&= |\boldsymbol{\omega_0}|\sum_{\boldsymbol{n}\in\mathcal{N}}\widehat{S_H(\boldsymbol{n\omega_0},\cdot)}(\boldsymbol{\xi}-(\boldsymbol{\nu}-\boldsymbol{n\omega_0}))\chi_{I_{\boldsymbol{\Omega_{c,q}}+\boldsymbol{\nu},\boldsymbol{\Omega}}}(\boldsymbol{\xi})\mathrm{e}^{-\mathrm{i}2\pi\langle\boldsymbol{C_0},\boldsymbol{\nu}-\boldsymbol{n\omega_0}\rangle}\mathrm{sinc}_{\boldsymbol{L_0}}(\boldsymbol{\nu}-\boldsymbol{n\omega_0}) =\\ &= |\boldsymbol{\omega_0 T''}|\sum_{\boldsymbol{n}\in\mathcal{N}}\sum_{\boldsymbol{p}\in\mathcal{P}} S^{\Omega''}_{\Omega''_{\mathbf{c}}}(\boldsymbol{n\omega_0},\boldsymbol{pT''})\mathrm{e}^{-\mathrm{i}2\pi\langle\boldsymbol{\xi}-(\boldsymbol{\nu}-\boldsymbol{n\omega_0}),\boldsymbol{pT''}\rangle}\chi_{I_{\boldsymbol{\Omega_{c,q}}+\boldsymbol{\nu},\boldsymbol{\Omega}}}(\boldsymbol{\xi})\times\\ &\qquad\times\mathrm{e}^{-\mathrm{i}2\pi\langle\boldsymbol{C_0},\boldsymbol{\nu}-\boldsymbol{n\omega_0}\rangle}\mathrm{sinc}_{\boldsymbol{L_0}}(\boldsymbol{\nu}-\boldsymbol{n\omega_0})\end{aligned}$$

Inverse Fourier transformation and (2.1) again gives

$$\begin{aligned}S_H^{\boldsymbol{\Omega_{c,q}},\boldsymbol{\Omega}}(\boldsymbol{\nu},\boldsymbol{t}) &= |\boldsymbol{\omega_0 T''}|\sum_{\boldsymbol{n}\in\mathcal{N}}\sum_{\boldsymbol{p}\in\mathcal{P}} S^{\Omega''}_{\Omega''_{\mathbf{c}}}(\boldsymbol{n\omega_0},\boldsymbol{pT''})\int_{I_{\boldsymbol{\Omega_{c,q}}+\boldsymbol{\nu},\boldsymbol{\Omega}}}\mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\xi},\boldsymbol{t}-\boldsymbol{pT''}\rangle}\,\mathrm{d}\boldsymbol{\xi}\times\\ &\qquad\times\mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{pT''}-\boldsymbol{C_0},\boldsymbol{\nu}-\boldsymbol{n\omega_0}\rangle}\mathrm{sinc}_{\boldsymbol{L_0}}(\boldsymbol{\nu}-\boldsymbol{n\omega_0})\\ &= |\boldsymbol{\omega_0 T''}|\sum_{\boldsymbol{n}\in\mathcal{N}}\sum_{\boldsymbol{p}\in\mathcal{P}} S^{\Omega''}_{\Omega''_{\mathbf{c}}}(\boldsymbol{n\omega_0},\boldsymbol{pT''})\mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\Omega_{c,q}}+\boldsymbol{\nu},\boldsymbol{t}-\boldsymbol{pT''}\rangle}\times\\ &\qquad\times\mathrm{sinc}_{\boldsymbol{\Omega}}(\boldsymbol{t}-\boldsymbol{pT''})\mathrm{e}^{-\mathrm{i}2\pi\langle\boldsymbol{pT''}-\boldsymbol{C_0},\boldsymbol{n\omega_0}\rangle}\mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{pT''}-\boldsymbol{C_0},\boldsymbol{\nu}\rangle}\mathrm{sinc}_{\boldsymbol{L_0}}(\boldsymbol{\nu}-\boldsymbol{n\omega_0}).\end{aligned}$$

Since $\mathrm{sinc}_{\boldsymbol{L_0}}(\cdot - \boldsymbol{n\omega_0})$ is the inverse Fourier transform of $\chi_{I_{\mathbf{0},\boldsymbol{L_0}}}(\cdot)\mathrm{e}^{-\mathrm{i}2\pi\langle\cdot,\boldsymbol{n\omega_0}\rangle}$,

$$
\begin{aligned}
&\widehat{S_H^{\boldsymbol{\Omega_{c,q}},\boldsymbol{\Omega}}(\cdot,\boldsymbol{t})}(\boldsymbol{t_0}) \\
&= |\boldsymbol{\omega_0 T''}| \sum_{\boldsymbol{n}\in\mathcal{N}} \sum_{\boldsymbol{p}\in\mathcal{P}} S_{\boldsymbol{\Omega_c''}}^{\boldsymbol{\Omega''}}(\boldsymbol{n\omega_0}, \boldsymbol{pT''}) \mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\Omega_{c,q}},\boldsymbol{t}-\boldsymbol{pT''}\rangle} \mathrm{sinc}_{\boldsymbol{\Omega}}(\boldsymbol{t}-\boldsymbol{pT''}) \times \\
&\qquad\qquad \times \int_{\mathbb{R}^d} \mathrm{e}^{-\mathrm{i}2\pi\langle\boldsymbol{\nu},\boldsymbol{t_0}+\boldsymbol{C_0}-\boldsymbol{t}\rangle} \mathrm{sinc}_{\boldsymbol{L_0}}(\boldsymbol{\nu}-\boldsymbol{n\omega_0})\,\mathrm{d}\boldsymbol{\nu} \mathrm{e}^{-\mathrm{i}2\pi\langle\boldsymbol{pT''}-\boldsymbol{C_0},\boldsymbol{n\omega_0}\rangle} \\
&= |\boldsymbol{\omega_0 T''}| \sum_{\boldsymbol{n}\in\mathcal{N}} \sum_{\boldsymbol{p}\in\mathcal{P}} S_{\boldsymbol{\Omega_c''}}^{\boldsymbol{\Omega''}}(\boldsymbol{n\omega_0}, \boldsymbol{pT''}) \mathrm{e}^{\mathrm{i}2\pi\langle\boldsymbol{\Omega_{c,q}},\boldsymbol{t}-\boldsymbol{pT''}\rangle} \mathrm{sinc}_{\boldsymbol{\Omega}}(\boldsymbol{t}-\boldsymbol{pT''}) \mathrm{e}^{-\mathrm{i}2\pi\langle\boldsymbol{pT''}-\boldsymbol{C_0},\boldsymbol{n\omega_0}\rangle} \times \\
&\qquad\qquad \times \chi_{I_{\mathbf{0},\boldsymbol{L_0}}}(\boldsymbol{t_0}+\boldsymbol{C_0}-\boldsymbol{t}) \mathrm{e}^{-\mathrm{i}2\pi\langle\boldsymbol{t_0}+\boldsymbol{C_0}-\boldsymbol{t},\boldsymbol{n\omega_0}\rangle}.
\end{aligned}
$$

This proves (4.16). □

REMARK. For physical wireless communications channels, we deduced a spreading function integral representation (3.9c) of the channel which is valid for functions with frequency localization "near ξ_0". For the OFDM and Satellite communications examples in Section 6, this assumption holds for the entire frequency band $I_{\Omega_c',\Omega'}$, since Ω'/Ω_c' is of the size $3 \cdot 10^{-3}$ and $2 \cdot 10^{-3}$, respectively. Then, the coefficients $S_{\boldsymbol{\Omega_c''}}^{\boldsymbol{\Omega''}}(\boldsymbol{n\omega_0}, \boldsymbol{pT''})$ above characterize the channel throughout this frequency band. In the underwater communications example, however, this is not the case, but the above theory can still be applied to each computed $Hg_{\boldsymbol{q,r}}$ as long as the relative bandwidth Ω of $g_{q,r}$ is much smaller than the carrier frequency $\Omega_{c,q} \stackrel{\text{def}}{=} \Omega_c + qb\Omega$. Normally we also want Ω to be larger than the maximum Doppler shift $|\nu_P \xi| = \left|\frac{V_P}{V_w}\xi\right|$ of (3.2), so as a rule of thumb $|V_P/V_w|$ should be very small, so that a larger but still small Ω/ξ can be chosen. In this case, and provided that the attenuation factor $A_\xi(P)$ in (3.3) is slowly varying with ξ, the following modifications allow for a "wideband use" of the propositions in this section:

1. Equation (3.9c) splits into separate operators H_q for basis functions $g_{q,r}$ with center frequency $\Omega_{c,q}$. For some compact sets $K_q \subseteq (-\Omega_{c,q}, \Omega_{c,q})$, these operators have integral representation

$$
\begin{aligned}
H_q g_{q,r}(\cdot) &= \int_{K_q \times [A,\infty)} S_{H_q}(\nu', t)(T_t M_{\nu'} g_{q,r})(\cdot)\,\mathrm{d}(\nu', t), \\
S_{H_q}(\nu', t) &\stackrel{\text{def}}{=} \frac{1}{\Omega_{\mathbf{c},q}} r_{\xi_0}\left(\frac{\nu'}{\Omega_{\mathbf{c},q}}, t\right) \mathrm{e}^{-\alpha_{\xi_0} t} \chi_{K\times[A,\infty)}\left(\frac{\nu'}{\Omega_{\mathbf{c},q}}, t\right).
\end{aligned}
$$

2. A straightforward multivariate extension of (3.9c) gives an operator spreading function S_H for signals with frequency localization near some fixed center frequency $\boldsymbol{\xi_0}$. Then the operator's action on narrowband functions $g_{\boldsymbol{q,r}}$ with center frequency $\boldsymbol{\Omega_{c,q}} \stackrel{\text{def}}{=} \boldsymbol{\Omega_c} + \boldsymbol{qb\Omega}$ is given by a spreading function

$$
S_{H_{\boldsymbol{q}}}(\boldsymbol{\nu}, \boldsymbol{t}) \stackrel{\text{def}}{=} \left|\frac{\boldsymbol{\xi_0}}{\boldsymbol{\Omega_{c,q}}}\right| S_H\left(\frac{\boldsymbol{\xi_0}}{\boldsymbol{\Omega_{c,q}}}\boldsymbol{\nu}, \boldsymbol{t}\right) \tag{4.17}
$$

3. With $\boldsymbol{\xi_0}$ chosen so that all $[\mathbf{0}, \boldsymbol{\Omega_{c,q}}] \subseteq [\mathbf{0}, \boldsymbol{\xi_0}]$, we keep the notation (4.7), now with $I_{\boldsymbol{\omega_c,\omega}} \times I_{\boldsymbol{C,L}}$ being the smallest interval containing $\operatorname{supp} S_{H_q}$ for all $\boldsymbol{q} \in \mathcal{Q}$.

4. These changes do not affect Propositions 2–4, but if $\widehat{S_H(\cdot, \boldsymbol{t})}$ has the support given by (4.15b), then the dilation with a factor $\frac{\boldsymbol{\xi_0}}{\boldsymbol{\Omega_{c,q}}}$ in (5) gives that $\widehat{S_{H_q}(\cdot, \boldsymbol{t})}$ has its support in
$$I_{\boldsymbol{C_{0,q}, L_{0,q}}} \stackrel{\text{def}}{=} I_{\frac{\boldsymbol{\xi_0}}{\boldsymbol{\Omega_{c,q}}}\boldsymbol{C_0}, \frac{\boldsymbol{\xi_0}}{\boldsymbol{\Omega_{c,q}}}\boldsymbol{L_0}}.$$

5. Hence we also know from (4.17) that if $\boldsymbol{\omega_{0,q}} \stackrel{\text{def}}{=} \frac{1}{\boldsymbol{L_{0,q}}}$, then
$$|\boldsymbol{\omega_{0,q}}|\, S_{H_q}(\boldsymbol{n\omega_{0,q}}, \boldsymbol{t}) = \left|\frac{\boldsymbol{\xi_0}}{\boldsymbol{\Omega_{c,q}}}\boldsymbol{\omega_{0,q}}\right| S_H\left(\boldsymbol{n}\frac{\boldsymbol{\xi_0}}{\boldsymbol{\Omega_{c,q}}}\boldsymbol{\omega_{0,q}}, \boldsymbol{t}\right) \stackrel{\text{def}}{=} |\boldsymbol{\omega_0}|\, S_H\left(\boldsymbol{n\omega_0}, \boldsymbol{t}\right)$$
so that (4.15c) takes the form
$$S_{H_q}(\boldsymbol{\nu}, \boldsymbol{t}) = |\boldsymbol{\omega_0}| \sum_{\boldsymbol{n}\in\mathcal{N}} S_H(\boldsymbol{n\omega_0}, \boldsymbol{t}) \mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{t}-\boldsymbol{C_{0,q}}, \boldsymbol{\nu}-\boldsymbol{n\omega_{0,q}}\rangle} \operatorname{sinc}_{\boldsymbol{L_{0,q}}}(\boldsymbol{\nu} - \boldsymbol{n\omega_{0,q}}).$$
Insertion of this in the proof of Proposition 5 then changes (4.16) into the following formula for computing all $\widehat{S_{H_q}^{\boldsymbol{\Omega_{c,q}}, \boldsymbol{\Omega}}(\cdot, \boldsymbol{t})}$ from the same coefficients $S_{\boldsymbol{\Omega_c''}}^{\boldsymbol{\Omega''}}(\boldsymbol{n\omega_0}, \boldsymbol{pT''})$:
$$\widehat{S_{H_q}^{\boldsymbol{\Omega_{c,q}}, \boldsymbol{\Omega}}(\cdot, \boldsymbol{t})}(\boldsymbol{t_0}) = |\boldsymbol{\omega_0 T''}|\, \chi_{\boldsymbol{C_{0,q}, L_{0,q}}}(\boldsymbol{t}-\boldsymbol{t_0}) \sum_{\boldsymbol{p}\in\mathcal{P}} \mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{\Omega_{c,q}}, \boldsymbol{t}-\boldsymbol{pT''}\rangle} \operatorname{sinc}_{\boldsymbol{\Omega}}(\boldsymbol{t} - \boldsymbol{pT''}) \times$$
$$\times \sum_{\boldsymbol{n}\in\mathcal{N}} S_{\boldsymbol{\Omega_c''}}^{\boldsymbol{\Omega''}}(\boldsymbol{n\omega_0}, \boldsymbol{pT''}) \mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{t}-\boldsymbol{t_0}-\boldsymbol{pT''}, \boldsymbol{n\omega_{0,q}}\rangle}. \quad (4.18)$$

Remark. The described procedure can also be used for the analysis of channel operators applied to signals not generated by a Gabor frame with narrowband window function, as well as for the analysis of Hilbert–Schmidt operators satisfying the properties in Figure 3 (b) in general. Then, the narrowband windows g and γ are chosen to investigate properties of the operator. Diagonalization properties are still useful, but now in the sense that they give a "simple" discretized descriptions of the operator.

Remark. The derivation of equation (4.2) does not require $(g_{\boldsymbol{q,r}})$ or $(\gamma_{\boldsymbol{q',r'}})$ to be *Gabor* frames. Thus one possible future variation of the results of this paper is to use, for example, compactly supported wavelet bases, such as B-spline, Daubechies or Morlet wavelets, which also give synthesis and analysis of signals that can be reconstructed from a finite number of sample values with bandlimiting conditions replaced by projections on certain shift-invariant wavelet subspaces [EG05, EG04].

Gabor bases are a natural first choice for the OFDM applications that we have described. Wavelet frame modifications of our algorithm might be more interesting for a wideband communications scenario since the "frequency-dependent modulation" in (3.2a) is actually a dilation that can only be reduced to a modulation in the narrowband scenario described in Section 3.2.

5 The algorithm and its implementation

In Section 5.1 we summarize the results of the last section in an algorithm for the coefficient operator matrix computation. Then we propose some further refinements for a fast implementation and compute its complexity in Section 5.2, followed by suggestions for how to choose windows and parameters in Section 5.3.

5.1 The algorithm

The results of Section 4 lead to the following procedure for computing the coefficient operator matrix of a Hilbert–Schmidt operator satisfying the conditions outlined in Figure 3(b).

1. Choose the spreading function coefficients $S_{\Omega_c''}^{\Omega''}(\boldsymbol{n\omega_0}, \boldsymbol{pT''})$, the Gabor windows g, γ and set all parameters to values typical for the application at hand. We give suggestions for how to do this in Section 5.3 and 6.

2. For all $\boldsymbol{q}, \boldsymbol{q'} \in \mathcal{Q}$ and $\boldsymbol{r}, \boldsymbol{r'} \in \mathcal{R}$, compute the matrix element $\langle Hg_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q'},\boldsymbol{r'}} \rangle$ in the following way:

   (a) Compute the samples, index sets and supports of $g_{\boldsymbol{q},\boldsymbol{r}}$ and $\gamma_{\boldsymbol{q'},\boldsymbol{r'}}$ as described in (4.6) and (4.7).

   (b) Compute the matrix elements $\langle Hg_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q'},\boldsymbol{r'}} \rangle$ by applying (4.8) to $Hg_{\boldsymbol{q},\boldsymbol{r}}$ and $\gamma_{\boldsymbol{q'},\boldsymbol{r'}}$. For this we need the samples $(Hg_{\boldsymbol{q},\boldsymbol{r}})(\boldsymbol{kT_\gamma})$, which we obtain by setting $f = g_{\boldsymbol{q},\boldsymbol{r}}$ in the finite sum formula (4.12), in which we get the samples

$$\left(S_H^{\Omega_c + Qb\Omega, \Omega}(\cdot, \boldsymbol{kT_\gamma} - \boldsymbol{mT_g})\right)\widehat{\ }(-\boldsymbol{mT_g})$$

   from the finite sum formula (4.16) or (4.18) for the more wideband underwater communications example below.

See Appendix B for a detailed description of a MATLAB implementation of this procedure in the univariate case.

5.2 Refinements and complexity

For an efficient implementation, we also recommend the following two refinements of the above algorithm:

1. In step 2 (b), the sum (B.9) can be obtained from a simple modulation of a small number of sample values which should be computed in advance.

   In fact, for the setup given in (4.7a) let $\mathcal{I}_{\boldsymbol{q}}$ and $\widehat{\mathcal{I}}_{\boldsymbol{r}}$ be the smallest intervals such that $Hg_{\boldsymbol{q},\boldsymbol{r}} \subseteq \mathcal{I}_{\boldsymbol{r}}$ and $\widehat{Hg_{\boldsymbol{q},\boldsymbol{r}}} \subseteq \widehat{\mathcal{I}}_{\boldsymbol{q}}$ for all $\boldsymbol{q} \in \mathcal{Q}$ and $\boldsymbol{r} \in \mathcal{R}$. We shall see below that there is only a small number of $\boldsymbol{q'} \in \mathcal{Q}$ and $\boldsymbol{r}, \boldsymbol{r'} \in \mathcal{R}$ such that the overlaps $\mathcal{I}_{\boldsymbol{r}} \cap \operatorname{supp} \gamma_{\boldsymbol{q'},\boldsymbol{r'}}$ and $\operatorname{supp} \widehat{\mathcal{I}}_{\boldsymbol{0}} \cap \operatorname{supp} \widehat{\gamma_{\boldsymbol{r'},\boldsymbol{q'}}}$ are nonempty. With (B.9) computed for the above overlaps, we only need simple modulation to also compute (B.9) for $u = Hg_{\boldsymbol{q},\boldsymbol{r}}$, $v = \gamma_{\boldsymbol{q'},\boldsymbol{r'}}$, $\boldsymbol{q}, \boldsymbol{q'} \in \mathcal{Q}$ and $\boldsymbol{r}, \boldsymbol{r'} \in \mathcal{R}$ in Proposition 3.

2. Recall from (4.16) and Figure 5 that $|\mathcal{N}| \cdot |\mathcal{P}|$ is the number of Fourier coefficients needed to describe the smooth truncation $S^{\boldsymbol{\Omega''}}_{\boldsymbol{\Omega''_c}}$ of $B_H(\boldsymbol{\nu}, \cdot)$ to the frequency interval $I_{\boldsymbol{\Omega''_c},\boldsymbol{\Omega''}}$.

   It is clear from Proposition 5 that $|\mathcal{P}|$ is proportional to the total bandwidth $I_{\boldsymbol{\Omega'_c},\boldsymbol{\Omega'}}$ occupied by the Gabor basis $(g_{\boldsymbol{q},\boldsymbol{r}})$ and thus also proportional to $|\mathcal{Q}|$. This dependence on $|\mathcal{Q}|$ comes from our choice to compute all $Hg_{\boldsymbol{q},\boldsymbol{r}}$ from the same $S^{\boldsymbol{\Omega''}}_{\boldsymbol{\Omega''_c}}$. This simplified the presentation and is also memory-efficient, since it minimizes the total bandwidth $I_{\boldsymbol{\Omega''_c},\boldsymbol{\Omega''}} \setminus I_{\boldsymbol{\Omega'_c},\boldsymbol{\Omega'}}$ added for the smooth truncation that is necessary to obtain a finite index set $\mathcal{P}$ finite. Thus it also minimizes the total number of coefficients that are needed to describe the channel behaviour in the entire frequency band $I_{\boldsymbol{\Omega'_c},\boldsymbol{\Omega'}}$. The resulting algorithm is also fast enough for the 2048×2048-matrix examples of Section 6.

   For more computational efficiency when $|\mathcal{Q}|$ is large, however, it is favourable to do a separate smooth cut-off $S^{\boldsymbol{\Omega''_q}}_{\boldsymbol{\Omega''_{c,q}}}$ of $B_H(\boldsymbol{\nu}, \cdot)$ for every $\boldsymbol{q}$ to an interval $I_{\boldsymbol{\Omega''_{c,q}},\boldsymbol{\Omega''_q}} \supseteq I_{\boldsymbol{\Omega_c}+\boldsymbol{q}b\boldsymbol{\Omega},\boldsymbol{\Omega}}$. This results in the coefficients $S^{\boldsymbol{\Omega''}}_{\boldsymbol{\Omega''_c}}(\boldsymbol{n\omega_0}, \boldsymbol{pT''})$ of (4.16) being replaced with a larger total number of coefficients, but with index sets $|\mathcal{P}_{\boldsymbol{q}}|$ proportional to $|\boldsymbol{\Omega}|$.

With these refinements, the complexity of the above procedure is that of one nested sum over the index sets $\mathcal{K}$, $\mathcal{M}$, $\mathcal{N}$ and $\mathcal{P}$ for each $\boldsymbol{q}, \boldsymbol{q'} \in \mathcal{Q}$ and $\boldsymbol{r}, \boldsymbol{r'} \in \mathcal{R}$. Altogether, this requires $\mathcal{O}(|\mathcal{K}| \cdot |\mathcal{M}| \cdot |\mathcal{N}| \cdot |\mathcal{P}| \cdot (|\mathcal{Q}| \cdot |\mathcal{R}|)^2)$ arithmetic operations. The following are typical index set sizes for the example applications of Section 6.

- $|\mathcal{R}|$ is the number of symbols for which the ISI shall be computed. For some example applications with optimally well time-frequency localized Gaussian window g, Figure 12 below shows that $|\mathcal{R}|$ of size 4–5 is enough to cover a decay of the average ISI/ICI to 10^{-6} times the average of the diagonal entries. For other windows g, we expect a need for larger $|\mathcal{R}|$.

- $|\mathcal{Q}|$ is the number of carrier frequencies, which normally equals the number of samples per received symbol. In radio communications $|\mathcal{Q}|$ is typically in the range 128–1024. For the inherently more wideband underwater example in Section 6, much smaller $|\mathcal{Q}|$ is possible. With the window and lattice matching described in Section 5.3, $|\mathcal{Q}| = 47$ in our underwater example plots.

- $|\mathcal{K}|$ and $|\mathcal{M}|$ depend on the time-frequency localization of γ and g, respectively. For the Gaussian windows used below, $|\mathcal{K}| \cdot |\mathcal{M}| = 28 \cdot 19 = 532$ for the underwater channel and $|\mathcal{K}| \cdot |\mathcal{M}| = 19 \cdot 19 = 361$ for the other two.

- $|\mathcal{N}| \cdot |\mathcal{P}|$ is a constant that depends on $|\boldsymbol{\Omega}|$ and is proportional to the area $\langle \boldsymbol{\omega}, \boldsymbol{L} \rangle$ of the spreading function support. Below $|\mathcal{N}| \cdot |\mathcal{P}|$ equals $13 \cdot 27 = 351$, $13 \cdot 58 = 754$, and $59 \cdot 237 = 13983$ in the OFDM, satellite and underwater example, respectively.

Hence, if g and γ have roughly $M = |\mathcal{M}|$ nonzero Nyquist frequency samples, and if the received symbols have $Q = |\mathcal{Q}|$ nonzero Nyquist frequency samples and if $R = |\mathcal{R}|$, then our refined algorithm requires $R^2 \cdot \mathcal{O}(M^2 \cdot Q^2)$ arithmetic operations. This can be compared to the $R^2 \cdot \mathcal{O}(Q^5)$ operations of the more naive and straightforward matrix computation approach discussed in section 4, which is clearly slower when the number of carrier frequencies Q is larger than M.

Example 1. We show in Figure 7 that the Gaussian window approximation used in Section 6 has $M = 19$ Nyquist frequency samples. We can compare this with the B-spline windows $B_0 \stackrel{\text{def}}{=} I_{0,1}$ and $B_n \stackrel{\text{def}}{=} B_{n-1} * I_{0,1}$, which also were proposed in [MSG+05]. Since $\widehat{B_n}(\xi) = \text{sinc}_1(\xi)^{n+1}$, its ϵ-essential support is contained in the interval $I_{0,\Omega}$ for which $1/(\pi\Omega/2)^{n+1} = \epsilon$, that is, $\Omega = 2\epsilon^{-1/(n+1)}/\pi$. Since the length of the support of B_n is $n+1$, M is now the smallest integer larger than $2\epsilon^{-1/(n+1)}(n+1)/\pi + 1$, which we plot for $\epsilon = 10^{-6}$ and some n in Figure 6. The smallest M is 25, which we get for B_{12}, B_{13} and B_{14}.

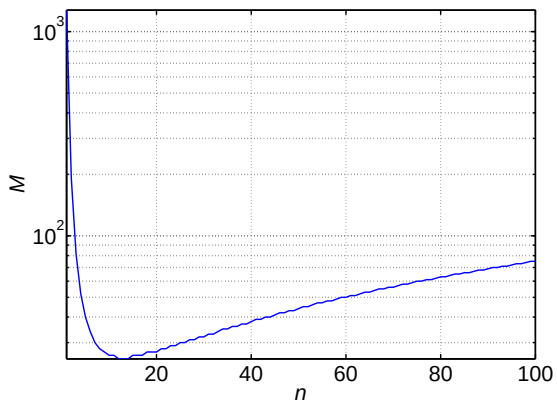


Figure 6: Number of nonzero Nyquist frequency samples for some B-splines B_n.

5.3 Parameters and window functions

In the operator discretizations (4.1), the shape of the Gabor window g can be optimized in different ways depending on the application. For wireless multicarrier systems, the pulse shaping of g is an active research topic in its own, which we do not pursue further here (see, for example, [MSG+05] for more details and references). We have already motivated in Section 4 that both bases should be Gabor bases with the same lattice parameters. For reasons described below, we choose to set the lattice "time" parameter to $\boldsymbol{aT_g}$ with $\boldsymbol{a} \in \mathbb{Z}_+$. Similarly, for some $\boldsymbol{b} \in \mathbb{R}_+^d$ we set the lattice frequency parameter to $\boldsymbol{b\Omega}$, so that the unit cell "area" $\boldsymbol{aT_g b\Omega} = \boldsymbol{ab}$ only depends on $\boldsymbol{a}$ and $\boldsymbol{b}$. In the examples of Section 6, we choose parameters so that $ab > 1$, which gives undersampling (see [Grö00]) and a Gabor system that is a Riesz basis for its span. In general, for finite index sets $\mathcal{Q} \times \mathcal{R}$, we will consider Gabor Riesz bases

$$g_{\boldsymbol{q},\boldsymbol{r}} = T_{\boldsymbol{raT_g}} M_{\boldsymbol{qb\Omega}} g \quad \text{and} \quad \gamma_{\boldsymbol{q},\boldsymbol{r}} = T_{\boldsymbol{raT_g}} M_{\boldsymbol{qb\Omega}} g, \qquad (\boldsymbol{q},\boldsymbol{r}) \in \mathcal{Q} \times \mathcal{R}. \tag{5.1a}$$

We are primarily interested in windows with very good joint time-frequency localization, which is of utmost importance for low ISI and ICI. Good frequency localization also allows for high transmission power and, therefore, large signal-to-noise ratio in the transmission band without exceeding power leakage bounds for other frequency bands. Such leakage is strictly regulated for radio communications, but not (yet) for underwater sonar communications, where, however, frequency bands already in use by, for example, the human ear or dolphins should be respected.

In the following, we shall use standard Gaussian windows

$$g(\boldsymbol{x}) = \mathrm{e}^{-\langle \boldsymbol{\alpha x}, \boldsymbol{x} \rangle} \qquad \text{with Fourier transform} \qquad \widehat{g}(\boldsymbol{\nu}) = \sqrt{\frac{\pi^d}{|\boldsymbol{\alpha}|}} \mathrm{e}^{-\pi^2 \left\langle \frac{\boldsymbol{\nu}}{\boldsymbol{\alpha}}, \boldsymbol{\nu} \right\rangle}. \tag{5.1b}$$

Gaussian windows have optimal time-frequency localization [Grö00, Section 2.2], which results in very low ISI/ICI. This also has the advantage of making the effects of truncation to ϵ-essential support clearly visible in the plots of Section 6.

The window γ is set to $\gamma = Hg$ in the example applications of Section 6. Lower ISI/ICI can be obtained with other choices described in [MSG+05].

Other application-dependent parameters we choose in the following way:

1. *Index sets:* Choose the Gabor basis index set $\mathcal{R}$, the number of carrier frequencies $|\mathcal{Q}|$ and the desired minimum size of the index sets $\mathcal{N}$ and $\mathcal{P}$. Larger $|\mathcal{N}|$ and $|\mathcal{P}|$ can be used for obtaining better control of the smoothness and decay of $S_{\boldsymbol{\Omega}_{\mathrm{c}}''}^{\boldsymbol{\Omega}''}$.

2. *Channel-dependent parameters:* For the channel at hand, choose the total allowed frequency band $[\boldsymbol{\Omega_0}, \boldsymbol{\Omega_1}]$, a "rectangle" $I_{\boldsymbol{\omega}_{\mathrm{c}},\boldsymbol{\omega}} \times I_{\boldsymbol{C},\boldsymbol{L}}$ containing the ϵ-essential support of S_H and the type of time-decay of S_H. (See Section 6 for examples.)

3. Choose the lattice constants $\boldsymbol{a}$ and $\boldsymbol{b}$, as well as the parameter $\boldsymbol{\alpha}$ in (5.1b), which uniquely determines the ϵ-essential support $\boldsymbol{\Omega} = \frac{1}{\boldsymbol{T_g}}$ of $\widehat{g}$. For non-Gaussian g, this step corresponds to the choice of a dilation parameter $\boldsymbol{\alpha}$ in an equation of the kind $g(\boldsymbol{x}) \stackrel{\text{def}}{=} g_0(\boldsymbol{\alpha x})$.

   We have done the following choice, inspired by a similar design criterion in [KM98, Koz97, Mat00] For the smallest "rectangle" $I_{\boldsymbol{A}_c,\boldsymbol{A}} \times I_{\boldsymbol{B}_c,\boldsymbol{B}}$ that contains the ϵ-essential support of the short-time Fourier transform of g with window g (defined in (2.10)), choose $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{\alpha}$ such that

$$\frac{\boldsymbol{\omega}}{\boldsymbol{L}} = \frac{\boldsymbol{B}}{\boldsymbol{A}} = \frac{\boldsymbol{b\Omega}}{\boldsymbol{aT_g}} \tag{5.2}$$

   and such that $(g_{\boldsymbol{q},\boldsymbol{r}})$ is a Riesz basis for its span (obtained by setting $ab > 1$ in the results presented in Section 6). See Section B.13 for details.

4. Choose $\mathcal{Q}$ so that the supports of all $\widehat{g_{\boldsymbol{q},\boldsymbol{r}}}$ are in the allowed frequency band $[\boldsymbol{\Omega_0}, \boldsymbol{\Omega_1}]$. Set $\boldsymbol{\Omega'_c}$ and $\boldsymbol{\Omega'}$ according to (4.15a). Set $\boldsymbol{\Omega''_c} = \boldsymbol{\Omega'_c}$ and choose $\boldsymbol{\Omega''}$ to be large enough to obtain the desired minimum size of $\mathcal{P}$ and also large enough to for the desired smooth truncation (4.15d) and (4.15e) ($\Omega'' \geq 1.5\Omega'$ in our implementation). Set $\boldsymbol{T_\gamma} = \frac{1}{\boldsymbol{\Omega}+\boldsymbol{\omega}}$ and $\boldsymbol{T''} = \frac{1}{\boldsymbol{\Omega''}}$.

5. Choose $\boldsymbol{L_0}$ and $\boldsymbol{C_0}$ so that (4.15b) holds and such that $\boldsymbol{L_0}$ is large enough to obtain the desired minimum size $\mathcal{N} = \frac{\boldsymbol{\omega}}{\boldsymbol{\omega_0}} = \boldsymbol{\omega L_0}$. Also set $\boldsymbol{\omega_0} = \frac{1}{\boldsymbol{L_0}}$.

6. Compute the Nyquist frequency samples of the restriction of g to its ϵ-essential support (as illustrated in Figure 7 (a)).

7. Compute the samples $S^{\boldsymbol{\Omega''}}_{\boldsymbol{\Omega''_c}}(\boldsymbol{n\omega_0}, \boldsymbol{pT''})$ that appear as coefficients in (4.16). There are basically two ways to obtain these coefficients:

   (a) Compute them from measurements or from a detailed model of a real channel.

   (b) Generate a random choice of the samples that satisfies the decay and support properties described in Section 3 and Section 6. It seems reasonable to assume that then there is always some constellation of antennas and reflecting objects that correspond to this particular shape of the spreading function.

   For the results in Section 6 we have chosen approach (b).

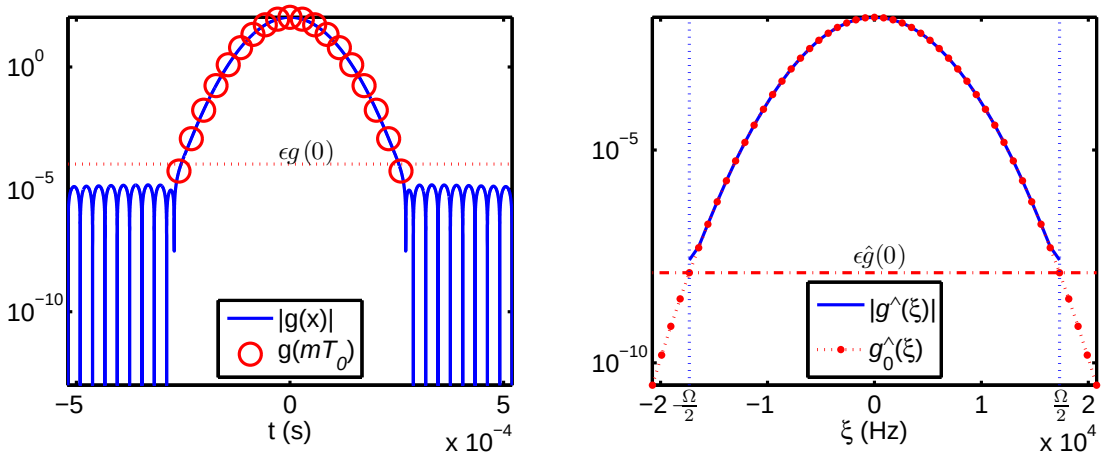
6 Applications

We shall now describe parameter values and show sample plots for three channels with different spreading function support areas.

6.1 Mobile phone communications

For a mobile phone moving at 100 km/h, equations (3.2) imply that the maximum Doppler spread is $\frac{V_P}{V_w}\xi = 100/(299792.458 \cdot 3600)\xi \approx 10^{-7}\xi$. For the GSM mobile phone frequency bands ξ is ranging from 450.4–1990 MHz, so that frequency shifts caused by the Doppler effect is in the interval $[-184, 184]$ Hz. Typical time delays are of the order 10^{-6} s, so the spreading function support area is of size $4 \cdot 10^{-4}$. This support would exceed the critical value 1 (see page 11) if the highest channel frequencies in use exceed 5 THz. These are infrared light frequencies, for which the water in the Earth's atmosphere absorbs too strongly for us to expect these frequencies to be useful for wireless communications. Thus, with the terminology of Section 2.4, mobile phone channels are inherently underspread.

With other parameters chosen as described in Section 5.3, we obtain a Gaussian window g_0 with Fourier transform ϵ-essential support Ω. Hence we can approximate g_0 by reconstructing it from the Nyquist sampling theorem with sample interval $T_g = 1/\Omega$ and samples outside the ϵ-essential support of g_0 discarded. This construction gives a very small upper bound for both the supremum and the L^2-norm of the resulting truncation error when only a small number of nonzero samples are kept, as can be seen in the plots of g_0, g, $\widehat{g_0}$ and $\widehat{g}$ in Figure 7. In practice, ϵ should be chosen small enough for the resulting errors to be dominated by the overall noise level of the application at hand. For example, note in Figure 7 (a) how the truncation to a truly bandlimited g with a finite number of



(a) (b)

Figure 7: We use the Gabor window/pulse shape g that we obtain from a Gaussian $g_0(x) = \mathrm{e}^{-\alpha x^2}$ by assuming $\widehat{g_0}$ to have bandwidth Ω given by its ϵ-essential support and truncating the reconstruction of g_0 from its Nyquist frequency samples to samples in the ϵ-essential support of g_0. We do this for $\epsilon = 10^{-6}$. Plots (a) and (b) show a comparison of the resulting g and $\widehat{g}$ with g_0 and $\widehat{g_0}$, respectively.

nonzero samples gives "Gaussian decay" down to amplitudes below ϵ and then a slowly decaying tail with negligible amplitude and L^2-norm. In this and the following examples, $\epsilon = 10^{-6}$.

We show some example plots in Figure 8 for a system with 128 carrier frequencies and 16 OFDM symbols. In (a) we show the ϵ-essential support of the short-time Fourier transform $\mathcal{V}_{g_{\boldsymbol{q},r}} g_{\boldsymbol{q},\boldsymbol{r}}$ (defined in (2.10)) for some neighbouring basis functions $g_{\boldsymbol{q},\boldsymbol{r}}$, from which we se that we can expect nonzero ISI and ICI at least for basis functions at distance

$$\|(\boldsymbol{q},\boldsymbol{r}) - (\boldsymbol{q}',\boldsymbol{r}')\|_{l^2} \stackrel{\text{def}}{=} \sqrt{(\boldsymbol{q}-\boldsymbol{q}')^2 + (\boldsymbol{r}-\boldsymbol{r}')^2} \leq 4$$

in the Gabor lattice. In (b) we plot the "bandpass filtered spreading function"

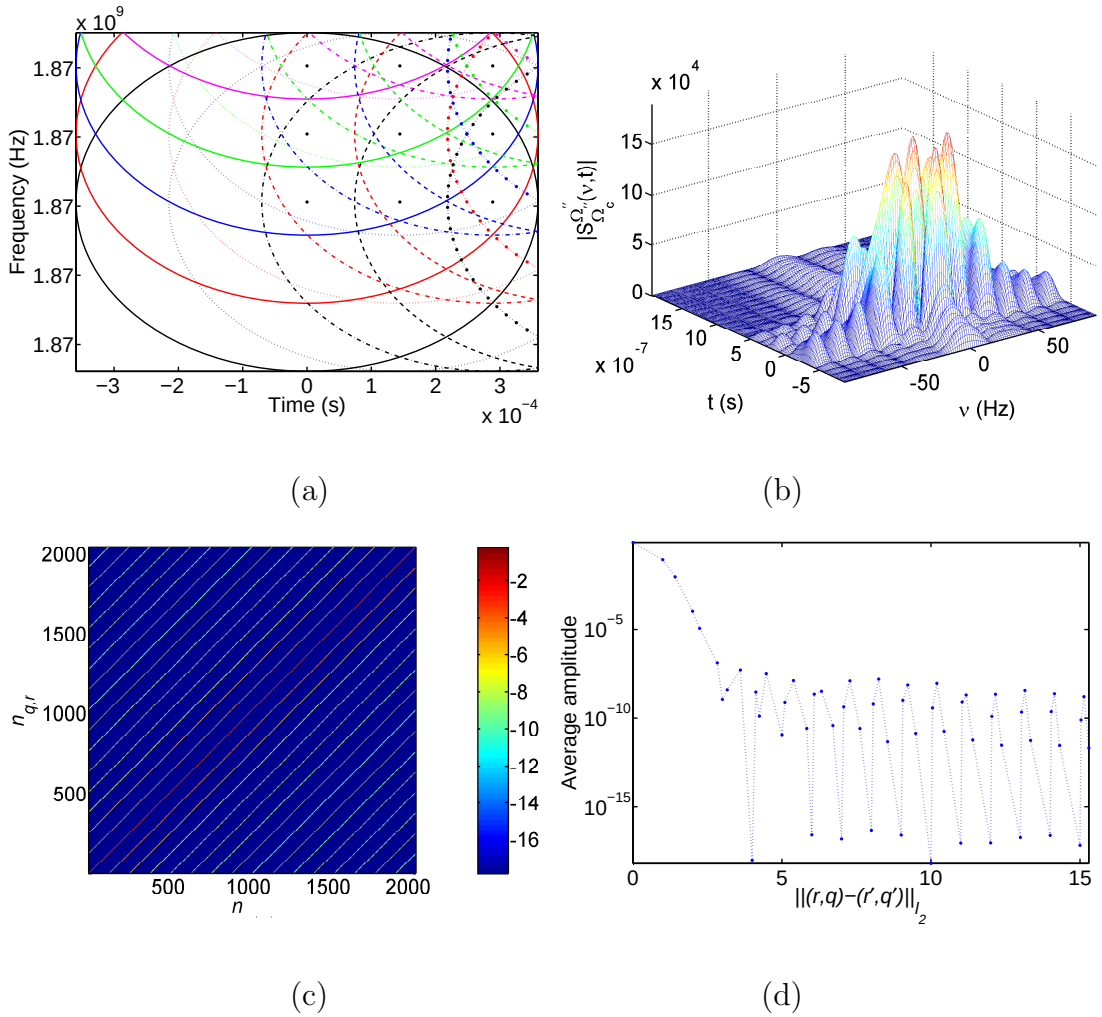


Figure 8: OFDM channel example. (a) ϵ-essential support of the short-time Fourier transform $\mathcal{V}_{g_{\boldsymbol{q},r}} g_{\boldsymbol{q},\boldsymbol{r}}$ for some neighbouring basis functions $g_{\boldsymbol{q},\boldsymbol{r}}$. (b) The spreading function. (c) The 10-logarithm of $|\langle H g_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q}',\boldsymbol{r}'} \rangle|$ with index ordering $n_{q,r}$ defined in (6.1). (d) Off-diagonal decay.

$S_{\Omega_c''}^{\Omega''}(\boldsymbol{\nu}, \cdot)$ computed from its samples by using (4.15c) and (4.15f) in a way similar to the proof of Proposition 5 (see Appendix B.1 for details). In this plot we have assumed an environment with a very large amount of small scatterers adding up to a white Gaussian noise distribution of the coefficient values (see Section 6.4 for examples with a "smother" spreading function with more correlated coefficients).

For plotting the coefficient operator matrix, we need to define a linear ordering of the index sets

$$\mathcal{Q}\times\mathcal{R} = \{q_0, q_0+1, q_0+2, \ldots, q_0+(|\mathcal{Q}|-1)\}\times\{r_0, r_0+1, r_0+2, \ldots, r_0+(|\mathcal{R}|-1)\}\,.$$

We have chosen to group together indices belonging to the same OFDM symbol by using the order

$$n_{q,r} \stackrel{\text{def}}{=} (r-r_0)\cdot|\mathcal{Q}| + q - q_0 + 1, \tag{6.1}$$

for which we have plotted the 10-logarithm of the matrix element amplitudes in Figure 8 (c). With this ordering, the matrix is divided into 16×16 submatrices, such that in each submatrix, r and r' are fixed. The submatrices for which $r = r'$ show the ICI of symbol number r. Submatrices for which $r \neq r'$ show the ISI between OFDM symbols number r and r'.

Due to the matching of the Gabor lattice and the shape of the frequency localization of g to the shape of the spreading function support, which we explained in Section 5.3, the size of $|\langle H g_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q}',\boldsymbol{r}'}\rangle|$ should mainly depend on the distance $\|(\boldsymbol{q},\boldsymbol{r}) - (\boldsymbol{q}',\boldsymbol{r}')\|_{l^2}$ between the time-frequency support centerpoints in the Gabor lattice. For making this off-diagonal decay more visible, we have grouped together matrix elements for which this distance is the same and plotted the average amplitude in Figure 8 (d). Note the clearly visible effect that occurs at distances more than four, which is caused by the truncations of windows and spreading functions to their ϵ-essential support.

6.2 Satellite communications

The speed of a communications satellite in geostationary orbit is about 3 km per second. Thus the maximum Doppler shift is $\frac{V_{\mathrm{P}}}{V_{\mathrm{w}}}\xi = (3/299792.458)\xi \approx 10^{-5}\xi$. With typical transmission frequencies 1-30 GHz [MB02], we can expect Doppler shifts up to some 10^4 Hz. Here, we will use an example from [MB02, p. 47] with transmission frequency 6 GHz and Doppler shift 18 kHz. Again, we assume the maximum time-spread to be some 10^{-6} s. Then the spreading function support area is less than 0.036, so this is an underspread channel as well.

We show the same plots as for the OFDM example in Figure 9. Note that as a result of the window and lattice constant matching, the plots (a) in Figure 8 an 9 are largely the same up to scaling.

6.3 Underwater sonar communications

For a vehicle travelling at 30 knot in sea water and using sonar communications, we have $\frac{V_{\mathrm{P}}}{V_{\mathrm{w}}}\xi \approx \frac{30 \cdot 0.51444}{1531}\xi \approx 10^{-2}\xi$. We will use parameters typical for some medium range systems described in [Sto99] with maximum time spread around 0.01 s and a typical frequency band 20-35 kHz, so that the maximum Doppler shift is about 350 Hz. (More examples can be found, for example, in [LO97, Mid87, Sto96, ZK00, ZT02].) These settings give spreading function support area 7 and an overspread channel, which is typical for underwater sonar communications channels in general.

We show the same plots as for the previous two examples in Figure 10.

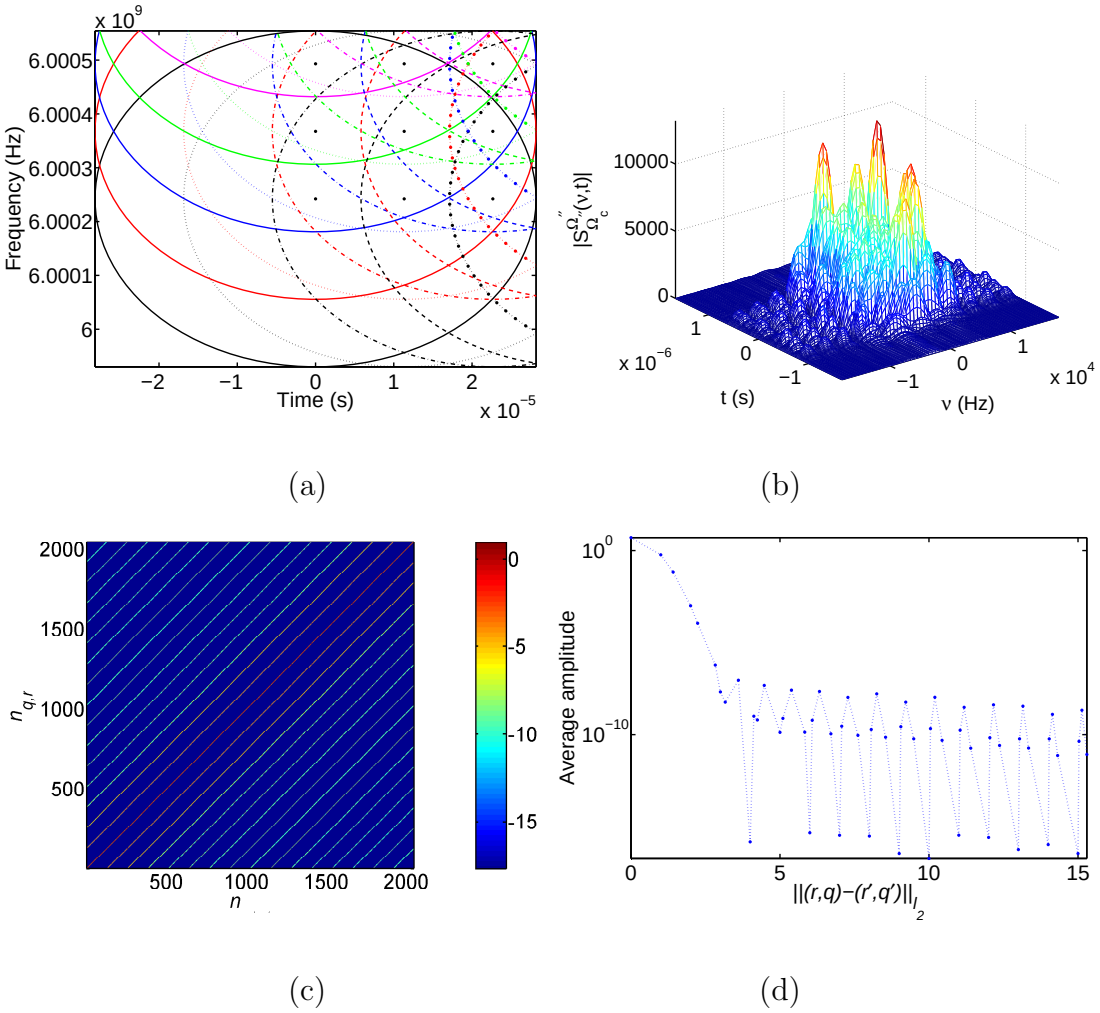


Figure 9: Satellite channel example. (a) ϵ-essential support of the short-time Fourier transform $\mathcal{V}_{g_{\boldsymbol{q},\boldsymbol{r}}} g_{\boldsymbol{q},\boldsymbol{r}}$ for some neighbouring basis functions $g_{\boldsymbol{q},\boldsymbol{r}}$. (b) The spreading function. (c) The 10-logarithm of $|\langle H g_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q}',\boldsymbol{r}'} \rangle|$ with index ordering $n_{q,r}$ defined in (6.1). (d) Off-diagonal decay.

6.4 Further spreading function examples

In the above examples we used independent Gaussian distributions to obtain the spreading function coefficients. For a "nicer" environment, one can expect more correlation between the samples. We show examples of such spreading functions in Figure 11. In Figure 12 we compare these two different spreading functions with all other channel parameters being identical. The plots show that these two different correlations of the spreading function samples do not affect the speed of the off-diagonal decay significantly.

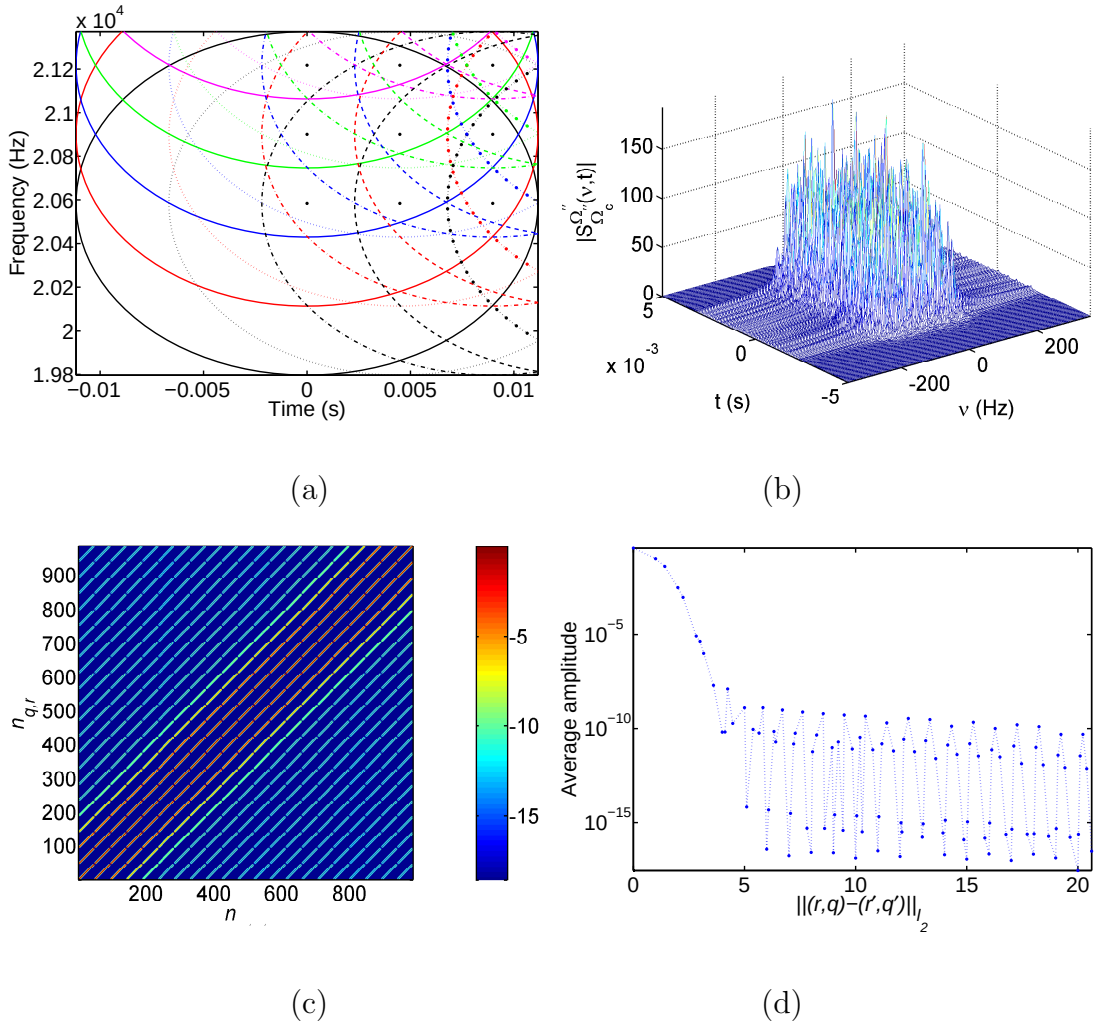
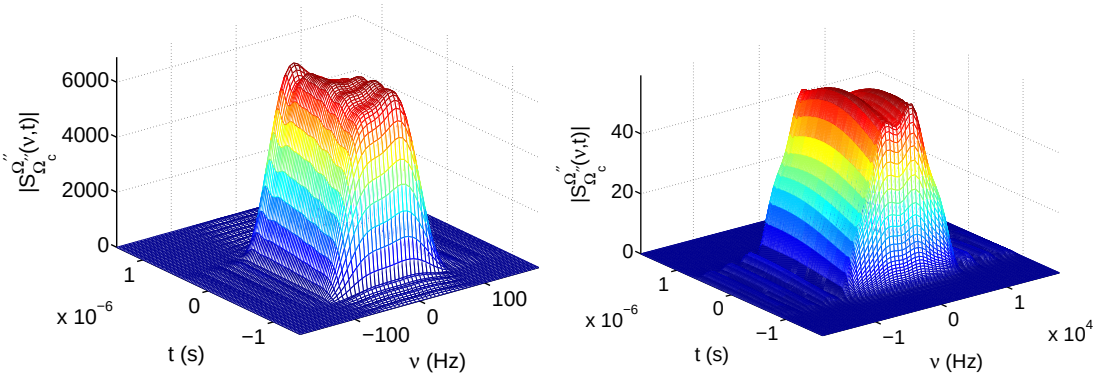
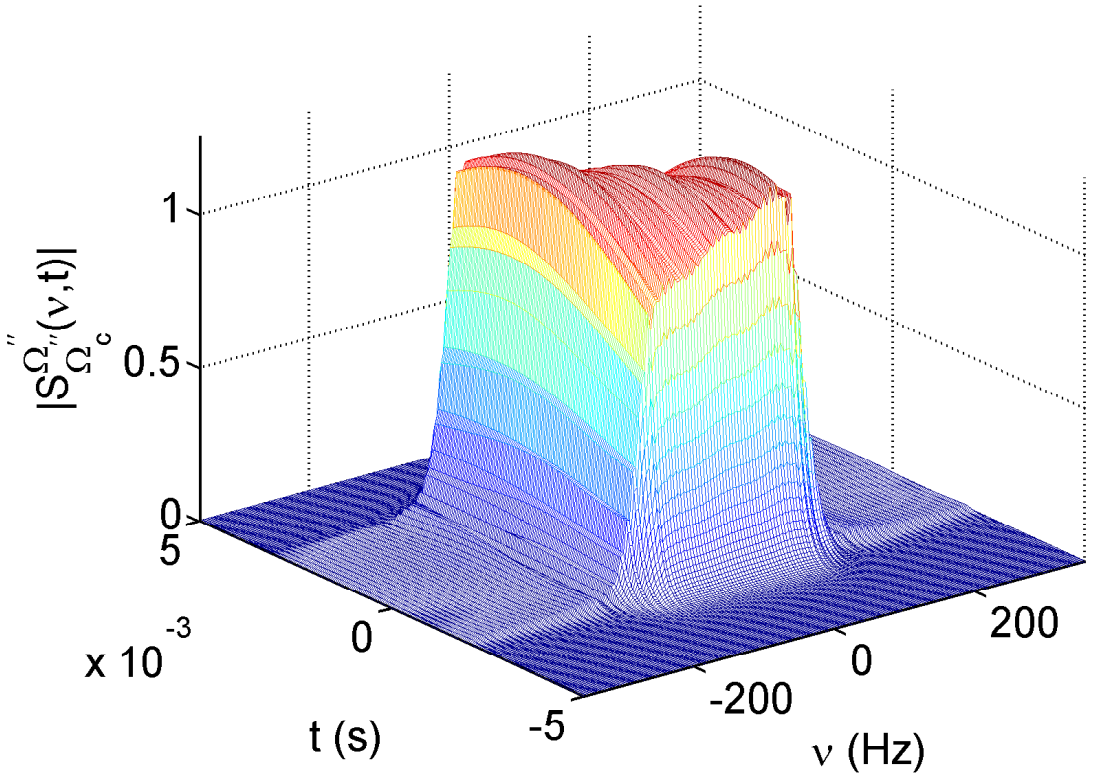


Figure 10: Underwater channel example. (a) ϵ-essential support of the short-time Fourier transform $\mathcal{V}_{g_{\boldsymbol{q},\boldsymbol{r}}} g_{\boldsymbol{q},\boldsymbol{r}}$ for some neighbouring basis functions $g_{\boldsymbol{q},\boldsymbol{r}}$. (b) The spreading function. (c) The 10-logarithm of $|\langle H g_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q}',\boldsymbol{r}'} \rangle|$ with index ordering $n_{q,r}$ defined in (6.1). (d) Off-diagonal decay.



(a) (b)



(c)

Figure 11: Spreading functions with more correlated coefficients $S^{\Omega''}_{\Omega''_c}(\boldsymbol{n\omega_0}, \boldsymbol{pT''})$. (a)–(c) shows the spreading functions for the OFDM, satellite and underwater example, respectively. Figure 12 shows a comparison of the resulting off-diagonal decays with those in figures 8–10.

7 Conclusions

Using a refinement of the standard multipath propagation model for the short time behaviour of narrowband wireless channels, we have derived a spreading function integral representation of such channels with a $C^{(\infty)}$ spreading function with subexponential decay.

This, together with a channel discretization using well time-frequency localized Gabor bases, allowed us to derive formulas and an algorithms for the efficient computation of certain matrix representations of communication channels. The elements of this matrix describe the intersymbol and intercarrier interference for the transmitted signal. We derived the algorithm, as well as some refinements of it, under a minimum of assumptions or simplifications beyond the channel and signal properties that are known from our channel and signal model.

Next, we discussed parameter choices in general and for three different channels. For these channels we used a MATLAB implementation of our algorithm to compute example plots showing the time-frequency localization of the Gabor

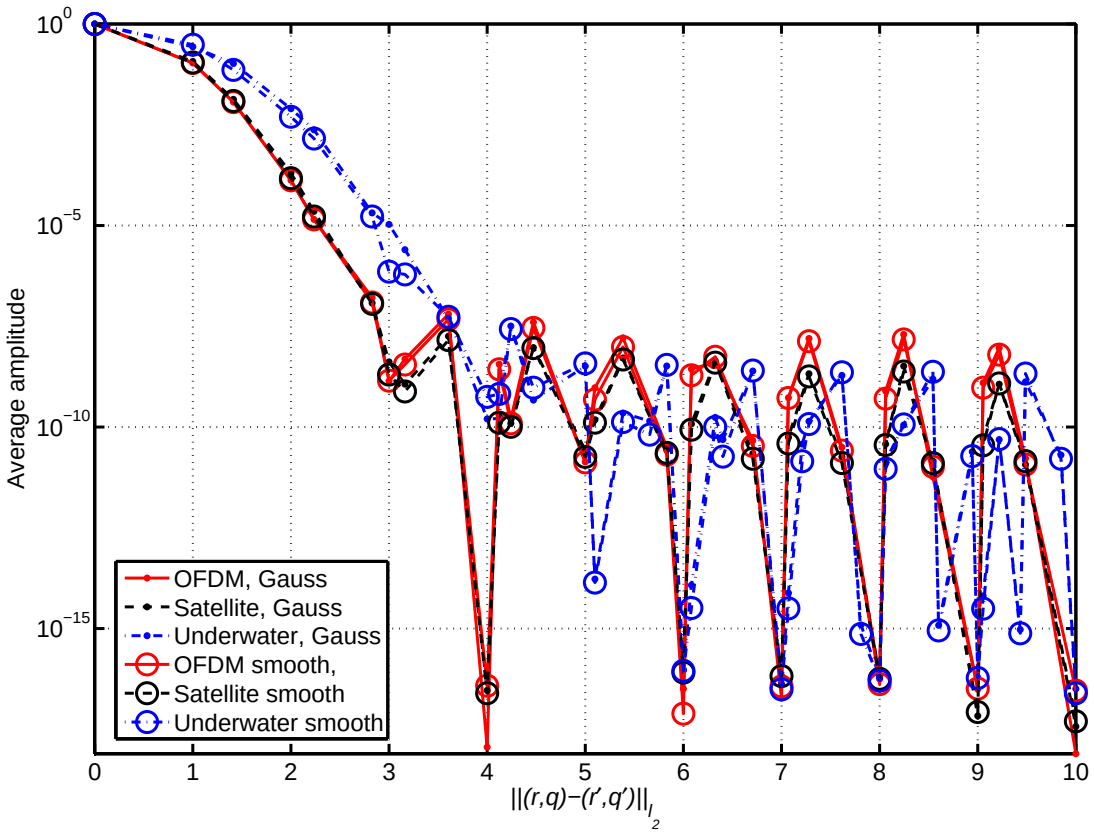


Figure 12: Off-diagonal decays with spreading function coefficients normalized so that the average of $\|Hg_{\boldsymbol{q},\boldsymbol{r}}\|_2$ equals one.

basis functions, the spreading functions, the coefficient operator matrix, and its off-diagonal decay.

Our implementation is fast enough for at least 2048×2048 matrices and considerably faster than a simpler and more straightforward approach to computing the matrix elements.

Due to bandwidth and delay restrictions, multicarrier communications must use bandlimited basis functions defined by a finite number of nonzero Nyquist frequency samples. Our plots show clearly how such restrictions affect the off-diagonal decay of the coefficient operator matrix. Thus the algorithm and software can be useful for the numerical comparisons of the off-diagonal decay for different pulse shapes and parameter settings.

Moreover, although we primarily consider communications applications in this report, we derived our algorithm in a more general multivariate setting, as an analysis tool for certain classes of Hilbert Schmidt operators with potential other theoretical and practical applications.

Acknowledgements: The authors would like to thank Dr. Werner Kozek at Siemens AG in Vienna, Austria, and Professor Harald Haas at Jacobs University Bremen, Germany, for insightful discussions about the physical properties of wireless channels. We are also thankful to Professor Karlheinz Gröchenig at the Numerical Harmonic Analysis Group in Vienna, Austria, for providing the references on which Section 2.2 is based.

Appendices

A Fourier transform decay vs differentiability

We claimed at page 7 that

> "...the function $g(x) = (1+\cos(\pi x))^4 \chi_{[-1,1]}(x)$ has Fourier transform decay $\widehat{g}(\xi) = \mathcal{O}((1+|\xi|)^{-\alpha})$ for $\alpha = 9$ but not for $\alpha > 9$".

This is not entirely obvious, but follows from Corollary 2 below.

Theorem 3 (Decay vs. Fourier transform smoothness)**.**

(a) For compactly supported $f \in C^{(n)}(\mathbb{R})$, it holds that $\widehat{f}(\xi) = o(\xi^{-n})$.

(b) If f is the inverse Fourier transform of some $\widehat{f} \in L^1(\mathbb{R})$ and $\xi^n \widehat{f}(\xi) \in L^1(\mathbb{R})$, then f is n times continuously differentiable and

$$\widehat{f^{(n)}}(\xi) = (\mathrm{i}2\pi\xi)^n \widehat{f}(\xi).$$

Proof. (a) For compactly supported $f \in C^{(n)}(\mathbb{R})$ it follows that $f^{(k)} \in L^1(\mathbb{R})$ for $k = 0, 1, \ldots, n$, so that $\widehat{f^{(n)}}(\xi)$ is continuous, $\widehat{f^{(n)}}(\xi) \to 0$ as $|\xi| \to \infty$ and integration by parts gives that

$$\widehat{f^{(n)}}(\xi) = \int_{-\infty}^{\infty} f^{(n)}(x) \mathrm{e}^{-\mathrm{i}2\pi\xi x}\,\mathrm{d}x = (\mathrm{i}2\pi\xi)^n \int_{-\infty}^{\infty} f(x) \mathrm{e}^{-\mathrm{i}2\pi\xi x}\,\mathrm{d}x = (\mathrm{i}2\pi\xi)^n \widehat{f}(\xi).$$

Thus $\widehat{f}(\xi) = o(\xi^{-n})$.

(b) is proved for $n = 1$ in [Kat04, Theorem 1.6, p. 136]. Hence the result follows also for larger n, since if $\widehat{f} \in L^1(\mathbb{R})$ and $\xi^n \widehat{f}(\xi) \in L^1(\mathbb{R})$, then $\xi^k \widehat{f}(\xi) \in L^1(\mathbb{R})$ for $k = 1, 2, \ldots, n-1$. □

Corollary 2. *The function $g(x) \stackrel{\text{def}}{=} (1+\cos(\pi x))^n \chi_{[-1,1]}(x)$ has Fourier transform decay $\widehat{g}(\xi) = \mathcal{O}((1+|\xi|)^{-\alpha})$ for $\alpha = 2n+1$ but not for $\alpha > 2n+1$.*

Proof. Let γ be the continuous function

$$\gamma(x) \stackrel{\text{def}}{=} (1+\cos(\pi x))\chi_{[-1,1]}(x).$$

Then

$$\begin{aligned} \gamma'(x) &= -\pi \sin(\pi x)\chi_{[-1,1]}(x) \qquad \text{(continuous)} \qquad \text{and} \\ \gamma''(x) &= -\pi^2 \cos(\pi x)\chi_{[-1,1]}(x) \qquad \text{(discontinuity at } \pm 1,\ \gamma''(\pm 1) = \pi^2). \end{aligned}$$

Set

$$K \stackrel{\text{def}}{=} \left\{ (k_1, k_2, \ldots, k_n) \in \mathbb{N}. : \sum_m k_m = 2n \right\}.$$

Now if we do an iterative application of the product rule for differentiation, then the first discontinuous term that appears is the term $c_{(2,2,\ldots,2)}\gamma''(x)^n$ in the $2n$th derivative

$$\begin{aligned} g^{(2n)}(x) =& \sum_{\boldsymbol{k}=(k_1,k_2,\ldots,k_n)\in K} c_{\boldsymbol{k}}\gamma^{(k_1)}(x)\gamma^{(k_2)}(x)\cdots\gamma^{(k_n)}(x) \\ =& c_{(2,2,\ldots,2)}\gamma''(x)^n + \sum_{\boldsymbol{k}\in K\setminus\{(2,2,\ldots,2)\}} c_{\boldsymbol{k}}\gamma^{(k_1)}(cx)\gamma^{(k_2)}(x)\cdots\gamma^{(k_n)}(x) \end{aligned}$$

with

$$c_{\boldsymbol{k}} \stackrel{\text{def}}{=} \binom{k_1}{2n}\binom{k_2}{2n-k_1}\cdots\binom{k_n}{2n-k_1-k_2-\cdots-k_{n-1}}.$$

Hence g is $2n-1$ times continuously differentiable and $2n$ times differentiable. Similarly to the proof of Theorem 3 (a), integration by parts gives that

$$\begin{aligned} \widehat{g^{(2n+1)}}(\xi) =& \int_{\mathbb{R}} g^{(2n+1)}(x)\mathrm{e}^{-\mathrm{i}2\pi\xi x}\,\mathrm{d}x = \lim_{\varepsilon\searrow 0}\left[g^{(2n)}(x)\mathrm{e}^{-\mathrm{i}2\pi\xi x}\right]_{x=-1+\varepsilon}^{1-\varepsilon} + \int_{-1}^{1} g^{(2n)}(x)\mathrm{e}^{-\mathrm{i}2\pi\xi x}\,\mathrm{d}x \\ =& -2\mathrm{i}c_{(2,2,\ldots,2)}\pi^{2n}\frac{\mathrm{e}^{\mathrm{i}2\pi\xi}-\mathrm{e}^{-\mathrm{i}2\pi\xi}}{2\mathrm{i}} + \mathrm{i}2\pi\xi\int_{-1}^{1} g^{(2n)}(x)\mathrm{e}^{-\mathrm{i}2\pi\xi x}\,\mathrm{d}x \\ =& -2\mathrm{i}c_{(2,2,\ldots,2)}\pi^{2n}\sin(2\pi\xi) + (\mathrm{i}2\pi\xi)^{2n+1}\int_{-1}^{1} g(x)\mathrm{e}^{-\mathrm{i}2\pi\xi x}\,\mathrm{d}x \\ =& -2\mathrm{i}c_{(2,2,\ldots,2)}\pi^{2n}\sin(2\pi\xi) + (\mathrm{i}2\pi\xi)^{2n+1}\widehat{g}(x). \end{aligned}$$

From this and the fact that $g, g^{(2n+1)} \in L^1$, it follows that $\widehat{g}, \widehat{g^{(2n+1)}} \in C_0$ and

$$\widehat{g}(\xi) = \mathcal{O}(\xi^{-(2n+1)})$$

For the converse statement, note that we cannot have $\widehat{g}(\xi) = \mathcal{O}(\xi^{-(2n+1+\varepsilon)})$ for any $\varepsilon > 0$, because then $\widehat{g}(\xi), \xi^{2n}\widehat{g}(\xi) \in L^1$ and Theorem 3 would imply that g is $2n$ times continuously differentiable. □

B Matlab implementation: overview and function reference

This appendix includes and explains the MATLAB source code used for computing the channel matrices and producing all the plots in this report.

Figure 13 shows the structure of our MATLAB code for computing the channel matrix. The main function `ChannelSetupAndAnalysis` (see page 50) begins with setting a few basic parameters, such as the type of channel, matrix sizes and the level of detail in some of the plots. Then the function `SetParameters` (page 87) is called for computing the system dependent parameters and window functions that were described in sections 5.3 and 6. Finally, the function `CoeffOpMat` is called for computing and plotting the matrix elements. If the variable `VERBOSE` is set to `true`, then `ChannelSetupAndAnalysis` will also call some other functions, described in further subsections, that produce some of the other plots in this report.

For computing the matrix elements, the function `CoeffOpMat` (see page 53) first computes the Gabor basis elements $\gamma_{\boldsymbol{q}',\boldsymbol{r}'}$ and $g_{\boldsymbol{q},\boldsymbol{r}}$ by calling the functions `TFshiftAndResample` (page 107), `resample` (page 85) and `TFshift` (page 105) for the necessary time-frequency shifts . The application of H to each $g_{\boldsymbol{q},\boldsymbol{r}}$ is computed by the function `H` (page 76), which in turn calls the function `FTBPFS` for computing $\widehat{S_H^{\boldsymbol{\Omega_c},\boldsymbol{\Omega}}(\cdot,\boldsymbol{t})}$ (page 69). Finally the matrix element inner product $\langle H g_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q}',\boldsymbol{r}'} \rangle$ is computed by calling the function `l2Innerprod` (page 78).

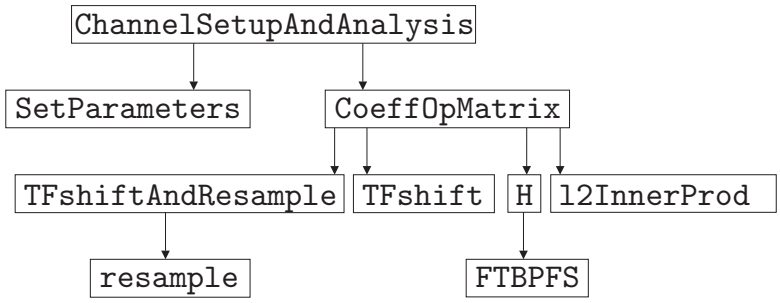


Figure 13: Program structure of the MATLAB code for computing the channel matrix.

B.1 BPFS: Compute $S_{\Omega_c''}^{\Omega''}(\boldsymbol{\nu}, \boldsymbol{t})$

File path: Matlab/ChannelMatrix/BPFS.m
Short description: Compute $S_{\Omega_c''}^{\Omega''}(\boldsymbol{\nu}, \boldsymbol{t})$.
Additional explanation: This function computes the samples of $S_{\Omega_c''}^{\Omega''}(\boldsymbol{\nu}, \boldsymbol{t})$ that we plotted, for example, in Figure 8 (b). This is done as follows:

In the special case when $S_H = S_{\Omega_c''}^{\Omega''}$, (4.15c) gives that

$$S_{\Omega_c''}^{\Omega''}(\boldsymbol{\nu}, \boldsymbol{t}) = |\boldsymbol{\omega_0}| \sum_{\boldsymbol{n}\in\mathcal{N}} S_{\Omega_c''}^{\Omega''}(\boldsymbol{n\omega_0}, \boldsymbol{t}) \mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{t}-\boldsymbol{C_0}, \boldsymbol{\nu}-\boldsymbol{n\omega_0}\rangle} \mathrm{sinc}_{\boldsymbol{L_0}}(\boldsymbol{\nu}-\boldsymbol{n\omega_0}), \qquad |\mathcal{N}| < \infty. \tag{B.1}$$

Moreover, by rewriting (4.15f) in the Nyquist sampling theorem form (4.5b) we get

$$S_{\Omega_c''}^{\Omega''}(\boldsymbol{n\omega_0}, \boldsymbol{t}) = |\boldsymbol{T''}| \sum_{\boldsymbol{p}\in\mathcal{P}} S_{\Omega_c''}^{\Omega''}(\boldsymbol{n\omega_0}, \boldsymbol{pT''}) \mathrm{e}^{\mathrm{i}2\pi\langle \Omega_c'', \boldsymbol{t}-\boldsymbol{pT''}\rangle} \mathrm{sinc}_{\boldsymbol{\Omega''}}(\boldsymbol{t}-\boldsymbol{pT''})$$

Insertion in (B.1) gives

$$\begin{aligned} S_{\Omega_c''}^{\Omega''}(\boldsymbol{\nu}, \boldsymbol{t}) = |\boldsymbol{\omega_0 T''}| &\sum_{\boldsymbol{p}\in\mathcal{P}} \mathrm{e}^{\mathrm{i}2\pi\langle \Omega_c'', \boldsymbol{t}-\boldsymbol{pT''}\rangle} \mathrm{sinc}_{\boldsymbol{\Omega''}}(\boldsymbol{t}-\boldsymbol{pT''}) \times \\ &\times \sum_{\boldsymbol{n}\in\mathcal{N}} S_{\Omega_c''}^{\Omega''}(\boldsymbol{n\omega_0}, \boldsymbol{pT''}) \mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{t}-\boldsymbol{C_0}, \boldsymbol{\nu}-\boldsymbol{n\omega_0}\rangle} \mathrm{sinc}_{\boldsymbol{L_0}}(\boldsymbol{\nu}-\boldsymbol{n\omega_0}) \end{aligned}$$

The bandpass filtered spreading function samples are provided in the input

$$\mathtt{S\_np}(\mathtt{n}, \mathtt{p}) = S_{\Omega_c''}^{\Omega''}(\mathtt{N}(\mathtt{n})\boldsymbol{\omega_0}, \mathtt{P}(\mathtt{p})\boldsymbol{T''}). \tag{B.2}$$

Source code

```
function y=BPFS(N,P,S_np,Omega,omega_c,omega,Omega_cBis,OmegaBis,...
                L_0,C_0,t_vec,nu_vec,VERBOSE)
% USAGE: y=BPFS(N,P,S_np,Omega,omega_c,omega,Omega_cBis,OmegaBis,...
%                L_0,C_0,t_vec,nu_vec,VERBOSE)
%
% Computes spreading function at time(s) t_vec and fequency/ies xi.
%
% INPUT:
% N                 = Index set for n -> S_np(n,p)
% P                 = Index set for p -> S_np(n,p)
% S_np              = Bandpass filtered spreading function samples
% Omega             = Bandwidth of the analyzed Gabor window.
% omega             = Bandwidth of the the spreading function eta.
```

```
% Omega_c           = Center frequency of the analyzed Gabor window.
% omega_c           = Center frequency of the spreading function eta.
% Omega_cBis        = Center frequency of interval containing all basis
%                     function frequency supports (see documentation).
% OmegaBis          = Length of interval containing all basis
%                     function frequency supports (see documentation).
% L_0 and C_0       = Length and centerpoint of the shortest interval
%                     containing the support of x -> h(x,t)
% t_vec             = scalar/vector
% nu_vec            = scalar/vector
%
% OUTPUT:
% y = matrix of size length(t_vec)xlength(nu_vec) with
%     y(r,c) = spreading function value at time t_vec(r)
%              and frequency nu=nu_vec(c).

omega_0 = 1/L_0;
T_0=1/Omega;
TBis=1./OmegaBis;
P=P(:).'; % Make row vector
N=N(:);   % Make column vector
t_vec  = t_vec(:).';   % Make row vector
nu_vec = nu_vec(:).';  % Make row vector
nu_vec_len = length(nu_vec);
t_vec_len  = length(t_vec);
y=zeros( length(t_vec), length(nu_vec));

CompStartTime=toc;
r=0;
for t = t_vec
  r=r+1;
  c=0;
  for nu = nu_vec
    c=c+1;
    if abs(nu-omega_c)<omega/2
      for P_ind=1:length(P)
        pTBis=P(P_ind)*TBis;
        y(r,c)= y(r,c) ...
          + exp(i*2*pi*Omega_cBis*(t-pTBis))...
            *SincOmega(OmegaBis,t-pTBis)...
            *sum(S_np(:,P_ind).*exp(i*2*pi*(t-C_0).*(nu-N.*omega_0)) ...
                                .*SincOmega(L_0,nu-N.*omega_0));
      end
```

```
    end
  end
  if VERBOSE
    disp([ int2str(r*nu_vec_len) ' function values (' ...
           num2str(r/t_vec_len*100) ' %) in ' ...
           num2str((toc-CompStartTime)/60) ' minutes.'])
  end
end
y=y.*omega_0.*TBis;
```

B.2 `ChannelSetupAndAnalysis`: The main program

File path: Matlab/ChannelMatrix/ChannelSetupAndAnalysis.m
Short description: The main program.
Additional explanation: See description on page 46.

Source code

```
function ChannelSetupAndAnalysis()

VERBOSE=false;     % If true, then some additional output will be generated
close all hidden; % Close figures
SetPlot(1);       % defined in the very end of this file

%% ================ CHOOSE CHANNEL AND LATTICE PARAMETERS  ================

%% Choose channel type
%channel = 'OFDM';
channel = 'Satellite';
%channel = 'Underwater';

%% Choose box-shaped or exponentially decaying envelope function for the
%  spreading function time dependence
%SprdFctTime ='box';
SprdFctTime ='exp';

%% Choose how c_np depends on n
%SprdFctCoeffType  ='Kronecker delta'; % Gives box-shaped spreading function
%SprdFctCoeffType ='pseudo random';
%SprdFctCoeffType ='uniformly distributed random';
SprdFctCoeffType ='Normally distributed random';
%SprdFctCoeffType ='smooth';

%% Pick size of Gabor lattice
%sys_size='small';
%sys_size='medium';
sys_size='large';

%% Choose minimum number of coefficients in the index set P
%P_min_size=0;%25;
P_min_size=25;

%% Choose fast or detailed plots when in VERBOSE mode:
DraftOrFinal='draft';
```

```
%DraftOrFinal='medium';
%DraftOrFinal='final';

%% Choose a desired TF-lattice unit cell area a*b
%  (the real one will be equal to or smaller than the desired one,
%  due to rounding off b to integer value.
D=1;

%% Choose amplitude threshold for "essential supports"
epsilon=1e-6;

%% =============== COMPUTE AND PLOT SYSTEM MATRIX ELEMENTS  ===============

tic
SPstart=toc;
[Omega_c,Omega,omega_c,omega,Omega_cBis,OmegaBis,...
 B_0,C_0,T_0,T,g,a,b,M,N,P,Q,R,S_np]=...
  SetParameters(channel,SprdFctTime,SprdFctCoeffType,sys_size,...
                P_min_size,'Gaussian',D,epsilon,VERBOSE);
SPtime=toc-SPstart;


MatStart=toc;
 K = CoeffOpMat(Omega_c,Omega,omega_c,omega,Omega_cBis,OmegaBis, ...
               B_0,C_0,a,b,M,N,P,Q,R,S_np,g,epsilon,...
               SprdFctCoeffType,channel,DraftOrFinal,VERBOSE);

disp('Summary:')
disp(['Channel: ' channel]);
disp(['Omega_c  = ' num2str(Omega_c)    ' Hz.'])
disp(['Omega    = ' num2str(Omega)      ' Hz.'])
disp(['omega_c  = ' num2str(omega_c)    ' Hz.'])
disp(['omega    = ' num2str(omega)      ' Hz.'])
disp(['Omega_c" = ' num2str(Omega_cBis) ' Hz.'])
disp(['Omega"   = ' num2str(OmegaBis)   ' Hz.'])
disp(['B_0      = ' num2str(B_0)        ' s.'])
disp(['C_0      = ' num2str(C_0)        ' s.'])
disp(['a        = ' num2str(a)])
disp(['b        = ' num2str(b)])
disp(['a*b      = ' num2str(a*b)])
disp(['K        = [ ' int2str(K(1)) ' : ' int2str(K(end)) '  ].'])
disp(['M        = [ ' int2str(M(1)) ' : ' int2str(M(end)) '  ].'])
disp(['N        = [ ' int2str(N(1)) ' : ' int2str(N(end)) '  ].'])
```

```
disp(['P        = [ ' int2str(P(1)) ' : ' int2str(P(end)) '  ].'])
disp(['Q        = [ ' int2str(Q(1)) ' : ' int2str(Q(end)) '  ].'])
disp(['R        = [ ' int2str(R(1)) ' : ' int2str(R(end)) '  ].'])
disp(['epsilon  = ' num2str(epsilon)])
disp(['SprdFctCoeffType = ' SprdFctCoeffType ])
disp('')
disp('')
disp(['Parameters set in ' num2str(SPtime/60) ' minutes.'])
disp(['Matrix computed in ' num2str((toc-MatStart)/60) ' minutes.'])
disp([''])


%% ========================== INTERNAL FUNCTIONS ==========================

function [Mzp,Mindzp]=ZeropaddRows(M,Mind);
[r,c]=size(M);
if length(Mind)~=r
  error('Nonononononono!!!')
end
Mzp=[zeros(r,2) M zeros(r,2)];
BorderWidth=0.2*(Mind(end)-Mind(1));
D=0.2;
Mindzp=[Mind(1)-BorderWidth 2*Mind(1)-Mind(2) Mind ...
        2*Mind(end)-Mind(end-1) Mind(end)+BorderWidth];

%% ======================== That's all, folks... ========================
```

B.3 `CoeffOpMat`: Compute the Coefficient Operator Matrix

File path: Matlab/ChannelMatrix/CoeffOpMat.m
Short description: Compute the Coefficient Operator Matrix.
Additional explanation: Computes the matrix elements $\mathrm{G}_{\boldsymbol{q}',\boldsymbol{r}';\boldsymbol{q},\boldsymbol{r}} = \langle H g_{\boldsymbol{q},\boldsymbol{r}}, \gamma_{\boldsymbol{q}',\boldsymbol{r}'} \rangle$ of (4.2) for $\boldsymbol{q}, \boldsymbol{q}' \in \mathcal{Q}$ and $\boldsymbol{r}, \boldsymbol{r}' \in \mathcal{R}$ and with $g_{\boldsymbol{q},\boldsymbol{r}}$ and $\gamma_{\boldsymbol{q}',\boldsymbol{r}'}$ defined in (5.1a). Computed here from the bandpass filtered spreading function samples provided in the input `S_np` defined in (B.2).

The computations are done using the algorithm that is described in Section 5.1 with the modifications of the remark on page 29 implemented, but with refinement 2 on page 32 not implemented.

Source code

```
function Hg_support ....
          =CoeffOpMat(Omega_c,Omega,omega_c,omega,Omega_cBis, ...
                      OmegaBis,L_0,C_0,a,b,M,N,P,Q,R,S_np,g,epsilon,...
                      SprdFctCoeffType,channel,DraftOrFinal,VERBOSE)
% USAGE Hg_support=CoeffOpMatrix(Omega_c,Omega,omega_c,omega, ...
%                          Omega_cBis,OmegaBis, L_0,C_0,a,b,M,N,P,Q,R,...
%                          S_np,g,epsilon,SprdFctCoeffType,channel,...
%                          DraftOrFinal,VERBOSE)
% INPUTS
% Omega_c           = Center frequency (Hz) of the synthesis Gabor
%                     window g
% Omega             = Bandwidth (Hz) of synthesis Gabor window g
% omega_c           = Center frequency (Hz) of spreading function eta
% omega             = Bandwidth (Hz) of spreading function eta
% Omega_cBis        = Centerpoint of interval containing all basis
%                     function frequency supports (see documentation).
% C_0 and L_0       = Centerpoint and length of the shortest interval
%                     containing the support of t_0 -> h(t_0,t)
% a and b           : the Gabor system TF-lattice parameters are
%                     a/Omega and b*Omega.
% M                 = Vector containing all m for which g(mT_0) is
%                     nonzero.
% N                 = Index set for n -> S_np(n,p)
% P                 = Index set for p -> S_np(n,p)
% Q                 = We will consider modulations by Q(q)*b*Omega.
% R                 = We will consider shifts by R(r)*a/Omega.
% S_np              = Bandpass filtered spreading function samples
% g                 = Gabor window.
% epsilon           = Amplitude threshold used when restricting
%                     functions to their "essential support".
```

```
% SprFctCoeffSupport= Support of spreading function mapping
%                      t -> eta(t,nu)
%                      (before bandpass filtering with respect to t).
% SprdFctType       = Type of random distribution for elements in S_np.
%                      Text string used for pathnames of saved plots.
% channel           = Type of channel (text string for plot pathname
%                      generation).
% S_np              = Spreading function coefficients
%                      (length(N)xlength(P)-matrix).
% DraftOrFinal      = 'draft', 'mediuum' or 'final'.  Decides how fast
%                      or detailed some plots shall be that are made
%                      if VERBOSE==true
% VERBOSE           = true or false.  Tells whether additional plots
%                      and text output shall be generated

% warning('S_{np} Temporary change here!!!')
% [Nlen,Plen]=size(S_np);
% NhalfLen=floor(Nlen/2);
% PhalfLen=floor(Plen/2);
% S_np=zeros(Nlen,Plen);
% %S_np(round(Nlen/2),round(Plen/2))=1;
% for n=1:Nlen
%   for p=1:Plen
%     S_np(n,p)=exp( log(epsilon)*(  ((n-round(Nlen/2))/NhalfLen)^2 ...
%                                   +((p-round(Plen/2))/PhalfLen)^2 ) );
%   end
% end
% %S_np(round(Nlen/2),round(Plen/2))=1;
% Tbis=1./OmegaBis
% omega_0 = 1/L_0
% omega=omega
% T_0=1./Omega
% T  =1./(Omega+omega)
% Nlen=length(N)
% Plen=length(P)
% aT0=a*T_0
% bOmega=b*Omega
%
% % End of warning ======================================================

omega_0 = 1/L_0;

tic
```

```
T_0=1./Omega;
T  =1./(Omega+omega);
TBis=1./OmegaBis;

%% Choose support
% Support of g : in  M*T_0
% Support of Hg: in  M*T_0+P*TBis
suppHgstart = M( 1 )*T_0+P( 1 )*TBis;
suppHgend   = M(end)*T_0+P(end)*TBis;
Kstart      = floor(suppHgstart/T);
Kend        = ceil( suppHgend  /T);
K=[Kstart:Kend].';

Klen=length(K);  Mlen=length(M);  Nlen=length(N);
Plen=length(P);  Qlen=length(Q);  Rlen=length(R);

% Choose Gabor windows
% ====================
% Contrary to PerMat Deluxe (or however we will name it),
% we do here keep track of sample values and sample points by
% storing them in two separate vectors.  For example, if
% g_cont is defined on R and if g(m*T_0) is nonzero  only for
% m= 3, 4 ,5, then
%
%  M = [       3             4             5     ]  and
%  g = [g_cont(3*T_0) g_cont(4*T_0) g_cont(5*T_0)]


% We choose some Gabor window g with support [-1/2,1/2]:
% g      = ones(Mlen,1); % = Nonzero sample values of g.
StartComputeHgTime=toc;
WinDescr='Gaussian';
%gamma = ones(Klen,1); % = Nonzero sample values of gamma.
[Hg Hg_support] ...  % Set gamma = Hg
    = H(g,epsilon,L_0,C_0,K,M,N,P,S_np, ...
        Omega_c,omega_c,Omega,omega,OmegaBis);

% Normalize S_nk so that Hg has L2-norm approximately = 1:
PrelL2Norm=sqrt(sum(abs(Hg(:)).^2)*T);
S_np=S_np./PrelL2Norm;
Hg=Hg/PrelL2Norm;

TypeOfDual = 1;
```

```
if TypeOfDual == 1
  %Choose gamma = Hg ( = Cruuuuuuuude first choice.)
  gamma = Hg;
  gamma_support=Hg_support;
elseif TypeOfDual == 2
  % Choose gamma to give "approximately dual" Gabor basis
  [gamma,gamma_support] ...
     = GaborDual(Hg,Hg_support, ...
                 Omega_c+omega_c,Omega+omega,a*T_0,b*Omega,Q,R,epsilon);
elseif TypeOfDual == 3
  gamma = g;
  gamma_support = K;
  gamma=resample(Omega_c,Omega,M,g,K*T)
end

if VERBOSE
  figure
  OverSamplingFactor=10;
  Hg_t=[Hg_support(1):1/OverSamplingFactor:Hg_support(end)].'.*T;
  Hg_r=resample(Omega_c+omega_c,Omega+omega,Hg_support,Hg,Hg_t);
  gamma_t ...
    = [gamma_support(1):1/OverSamplingFactor:gamma_support(end)].'.*T;
  gamma_r=resample(Omega_c+omega_c,Omega+omega,gamma_support,gamma, ...
                   gamma_t);
  realArea=sum(abs(real(gamma_t)));
  imagArea=sum(abs(imag(gamma_t)));
  if (imagArea/realArea>epsilon)
    realArea=sum(abs(real(gamma_t)))
    imagArea=sum(abs(imag(gamma_t)))
    figure
    plot(gamma_t,real(gamma_r),'b-',gamma_t,imag(gamma_r),'r:');
    legend('real(\gamma(t))','imag(\gamma(t))')
    xlabel('t')
    error('Where did that imaginary part come from!?  (Se Figure).');
  else
    gamma  =real(gamma  );
    gamma_t=real(gamma_t);
  end
  plot(gamma_t,gamma_r,'b-',Hg_t,Hg_r,'r:');
  axis tight
  legend('\gamma(t)','Hg(t)')
  xlabel('t (s)')
  PathExcludingExtension = ...
```

```
    ['Figures\' channel '\GaborSystem\gammaAndHg' ...
     num2str((toc-StartComputeHgTime)/60) 'minutes'];
multiplot(PathExcludingExtension,'landscape')

end
%% Print and plot some input parameters and functions
if VERBOSE
  disp(['Index vector lengts: Klen = ' int2str(Klen) ...
        ',Mlen = ' int2str(Mlen) ',Nlen = ' int2str(Nlen) ...
        ',Plen = ' int2str(Plen) ',Qlen = ' int2str(Qlen) ...
        ',Rlen = ' int2str(Rlen)'.']);
  disp(['Klen*Mlen*Nlen*Plen*Qlen*Rlen = ' ...
        int2str(Klen*Mlen*Nlen*Plen*Qlen*Rlen) '.'])
  disp(['Preparations finished in ' num2str(toc) ' seconds.'])
  disp('Choosing basis functions.')

  close all hidden % ...to avoid a sometimes occurring Matlab-bug
  SetPlot(2)       % Set large fonts and not-so-thin-lines
  figure
  PlotSpreadingFunction(L_0,C_0,N,P,S_np,SprdFctCoeffType,channel,...
                        omega_c,Omega,omega,Omega_cBis,...
                        OmegaBis,TBis,omega_0,DraftOrFinal,VERBOSE)
end

%% MAIN PROGRAM: Compute matrix elements ==========================
if VERBOSE
  disp('Starts computing <Hg_{q,r},gamma_{q'',r''}>...')
end
%close all hidden % Save some memory

MatrixCompTimeStart=toc;

PlotTimeStart=toc;
GaborSystemSize=Qlen*Rlen;
% Operator matrix (G) and distances (D) for off-diagonal decay plot:
G=zeros(GaborSystemSize,GaborSystemSize);
D=zeros(GaborSystemSize,GaborSystemSize);
row=0;

ElementNr=1;

Omega_cqmax=Omega_c+Q(end)*b*Omega
for r=1:Rlen   % raT_0
```

```
for q=1:Qlen %qbOmega
  row=row+1;
  col=0;
  Omega_cq =Omega_c+Q(q)*b*Omega;
  omega_q = Omega_cq/Omega_cqmax*omega;
  C_0q=Omega_cqmax/Omega_cq*C_0;
  L_0q=Omega_cqmax/Omega_cq*L_0;

  [g_qr NewM g_NewOmega_c] ...
    = TFshift(g ,M, Omega_c, Omega, R(r)*a, Q(q)*b);

  Kshifted = [ floor((suppHgstart + R(r)*a*T_0)/T): ...
               ceil( (suppHgend   + R(r)*a*T_0)/T) ].';
  [Hg_qr Hg_qr_support] ...
      = H(g_qr,epsilon,L_0q,C_0q,Kshifted,NewM,N,P,S_np, ...
          g_NewOmega_c,omega_c,Omega,omega,OmegaBis);
   if sum(abs(Hg_qr))==0
     error('xxx')
   end
  for rprime=1:Rlen
    for qprime=1:Qlen
      ElementNr=ElementNr+1;
      % Apply TF-shifts to gamma:
      col=col+1;
      [gamma_qr gamma_qr_support gamma_NewOmega_c] ...
        = TFshiftAndResample(gamma,gamma_support, ...
                             Omega_c+omega_c,Omega+omega, ...
                             R(rprime)*a*T_0,Q(qprime)*b*Omega);
      G(row,col)= l2InnerProd(Hg_qr, Hg_qr_support, gamma_qr,...
                              gamma_qr_support, ...
                              g_NewOmega_c+omega_c, ...
                              gamma_NewOmega_c,Omega+omega);
      D(row,col)=sqrt( (Q(q)-Q(qprime))^2 + (R(r)-R(rprime))^2);
    end
  end
  if r==Rlen && q==Qlen
    figure(42)
    subplot(1,4,1)
    plot(Hg_qr_support*T,real(Hg_qr),'b',gamma_qr_support*T,...
         real(gamma_qr),'r:');
    xlabel('t (s)')
    axis tight
    legend('real(Hg_{q,r})','real(\gamma_{q,r})')
```

```
      subplot(1,4,2)
      plot(Hg_qr_support*T,imag(Hg_qr),'b',gamma_qr_support*T,...
           imag(gamma_qr),'r:');
      xlabel('t (s)')
      axis tight
      legend('imag(Hg_{q,r})','imag(\gamma_{q,r})')
      subplot(1,4,3)
      plot(Hg_qr_support*T,abs(Hg_qr),'b',gamma_qr_support*T,...
           abs(gamma_qr),'r:');
      xlabel('t (s)')
      axis tight
      legend('abs(Hg_{q,r})','abs(\gamma_{q,r})')
      subplot(1,4,4)
      plot(Hg_qr_support*T,angle(Hg_qr),'b',gamma_qr_support*T,...
           angle(gamma_qr),'r:');
      xlabel('t (s)')
      axis tight
      legend('angle(Hg_{q,r})','angle(\gamma_{q,r})')
      grid on
      axis tight
    end
    if VERBOSE
      disp([Int2str(ElementNr) '  matrix elements (=' ...
            num2str(ElementNr/(GaborSystemSize.^2)*100) ...
            '% of the matrix) computed in ' ...
            num2str((toc-MatrixCompTimeStart)/60) ' minutes.']);
    end
  end
end

close all hidden

%% Plot the channel operator matrix ================================
SetPlot(2); % Set larger fontsize and wider lines in plots

figure
PcolorFull(abs(G));
figsize('a4landscape')
axis square
colorbar
[r c]=size(G);
xlabel('\itn_{q{\prime},r{\prime}}');
ylabel('\itn_{q,r}');
```

```
zlabel('|G_{\itq,r;q{\prime},r{\prime}}|')
PathExcludingExtension = ...
  ['Figures\' channel '\ChannelOpMatrix\matrix' ...
    num2str(toc/60) 'minutes'];
multiplot(PathExcludingExtension,'landscape')

close all hidden

figure
PcolorFull(log10(abs(G)));
figsize('a4landscape')
axis square
colorbar
[r c]=size(G);
xlabel('\itn_{q{\prime},r{\prime}}');
ylabel('\itn_{q,r}');
zlabel('|G_{\itq,r;q{\prime},r{\prime}}|')
PathExcludingExtension = ...
  ['Figures\' channel '\ChannelOpMatrix\log10-matrix' ...
    num2str(toc/60) 'minutes'];
multiplot(PathExcludingExtension,'landscape')


% %% Plot l^2-distances ============================================
% figure
% PcolorFull(D);
% figsize('a4landscape')
% axis square
% colorbar
% %surf(D);
% %[C,h]=contour(D);
% %set(h,'ShowText','on')
% axis tight
% xlabel('\itn_{q{\prime},r{\prime}}');
% ylabel('\itn_{q,r}');
% zlabel('||\it(r,q)-(r{\prime},q{\prime})||_{\itl_2}')
% % title(['System matrix for ' WinDescr ' Gabor window and ' ...
% %         SprdFctCoeffType ' spreading function coeffients (' ...
% %         num2str((toc-PlotTimeStart)/60) ' minutes)'])
% PathExcludingExtension = ...
%   ['Figures\' channel '\ChannelOpMatrix\L2-distances' ...
%     num2str(toc/60) 'minutes'];
% multiplot(PathExcludingExtension,'landscape')
```

```
close all hidden

%% Prepare for off-diagonal decay plots ============================

G=G(:);          % Matrix entries
D=D(:);          % Distances
[D,ind]=sort(D); % Sort in order of increasing distance
G=G(ind);        % Same sorting


NrOfBins = 100;


G_average=zeros(NrOfBins,1);
BinWalls=linspace(D(1),D(end),NrOfBins+1);
D_remaining=D;
G_remaining=abs(G);
for BinNr = 1:NrOfBins
  inds=find(D_remaining<=BinWalls(BinNr+1));
  if length(inds)>0;
    ind=max(inds);
      G_average(BinNr)=sum(abs(G_remaining(1:ind)))./ind;
    if (BinNr <NrOfBins)
      D_remaining=D_remaining(ind+1:end);
      G_remaining=G_remaining(ind+1:end);
    end % if
  end % if
end % for
      if (ind < length(D_remaining))
        error('This should not have happened...')
      end % if



ind=1;
tmp=0;
NrOfTerms=0;
for n=1:length(G)
  if abs(D(n)-D(ind)) <= 10*eps
    tmp=tmp+abs(G(n));     % TEST!!!
    NrOfTerms=NrOfTerms+1;
  else
```

```
    G(ind)=tmp/NrOfTerms;  % TEST!!!
    ind=ind+1;
    NrOfTerms=1;
    D(ind)=D(n);
    tmp=abs(G(n));         % TEST!!!
  end
end
G(ind)=tmp/NrOfTerms;  % TEST!!!
D(ind)=D(end);
D=D(1:ind);
G=G(1:ind);
PathExcludingExtension = ...
  ['Figures\' channel '\ChannelOpMatrix\Off-diagDecay' ...
    num2str(toc/60) 'minutes'];
save([PathExcludingExtension '.mat'], 'D', 'G')

%% Off-diagonal decay, binned linear plot... ========================
figure
x=(BinWalls(1:end-1)+BinWalls(2:end))/2;
bar(x,G_average)
xlabel('||\it(r,q)-(r{\prime},q{\prime})||_{\itl_2}')
ylabel('Average amplitude')
axis tight
PathExcludingExtension = ...
  ['Figures\' channel '\ChannelOpMatrix\Off-diagDecayBins' ...
    num2str(toc/60) 'minutes'];
multiplot(PathExcludingExtension,'landscape')

%% Off-diagonal decay, linear plot... ===============================
figure
plot(D,G,'r.:')
xlabel('||\it(r,q)-(r{\prime},q{\prime})||_{\itl_2}')
ylabel('Average amplitude')
axis tight

PathExcludingExtension = ...
  ['Figures\' channel '\ChannelOpMatrix\Off-diagDecayLin' ...
    num2str(toc/60) 'minutes'];
multiplot(PathExcludingExtension,'landscape')
if VERBOSE
  disp(['...and those computations took ' num2str(toc/60) ' minutes.'])
  disp('Thank you for your patience.  :-)')
end
```

```
%% Off-diagonal decay, logarithmic plot... ==========================
figure
ind = find (abs(G)>0)
semilogy(D(ind),G(ind),'b.:')
xlabel('||\it(r,q)-(r{\prime},q{\prime})||_{\itl_2}')
ylabel('Average amplitude')
axis tight
PathExcludingExtension = ...
  ['Figures\' channel '\ChannelOpMatrix\Off-diagDecayLog' ...
    num2str(toc/60) 'minutes'];
multiplot(PathExcludingExtension,'landscape')

D=D(ind);
G=G(ind);

%% ======================== INTERNAL FUNCTIONS ======================

function y=PcolorFull(varargin)
% Use: y=pcolor2(C)
%      y=pcolor2(X,Y,C)
%
% Description:  Modified version of the Matlab command pcolor.
% The difference is that PcolorFull does not remove the last row and
% column of the matrix before plotting.

if (nargin==1)
  C=varargin{1};
  [m,n]=size(C);
  y=pcolor([C ones(m,1).*C(1,1)
            ones(1,n+1).*C(1,1)]);
elseif(nargin==3)
  X=varargin{1};
  Y=varargin{2};
  C=varargin{3};

  X=X(:); % Make column vector
  Y=Y(:); % Make column vector

  [m,n]=size(C);
  X=[X ; X(end)+(X(end)-X(end-1))];
  Y=[Y ; Y(end)+(Y(end)-Y(end-1))];
  y=pcolor(X,Y,[C ones(m,1).*C(1,1)
```

```
                ones(1,n+1).*C(1,1)]);
else
  error('WrongINcorrect number of inputs.')
end;
shading flat
```

B.4 CutToEssSupp: Truncate a function to its essential support

File path: Matlab/ChannelMatrix/CutToEssSupp.m
Short description: Truncate a function to its essential support.
Additional explanation: Truncate a function to its essential support, as defined on page 5.

Source code

```
function [gc_sample_ind,gc_samples] ...
          = CutToEssSupp(epsilon,OversampFact,g_centerfreq, ...
                         g_bandwidth,g_sample_ind,g_samples)
% Reduces the support of a function g sampled at Nyquist frequency
% by applying an amplitude threshold epsilon on g sampled at
% OverSamplingFact times the Nyquist frequency
%
% USAGE: [gc_sample_ind,gc_samples] ...
%          = CutToEssSupp(epsilon,OversampFact,g_centerfreq, ...
%                         g_bandwidth,g_sample_ind,g_samples)
% INPUT
% epsilon      = Cutting threshold
% OversampFact = Oversampling factor
% g_centerfreq = Center frequency of g.
% g_bandwidth  = Bandwidth of g.
% g_sample_ind = Integer sample indices.
% g_samples    = Samples of g such that the mth sample
%                is sampled at "time" g_sample_ind(m)*T
%                with T = 1/g_bandwidth
%
% OUTPUT
% g = Number, vector or matrix of the same size as t, containing the
%     computed new samples

T = 1/g_bandwidth;

t=[g_sample_ind(1)*T:T/OversampFact:g_sample_ind(end)*T].';
g=resample(g_centerfreq,g_bandwidth,g_sample_ind,g_samples,t);
absg=abs(g);
ind = find( absg./max(absg(:))>=epsilon );
tstart=t(min(ind));
tend  =t(max(ind));
ind=find( ((g_sample_ind*T)>=tstart) & ((g_sample_ind*T)<=tend));
```

```
gc_sample_ind=g_sample_ind(min(ind):max(ind));
gc_samples   =g_samples(min(ind):max(ind));
if length(gc_samples) == length(g_samples)
  warning(['Sorry,no cutting was possible.  Smallest relative amplitude was '...
           num2str(min(absg(:)./max(absg(:)))./epsilon) '\epsilon.'])
end
```

B.5 figsize: Set the orientation and size of the current figure

File path: Matlab/ChannelMatrix/figsize.m
Short description: Set the orientation and size of the current figure.
Additional explanation:

Source code

```
function figsize(size)
% Sets the size of the current figure
%
% Use: figsize(size)

if strcmp(size,'a4')
  %orient portrait
  set(gcf,'PaperType','a4');
  wh=get(gcf,'PaperSize');
  width=wh(1);
  height=wh(2);
  set(gcf,'PaperPosition',[0.25,0.25,width,height]);
elseif strcmp(size,'a4landscape')
  %orient landscape
  set(gcf,'PaperType','a4');
  wh=get(gcf,'PaperSize');
  width=wh(2);
  height=wh(1);
  set(gcf,'PaperPosition',[0.25,0.25,width,height]);
elseif strcmp(size,'a4HalfLandscape')
  set(gcf,'PaperType','a4');
  wh=get(gcf,'PaperSize');
  width=wh(2);
  height=wh(1)/2;
  set(gcf,'PaperPosition',[0.25,0.25,width,height]);
elseif strcmp(size,'subplot13')
  set(gcf,'PaperType','a4');
  wh=get(gcf,'PaperSize');
  width=wh(1);
  height=wh(2)/6;
  set(gcf,'PaperPosition',[0.25,0.25,width,height]);
elseif strcmp(size,'subplot12')
  set(gcf,'PaperType','a4');
  wh=get(gcf,'PaperSize');
  width=wh(1);
```

```
  height=wh(2)/4;
  set(gcf,'PaperPosition',[0.25,0.25,width,height]);
elseif strcmp(size,'subplot21')
  set(gcf,'PaperType','a4');
  wh=get(gcf,'PaperSize');
  width=wh(1);
  height=wh(2)/3;
  set(gcf,'PaperPosition',[0.25,0.25,width,height]);
elseif strcmp(size,'subplot22')
  set(gcf,'PaperType','a4');
  wh=get(gcf,'PaperSize');
  width=wh(1);
  height=wh(2)/2;
  set(gcf,'PaperPosition',[0.25,0.25,width,height]);
else
  error(['Plotsize ''' size ''' not known.']);
end
```

B.6 FTBPFS: Compute $\widehat{S_H^{\Omega_c,\Omega}(\cdot,t)}(t_0)$

File path: Matlab/ChannelMatrix/FTBPFS.m

Short description: Compute $\widehat{S_H^{\Omega_c,\Omega}(\cdot,t)}(t_0)$.

Additional explanation: Computes $\widehat{S_H^{\Omega_c,\Omega}(\cdot,t)}(t_0)$ using the formula (4.16). Computed here with coefficients given as the input S_np defined in (B.2), page (B.2).

Source code

```
function y=FTBPFS(Omega_c,omega_c,Omega,omega,OmegaBis, ...
                  B_0,C_0,N,P,S_np,t,t_0)
% y=FTBPFS(Omega_c,omega_c,Omega,omega,OmegaBis, ...
%                  B_0,C_0,N,P,S_np,t,t_0)
%
%  Computes Fourier transform of bandpass filtered spreading function
%  at time t and fequency xi
%
% USAGE:
%
% INPUT:
% Omega_c             = Center frequency of the analyzed Gabor window.
% omega_c             = Center frequency of the spreading function eta.
% Omega               = Bandwidth of the analyzed Gabor window.
% omega               = Bandwidth of the the spreading function eta.
% OmegaBis            = Length of interval containing all basis
%                       function frequency supports (see documentation).
% B_0 and C_0         = Length and centerpoint of the shortest interval
%                       containing the support of t_0 -> h(t_0,t)
% N                  = Index set for n -> S_np(n,p)
% P                  = Index set for p -> S_np(n,p)
% S_np               = Bandpass filtered spreading function samples
% t                    = real number
% t_0                    = real number

T_0=1/Omega;
TBis=1./OmegaBis;
omega_0=1/B_0;
P=P(:).'; % Make row vector
N=N(:);   % Make column vector

y=0;
tmint0 = t-t_0; % Compute only once to save time
```

```
if ( (C_0-B_0/2<=tmint0) && (tmint0<=C_0+B_0/2) )
  for P_ind=1:length(P)
    p=P(P_ind);
    pTBis = p*TBis;
    y = y +...
        exp(i*2*pi*Omega_c*(t-pTBis))...
        *SincOmega(Omega,t-pTBis)...
        *sum(S_np(:,P_ind).*exp(i*2*pi*(tmint0-pTBis).*N.*omega_0));
  end
  y=y*omega_0*TBis;
else
    warning('Oh dearie me, this should really never happen...')
end
```

B.7 `GaborDual`: Compute the dual of a Gabor Riesz basis

File path: Matlab/ChannelMatrix/GaborDual.m
Short description: Compute the dual of a Gabor Riesz basis.
Additional explanation: **Not carefully tested!!!** *This function was at first considered for use but then never needed in this report due to the good diagonalization properties of Gaussian windows in use. It might, however, be modified, tested and included in future version of the software for use with other windows g.*

Consider a Gabor Riesz basis for some subspace of $L^2(\mathbb{R}^d)$ with the elements

$$g_{\boldsymbol{q},\boldsymbol{r}} \stackrel{\text{def}}{=} T_{\boldsymbol{ra}} M_{\boldsymbol{qb}} g, \qquad \mathcal{Q}, \mathcal{R} \subseteq \mathbb{Z}^d$$

Fix some $(\boldsymbol{q_0}, \boldsymbol{r_0}) \in \mathbb{Z}^2$. For $\gamma L^2(\mathbb{R}^d)$ it follows that γ is dual to

-series expansion $\gamma \stackrel{\text{def}}{=} \sum_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d} c_{\boldsymbol{q},\boldsymbol{r}} g_{\boldsymbol{q},\boldsymbol{r}}$. recall from page 9 that γ is the canonical dual window if and only if the coefficient sequence $(c_{\boldsymbol{q},\boldsymbol{r}})_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d}$ is the unique minimum l^2-norm solution of

$$\begin{aligned}\delta_{(\boldsymbol{q_0},\boldsymbol{r_0})}(\boldsymbol{q'},\boldsymbol{r'}) = \langle \gamma_{\boldsymbol{q_0},\boldsymbol{r_0}}, g_{\boldsymbol{q'},\boldsymbol{r'}} \rangle &= \left\langle \sum_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d} c_{\boldsymbol{q},\boldsymbol{r}} g_{\boldsymbol{q}+\boldsymbol{q_0},\boldsymbol{r}+\boldsymbol{r_0}}, g_{\boldsymbol{q'},\boldsymbol{r'}} \right\rangle \\ &= \sum_{\boldsymbol{q},\boldsymbol{r}\in\mathbb{Z}^d} c_{\boldsymbol{q},\boldsymbol{r}} \left\langle g_{\boldsymbol{q}+\boldsymbol{q_0},\boldsymbol{r}+\boldsymbol{r_0}}, g_{\boldsymbol{q'},\boldsymbol{r'}} \right\rangle,\end{aligned}$$

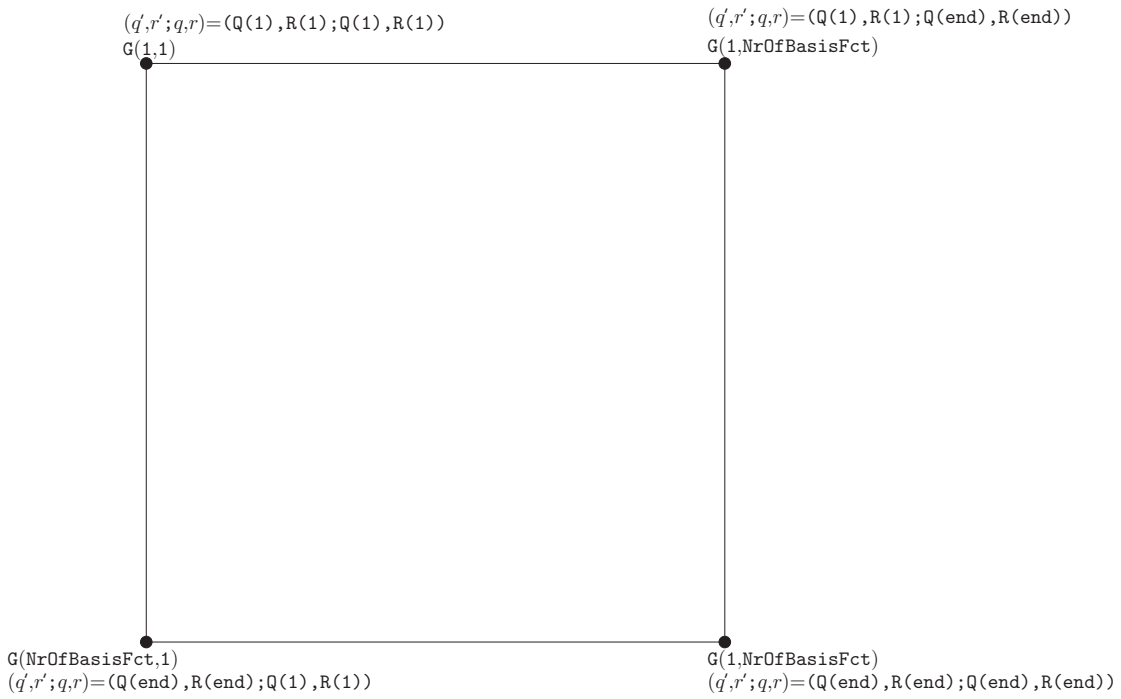


Figure 14: Indexing of `G` and the corresponding values of $(\boldsymbol{q'},\boldsymbol{r'};\boldsymbol{q},\boldsymbol{r})$.

or in biinfinite vector notation with indices in $\mathbb{Z}^d \times \mathbb{Z}^d$,

$$\mathrm{G}\boldsymbol{c} = \boldsymbol{\delta}_{(\mathbf{0},\mathbf{0})}, \qquad \text{where} \qquad (\mathrm{G})_{\boldsymbol{q}',\boldsymbol{r}';\boldsymbol{q},\boldsymbol{r}} = \langle g_{\boldsymbol{q},\boldsymbol{r}}, g_{\boldsymbol{q}',\boldsymbol{r}'} \rangle = \langle g_{\boldsymbol{q}-\boldsymbol{q}',\boldsymbol{r}-\boldsymbol{r}'}, g_{\mathbf{0},\mathbf{0}} \rangle$$

and

$$(\boldsymbol{\delta}_{(\mathbf{0},\mathbf{0})})_{(\boldsymbol{q}',\boldsymbol{r}')} \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } (\boldsymbol{q}',\boldsymbol{r}') = (\mathbf{0},\mathbf{0}), \\ 0 & \text{otherwise.} \end{cases}$$

G is known as the *Gram matrix*. Figure (14) together with the code shows how G is indexed.

Source code

```
function [g_dual_samples,g_dual_sample_ind] ...
          = GaborDual(g_samples,g_sample_ind, ...
                      g_center_freq,g_bandwidth,a,b,Q,R,epsilon)
% Computes the canonical dual of a Gabor window g
% that is completely determined by a finite number of
% input Nyquist frequency samples.
% QUICK & DIRTY implementation for finite index sets.
% E.g. Fourier series approach should be much faster!
%
% USAGE: [g_dual_samples,g_dual_sample_ind]
%          = GaborDual(g_samples,g_sample_ind, ...
%                      g_center_freq,g_bandwidth,a,b,Q,R)
%
% INPUTS
% g_samples     = Samples of g such that the mth sample
%                 is sampled at "time" g_sample_ind(m)*T
%                 with T = 1/g_bandwidth, where ...
% g_sample_ind  = Integer sample indices.
% g_center_freq = Center frequency of g.
% g_bandwidth   = Bandwidth of g.
% a             = TF-lattice      time-constant.
% b             = TF-lattice frequency-constant.
% Q             = Integer frequency-shifts in TF-lattice.
% R             = Integer      time-shifts in TF-lattice.
% epsilon       = Amplitude threshold for truncation to
%                 epsilon-essential support.
%
% OUTPUT
% g_dual_samples     = Sample values of the dual window.
% g_dual_sample_ind  = Sample indices of the dual window.
T=1/g_bandwidth;
Qlen=length(Q);
```

```
Rlen=length(R);
q0=Q(round(Qlen/2));
r0=R(round(Rlen/2));

NrOfBasisFcts=Qlen*Rlen;
G=zeros(NrOfBasisFcts,NrOfBasisFcts);

%% Compute row 1  of the Gram matrix ================================
qprime=Q(1); rprime=R(1);
row=1;
col=0;
[g_qprimerprime_samples,...
 g_qprimerprime_sample_ind,...
 g_qprimerprime_center_freq] ...
  = TFshiftAndResample(g_samples,g_sample_ind, ...
                       g_center_freq,g_bandwidth,rprime*a,qprime*b);

for r=R(:).';
  for q=Q(:).';
    col=col+1;
    [g_qr_samples,g_qr_sample_ind,g_qr_center_freq] ...
      = TFshiftAndResample(g_samples,g_sample_ind, ...
                           g_center_freq,g_bandwidth,(r+r0)*a,(q+q0)*b);
    G(row,col)=l2InnerProd(g_qr_samples,g_qr_sample_ind, ...
                           g_qprimerprime_samples,...
                           g_qprimerprime_sample_ind,T);
  end
end

%% Compute Gram matrix elements to the right of the main diagonal ====
for row = 2:NrOfBasisFcts-1
  G(row,row+1:end)=G(1,2:end-(row-1));
end

%% Compute the last row of the Gram matrix ==========================
qprime=Q(end); rprime=R(end);
row=NrOfBasisFcts;
col=0;
[g_qprimerprime_samples,...
 g_qprimerprime_sample_ind,...
 g_qprimerprime_center_freq] ...
  = TFshiftAndResample(g_samples,g_sample_ind, ...
                       g_center_freq,g_bandwidth,rprime*a,qprime*b);
```

```
for r=R(:).';
  for q=Q(:).';
    col=col+1;
    if ( (r==r0) && (q==q0))
      zero_ind= col;
    end
    [g_qr_samples,g_qr_sample_ind,g_qr_center_freq] ...
      = TFshiftAndResample(g_samples,g_sample_ind, ...
                           g_center_freq,g_bandwidth,(r+r0)*a,(q+q0)*b);
    G(row,col)=l2InnerProd(g_qr_samples,g_qr_sample_ind, ...
                           g_qprimerprime_samples,...
                           g_qprimerprime_sample_ind,T);
  end
end




%% Compute Gram matrix elements to the left of the main diagonal =====
for row = 2:NrOfBasisFcts-1
  G(row,1:row-1)=G(end,end-(row-1):end-1);
end

%% Compute coefficients ==============================================

pinvG=pinv(G);       % Invertiing the Gram matrix
clear G;
c=pinvG(:,zero_ind); % =  pinvG*(Kronecker Delta)
clear pinvG;

%% Compute the dual window from the coefficients =====================
g_dual_ind_start  = floor(R( 1 )/T)+g_sample_ind( 1 );
g_dual_ind_end    = floor(R(end)/T)+g_sample_ind(end);
g_dual_sample_ind = [g_dual_ind_start:g_dual_ind_end].';
g_dual_samples    = zeros(size(g_dual_sample_ind));

c_ind=0;
for r=R(:).';
  for q=Q(:).';
    c_ind=c_ind+1;

    [g_qr_samples,g_qr_sample_ind,g_qr_center_freq] ...
```

```
      = TFshiftAndResample(g_samples,g_sample_ind, ...
                           g_center_freq,g_bandwidth,r*a,q*b);
    indstart = find(g_dual_sample_ind == g_qr_sample_ind( 1 ));
    indend   = find(g_dual_sample_ind == g_qr_sample_ind(end));
    g_dual_samples(indstart:indend) ...
      = g_dual_samples(indstart:indend) + c(c_ind)*g_qr_samples;
  end
end
ind=find(abs(g_dual_samples)>=epsilon);
g_dual_sample_ind = g_dual_sample_ind(ind(1):ind(end));
g_dual_samples = g_dual_samples(ind(1):ind(end));
```

B.8 H: Compute samples $(Hg)(kT_\gamma)$ using (4.12)

File path: Matlab/ChannelMatrix/H.m
Short description: Compute samples $(Hg)(\boldsymbol{kT_\gamma})$ using (4.12).
Additional explanation: Function for computing the samples

$$(Hg)(\boldsymbol{kT_\gamma}) = |\boldsymbol{T_g}| \sum_{\boldsymbol{m}\in\mathcal{M}} g(\boldsymbol{mT_g}) \left(S_H^{\boldsymbol{\Omega_c},\boldsymbol{\Omega}}(\cdot, \boldsymbol{kT_\gamma} - \boldsymbol{mT_g})\right)\widehat{\ }(-\boldsymbol{mT_g}).$$

Source code

```
function [Hg Hg_support]=H(g,epsilon,L_0,C_0,K,M,N,P,S_np, ...
                           Omega_c,omega_c,Omega,omega,OmegaBis)
% USAGE: [Hg Hg_support]=H(g,epsilon,L_0,C_0,K,M,N,P,S_np, ...
%                           Omega_c,omega_c,Omega,omega,OmegaBis)
% Function for computating samples of Hg
%
% INPUT:
% g         = Input sample values of some function g_cont:
%             g(m)=g_cont(M(m)*T_g)
% epsilon   = Amplitude threshold used for restricting the output to
%             its "essental support"
% C_0 & L_0 = Centerpoint ansd length of the shortest interval
%             containing the support of t_0 -> h(t_0,t)
% K         = Index set for which Hg shall be computed.
% M         = Index set for g.
% N         = Index set for spreading function trig poly coefficients.
% P         = Index set for nonzero sample velues of spreading function
% S_np      = Spreading function coefficients
%             (length(N)xlength(P)-matrix).
% Omega_c   = Center frequency of the analyzed Gabor window.
% omega_c   = Center frequency of the spreading function eta.
% Omega     = Bandwidth of the analyzed Gabor window.
% omega     = Bandwidth of the the spreading function eta.
% OmegaBis  = Length of interval containing all basis function
%             frequency supports (see the software documentation).
%
% OUTPUT:
% Hg         = Vector of the same dimension and length as K with
%              Hg(k) = sample value of corresponding continuous
%                      function Hg_cont.  Hg(k)=Hg_cont(K(k)*T_gamma).
% Hg_support = index such that Hg(j)=0 for j not in Hg_support.
```

```
T_g=1/Omega;
T_gamma=1/(Omega+omega);


Klen=length(K);
Mlen=length(M);

Hg=zeros(size(K));
K=K(:); % Make column vector
g=g(:); % Make column vector

for k= 1:Klen
  for m= 1:Mlen
    FBS=FTBPFS(Omega_c,omega_c,Omega,omega,OmegaBis, ...
               L_0,C_0,N,P,S_np,K(k)*T_gamma-M(m)*T_g,-M(m)*T_g);
    if FBS==0
      ind = find( (K*T_gamma<C_0-L_0/2) | (K*T_gamma>C_0+L_0/2) )
      warning(['Element(s) nr ' num2str(ind) ' of ' ...
                int2str(length(K)) ...
                'in K outside expecteded range! (' ...
               'Perhaps L_0 is too small?)']);
    end
    Hg(k)=Hg(k)+g(m)*FBS;
  end
end
Hg=T_g*Hg;
% Cut g and Hg to the essential support of Hg :
OversamplFact=2;
[Hg_support,Hg] = ...
   CutToEssSupp(epsilon,OversamplFact,Omega_c+omega_c,Omega+omega,K,Hg);
```

B.9 `l2Innerprod`: Computes the l^2 inner product (4.8a)

File path: Matlab/ChannelMatrix/l2Innerprod.m
Short description: Computes the l^2 inner product (4.8a).
Additional explanation: This function computes the inner products (4.8), that is, the inner products

$$\langle u, v\rangle_{L^2(\mathbb{R}^d)} = |\boldsymbol{T}| \sum_{\boldsymbol{k}\in\mathcal{I}_u} u(\boldsymbol{kT})\overline{v_{\mathrm{bpf}}(\boldsymbol{kT})},$$

with $v_{\mathrm{bpf}} = v$ if $\boldsymbol{C_u} = \boldsymbol{C_v}$ and

$$v_{\mathrm{bpf}}(\boldsymbol{kT}) = |\boldsymbol{T}| \sum_{\boldsymbol{k'}\in\mathcal{I}_v} v(\boldsymbol{k'T})\mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{C_{uv}}, \boldsymbol{(k-k')T}\rangle}\mathrm{sinc}_{\boldsymbol{B_{uv}}}(\boldsymbol{(k-k')T}).$$

otherwise.

Source code

```
function uv = l2InnerProd(u,u_ind,v,v_ind,C_u,C_v,B)
% USAGE: uv = l2InnerProd(u,u_ind,v,v_ind,C_u,C_v,B)
%
% INPUT:
% u     = Vector of sample values of some function u_cont, ...
% u_ind = ... such that u(k) = u_cont(u_ind*T)
% v     = Vector of sample values of some function v_cont, ...
% v_ind = ... such that v(k) = v_cont(v_ind*T)
% C_u   = Center frequency of u.
% C_v   = Center frequency of v.
% B     = Bandwidth of u = Bandwidth of v.
%
% OUTPUT
% y = L_2 inner product of u and v
%
% NOTE: u_ind and v_ind MUST be row- or column vectors of the type
%         [a a+1 ... b-1 b]
%       but they do not need to be of equal length or even overlap.
%       (Yes, it's a waste of space to save more than the starting
%        point of each index sets, but this is just a quick & dirty
%        test-implementation without intentions to do any clever
%        programming. =) )

T=1/B;

u_ind=u_ind(:); % Make column vector
```

```
u=u(:);         % Make column vector
v_bpf=zeros(size(u_ind));
v_ind_len=length(v_ind);

JointFreqSuppStart = max(C_u-B/2,C_v-B/2);
JointFreqSuppEnd   = min(C_u+B/2,C_v+B/2);
B_uv=JointFreqSuppEnd-JointFreqSuppStart;
C_uv=(JointFreqSuppEnd+JointFreqSuppStart)/2;
if B_uv <= 0
  uv=0;  %  No overlap of frequency support
else
  for kprime_ind = 1:v_ind_len
    kprime=v_ind(kprime_ind);
    k_minus_kprime_T=(u_ind-kprime).*T;
    v_bpf = v_bpf ...
           + v(kprime_ind)*exp(i*2*pi*C_uv.*k_minus_kprime_T) ...
             .*SincOmega(B_uv,k_minus_kprime_T);
  end
  v_bpf=T*v_bpf;
  uv=T*sum(u.*conj(v_bpf));
end
```

B.10 `multiplot`: Command for saving the current figure in .eps and .emf format

File path: Matlab/ChannelMatrix/multiplot.m
Short description: Command for saving the current figure in .eps and .emf format.
Additional explanation:

Source code

```
function multiplot(PathExcludingExtension,orientation)
% Saves current figure in sizes and (colour or black and white) file
% formats suitable for importing into  LaTeX and Powerpoint documents.

if nargin==1 || strcmp(orientation,'portrait')
  orient portrait
  figsize('a4')
elseif strcmp(orientation,'a4HalfLandscape')
  orient portrait
  figsize('a4HalfLandscape')
else
  orient portrait
  figsize('a4landscape')
end
%print('-depsc', [PathExcludingExtension  '.eps'])
print('-depsc2',[PathExcludingExtension '2.eps'])
colormap(gray)
%print('-deps' ,[PathExcludingExtension  'BW.eps'])
print('-deps2',[PathExcludingExtension 'BW2.eps'])

figsize('a4landscape')
%print('-r300','-dmeta',[PathExcludingExtension '.emf'])
print('-dmeta',[PathExcludingExtension '.emf'])


if nargin==1 || strcmp(orientation,'landscape')
  %orient landscape
  orient portrait
  figsize('a4landscape')
elseif strcmp(orientation,'a4HalfLandscape')
  orient portrait
  figsize('a4HalfLandscape')
else
  orient portrait
```

```
  figsize('a4')
end
figsize('a4landscape')
%print('-r300','-dmeta',[PathExcludingExtension '.emf'])
print('-dmeta',[PathExcludingExtension 'BW.emf'])
```

B.11 `PlotSpreadingFunction`: Plots bandpass filtered spreading function

File path: Matlab/ChannelMatrix/PlotSpreadingFunction.m
Short description: Plots bandpass filtered spreading function.
Additional explanation: Plots the bandpass filtered spreading function $S_{\Omega_c''}^{\Omega''}$, which is computed here by calling the function `BPFS` (see Appendix B.1 with input coefficients `S_np` defined in (B.2), page 47).

Source code

```
function PlotSpreadingFunction(...
           L_0,C_0,N,P,S_np,SprdFctCoeffType,channel,omega_c,Omega, ...
           omega,Omega_cBis,OmegaBis,TBis,omega_0,DraftOrFinal,VERBOSE)
% USAGE: PlotSpreadingFunction(...
%           L_0,C_0,N,P,S_np,SprdFctCoeffType,channel,omega_c,Omega, ...
%           omega,Omega_cBis,OmegaBis,TBis,omega_0,DraftOrFinal,VERBOSE)
%
% Plots the bandpass filtered spreading function.
%
% INPUTS:
% L_0 and C_0       = Length and centerpoint of the shortest interval
% N                 = Index set for n -> S_np(n,p)
% P                 = Index set for p -> S_np(n,p)
% S_np              = Bandpass filtered spreading function samples
% SprdFctCoeffType  = Type of random distribution for elements in S_np.
%                     Text string used for pathnames of saved plots.
% channel           = Type of channel
%                     (text string for plot pathname generation).
% omega_c           = Center frequency of the spreading function eta.
% Omega             = Bandwidth of the analyzed Gabor window.
% omega             = Bandwidth of the the spreading function eta.
% Omega_cBis        = Center frequency of interval containing all basis
%                     function frequency supports (see documentation).
% OmegaBis          = Length of interval containing all basis
%                     function frequency supports (see documentation).
% TBis              = 1/OmegaBis.  (Hmmmm... why was this imported?)
% omega_0           = 1/L_0.  (Hmmmm... why was this imported?)
% DraftOrFinal      = true or false, with true giving more detailed
%                     plots.
% VERBOSE           = true or false, with true giving more textual
%                     output and more plots.

PlotTimeStart=toc;
```

```
if     strcmp(DraftOrFinal,'draft')
  OverSamplingFactor=2;
elseif strcmp(DraftOrFinal,'medium')
  OverSamplingFactor=5;
elseif strcmp(DraftOrFinal,'final')
  OverSamplingFactor=10;
else
  error('Ho hum...');
end;

%% Compute plotting intervals ---------------------------------------
% For both nu and t, we plot the bandpass filtered spreading function
% in an interval that is the smallest interval I containing all
% samples extended at each endpoint with an extension of length
% PlotIntervalExtension=0;*|I|:

SetPlot(2); % Set larger fontsize and wider lines in plots

PlotIntervalExtension=0;

t_start=P(1)*TBis;
t_end=P(end)*TBis;
t_ext=PlotIntervalExtension*(t_end-t_start)/2;
t_start=TBis*floor((t_start-t_ext)/TBis);
t_end=TBis*floor((t_end+t_ext)/TBis);
t=[t_start:TBis/OverSamplingFactor:t_end];
t=[t_start:TBis:t_end]; warning('Temporary removal of time oversampling!')
NrOfSamples_t=length(t);

nu_start=N(1)*omega_0;
nu_end=N(end)*omega_0;
nu_ext=PlotIntervalExtension*(nu_end-nu_start)/2;
nu_start=omega_0*floor((nu_start-nu_ext)/omega_0);
nu_end=omega_0*floor((nu_end+nu_ext)/omega_0);
nu=[nu_start:omega_0/OverSamplingFactor:nu_end];
NrOfSamples_nu=length(nu);

if VERBOSE
  disp(['Computing and plotting spreading function on a ' ...
    int2str(NrOfSamples_t) 'x' int2str(NrOfSamples_nu) ' grid' ...
    '(oversampling factor ' int2str(OverSamplingFactor) ')...'])
end
```

```
S_H=BPFS(N,P,S_np,Omega,omega_c,omega,Omega_cBis,OmegaBis,...
                L_0,C_0,t,nu,VERBOSE);
%surf(nu,t,abs(S_H));
mesh(nu,t,abs(S_H));
%set(gca,'YTick',[t(1) P(1)*TBis P(end)*TBis t(end)]);
%set(gca,'YTickLabel',{num2str(t(1)), num2str(P(1)*TBis), ...
%                      num2str(P(end)*TBis),  num2str(t(end))});
xlabel('\nu (Hz)')
ylabel('t (s)')
zlabel('|S_{\Omega_c^{\prime\prime}}^{\Omega^{\prime\prime}}(\nu,t)|');
% title(['Spreading function |(S_h(\nu,t))| computed from ' ...
%   SprdFctCoeffType ' coefficients.']);
axis tight
PathExcludingExtension = ...
  ['Figures\' channel '\SpreadingFunction\' ...
    num2str(toc/60) 'minutes'];
multiplot(PathExcludingExtension,'landscape')
```

B.12 `resample`: Compute new samples from Nyquist frequency samples

File path: Matlab/ChannelMatrix/resample.m
Short description: Compute new samples from Nyquist frequency samples.
Additional explanation: The resampling done below is a more or less direct application of the Nyquist sampling theorem (4.5b), with the input parameters appearing as follows:

$$g(\mathtt{t}) = |\boldsymbol{T}| \sum_{\boldsymbol{m} \in \mathtt{g\_sample\_ind}} g(\boldsymbol{mT}) \mathrm{e}^{\mathrm{i}2\pi \langle \mathtt{g\_centerfreq}, \mathtt{t} - \boldsymbol{mT} \rangle} \mathrm{sinc}_{\mathtt{g\_bandwidth}}(\mathtt{t} - \boldsymbol{mT}).$$

Source code

```
function g=resample(g_centerfreq,g_bandwidth,g_sample_ind,g_samples,t)
% USAGE: g=resample(g_centerfreq,g_bandwidth,g_sample_ind,g_samples,t)
%
% Computes new samples of a function g that is completely determined
% by a finite number of input Nyquist frequency samples.
%
% INPUT
% g_centerfreq = Center frequency of g.
% g_bandwidth  = Bandwidth of g.
% g_sample_ind = Integer sample indices.
% g_samples    = Samples of g such that the mth sample
%                is sampled at "time" g_sample_ind(m)*T
%                with T = 1/g_bandwidth
% t            = Number, vector or matrix containing the point at which
%                g shall be computed.
%
% OUTPUT
% g = Number, vector or matrix of the same size as t, containing the
%     computed new samples
g = zeros(size(t));
T = 1/g_bandwidth;

NrOfInputSamples  = length(g_samples);

for m_ind=1:NrOfInputSamples
  m = g_sample_ind(m_ind);
  t_min_mT = t-m*T;
  g = g + g_samples(m_ind)*exp(i*2*pi*g_centerfreq.*t_min_mT) ...
          .*SincOmega(g_bandwidth,t_min_mT);
```

```
% % Correction for real-valued signals:
%   g = g + g_samples(m_ind)*cos(2*pi*g_centerfreq.*t_min_mT) ...
%           .*SincOmega(g_bandwidth,t_min_mT);
end
g=g*T;
```

B.13 `SetParameters`: Set system parameters for different applications

File path: Matlab/ChannelMatrix/SetParameters.m
Short description: Set system parameters for different applications.
Additional explanation: Here the system parameters are set with channel-dependent parameters chosen as described in Section 6 and other parameters chosen according to the following algorithm. The steps of this algorithm are enumerated with the same numbers as the corresponding parts of the source code.

1. Choose the Gabor system index set $\mathcal{R}$ the size of the index sets $\mathcal{Q}$, $\mathcal{N}$ and the minimum desired size of the index set $\mathcal{P}$.

2. Choose a spreading function according to the channel-dependent characteristics described in Section 6 by setting corresponding values on the Doppler spread parameters $\boldsymbol{\omega}_{\mathrm{c}}$ and $\boldsymbol{\omega}$ as well as the total bandwidth $[\boldsymbol{\Omega_0}, \boldsymbol{\Omega_1}]$ and a vector `t_exp_decay_par` containing the parameters $\begin{pmatrix} A & \epsilon & \boldsymbol{t_0} & \boldsymbol{t_1} & \boldsymbol{t_2} & \boldsymbol{t_3} \end{pmatrix}$ that describe the spreading function time decay envelope function a in (B.8) below.

3. Compute the bandwidth $\boldsymbol{\Omega}$ and sampling interval $\boldsymbol{T_g}$ of g according to (B.6) below. This is the bandwidth that we get for a Gaussian window

$$g(\boldsymbol{x}) = \mathrm{e}^{-\langle \boldsymbol{\alpha x}, \boldsymbol{x} \rangle} \qquad \text{with Fourier transform} \qquad \widehat{g}(\boldsymbol{\nu}) = \sqrt{\frac{\pi^d}{|\boldsymbol{\alpha}|}} \mathrm{e}^{-\pi^2 \left\langle \frac{\boldsymbol{\nu}}{\boldsymbol{\alpha}}, \boldsymbol{\nu} \right\rangle}. \tag{B.3}$$

and with lattice and window parameters chosen to match the shape of the spreading function support as proposed in [KM98, Koz97] (but here with an "ϵ-essential modification" (see (B.4) below) that seemed reasonable but not has been theoretically justified). We do this as follows: Let $I_{\boldsymbol{C},\boldsymbol{L}} \times I_{\boldsymbol{\omega}_{\mathrm{c}},\boldsymbol{\omega}}$ be the smallest "rectangle" in $\mathbb{R}^d \times \mathbb{R}^d$ that contains the ϵ-essential support of S_H. Similarly, let $I_{\boldsymbol{A}_{\mathrm{c}},\boldsymbol{A}} \times I_{\boldsymbol{B}_{\mathrm{c}},\boldsymbol{B}}$ be the smallest rectangle containing the ϵ-essential support of the *short-time Fourier transform* $\mathcal{V}_g g$ of g with window g (as defined in (2.10)), which for our choice of g is

$$\begin{aligned}
\mathcal{V}_g g(\boldsymbol{t}, \boldsymbol{\nu}) &= \int_{\mathbb{R}^d} \mathrm{e}^{-\langle \boldsymbol{\alpha x}, \boldsymbol{x} \rangle} \mathrm{e}^{-\langle \boldsymbol{\alpha}(\boldsymbol{x}-\boldsymbol{t}), \boldsymbol{x}-\boldsymbol{t} \rangle} \mathrm{e}^{-\mathrm{i}2\pi \langle \boldsymbol{\nu}, \boldsymbol{x}-\boldsymbol{t} \rangle} \, \mathrm{d}\boldsymbol{x} \\
&= \int_{\mathbb{R}^d} \mathrm{e}^{-2\left\langle \boldsymbol{\alpha}(\boldsymbol{x}-\frac{\boldsymbol{t}}{2}), \boldsymbol{x}-\frac{\boldsymbol{t}}{2} \right\rangle - \frac{1}{2}\langle \boldsymbol{\alpha t}, \boldsymbol{t} \rangle} \mathrm{e}^{-\mathrm{i}2\pi \langle \boldsymbol{\nu}, \boldsymbol{x}-\boldsymbol{t} \rangle} \, \mathrm{d}\boldsymbol{x} \\
&= \mathrm{e}^{\mathrm{i}2\pi \left\langle \boldsymbol{\nu}, \frac{\boldsymbol{t}}{2} \right\rangle - \frac{1}{2}\langle \boldsymbol{\alpha t}, \boldsymbol{t} \rangle} \int_{\mathbb{R}^d} \mathrm{e}^{-\langle \boldsymbol{2\alpha y}, \boldsymbol{y} \rangle} \mathrm{e}^{-\mathrm{i}2\pi \langle \boldsymbol{\nu}, \boldsymbol{y} \rangle} \, \mathrm{d}\boldsymbol{y} \\
|\mathcal{V}_g g(\boldsymbol{t}, \boldsymbol{\nu})| &= \mathrm{e}^{-\frac{1}{2}\langle \boldsymbol{\alpha t}, \boldsymbol{t} \rangle} \sqrt{\frac{\pi^d}{|2\boldsymbol{\alpha}|}} \mathrm{e}^{-\pi^2 \left\langle \frac{\boldsymbol{\nu}}{2\boldsymbol{\alpha}}, \boldsymbol{\nu} \right\rangle} = \sqrt{\frac{\pi^d}{|2\boldsymbol{\alpha}|}} \mathrm{e}^{-\frac{1}{2}\left(\langle \boldsymbol{\alpha t}, \boldsymbol{t} \rangle + \pi^2 \left\langle \frac{\boldsymbol{\nu}}{\boldsymbol{\alpha}}, \boldsymbol{\nu} \right\rangle \right)}.
\end{aligned}$$

Our "ϵ-essential" version of the design criterion in [KM98, Koz97] is

$$\frac{\boldsymbol{\omega}}{\boldsymbol{L}} = \frac{\boldsymbol{B}}{\boldsymbol{A}} = \frac{\boldsymbol{b\Omega}}{\boldsymbol{aT_g}}. \tag{B.4}$$

The boundary of the ϵ-essential support of $\mathcal{V}_g g$ is the set of all points $(\boldsymbol{t}, \boldsymbol{\nu})$ such that

$$\langle \boldsymbol{\alpha t}, \boldsymbol{t} \rangle + \pi^2 \left\langle \frac{\boldsymbol{\nu}}{\boldsymbol{\alpha}}, \boldsymbol{\nu} \right\rangle = -2\ln(\epsilon).$$

The smallest rectangular box $\frac{1}{2}[-(\boldsymbol{A}, \boldsymbol{B}), (\boldsymbol{A}, \boldsymbol{B})]$ that contains all such points $(\boldsymbol{t}, \boldsymbol{\nu})$ has sides

$$\boldsymbol{A} = \boldsymbol{\alpha}^{-\frac{1}{2}} \sqrt{-2\ln(\epsilon)} \quad \text{and} \quad \boldsymbol{B} = \frac{\boldsymbol{\alpha}^{\frac{1}{2}}}{\pi} \sqrt{-2\ln(\epsilon)}$$

Hence from (B.4) we get

$$\boldsymbol{\alpha} = \pi \frac{\boldsymbol{\omega}}{\boldsymbol{L}} \tag{B.5}$$

Similarly, if we define $I_{\boldsymbol{0,\Omega}}$ to be the smallest rectangle containing the ϵ-essential support of $\widehat{g}$, then by (B.3) $\left\langle \frac{\boldsymbol{\nu}}{\boldsymbol{\alpha}}, \boldsymbol{\nu} \right\rangle = -\frac{\ln(\epsilon)}{\pi^2}$, so that

$$\boldsymbol{\Omega} = \frac{2\boldsymbol{\alpha}^{\frac{1}{2}}}{\pi} \sqrt{-\ln(\epsilon)} \qquad \text{and} \qquad \boldsymbol{T_g} = \frac{1}{\boldsymbol{\Omega}}. \tag{B.6}$$

4. Compute the lattice constants $\boldsymbol{a}$ and $\boldsymbol{b}$ using equations (B.7) below. Recall from (5.1a) that we use Gabor frames $g_{\boldsymbol{q,r}} = T_{\boldsymbol{raT_g}} M_{\boldsymbol{qb\Omega}} g$ with lattice constants $\boldsymbol{b\Omega}$ and $\boldsymbol{aT_g}$. We have $\boldsymbol{b} \in \mathbb{R}_+^d$ but choose to require that $\boldsymbol{a} \in \mathbb{Z}_+^d$, because then any element in the frame is easily computed from only a few sample values $g(\boldsymbol{mT_g})$ as described in Theorem 2, page 22.

   Hence, given an input target sample density $\boldsymbol{D} = \boldsymbol{ab} = \boldsymbol{(aT_g)(b\Omega)}$, insertion in (B.4) gives that $\frac{\boldsymbol{\omega}}{\boldsymbol{L}} = \frac{\boldsymbol{b\Omega}}{\boldsymbol{aT_g}} = \frac{\boldsymbol{D}}{\boldsymbol{a^2 T_g^2}}$. For communications applications we want to have $\boldsymbol{D} \in \mathbb{R}_+ \setminus [\boldsymbol{0}, \boldsymbol{1}]$ (for undersampling and unique Gabor basis series expansion coefficients, see, e.g., [Grö00]). Thus we choose

$$\boldsymbol{a} = \left(\frac{\boldsymbol{DL}}{\boldsymbol{T_g^2 \omega}} \right)^{\frac{1}{2}} \quad \text{rounded off to the nearest element in } \mathbb{Z}_+^d \setminus \left[\boldsymbol{0}, \left(\frac{\boldsymbol{L}}{\boldsymbol{T_g^2 \omega}} \right)^{\frac{1}{2}} \right]. \tag{B.7a}$$

   After rounding off $\boldsymbol{a}$ we recompute $\boldsymbol{D}$ and get

$$\boldsymbol{D} = \boldsymbol{a}^2 \boldsymbol{T_g^2} \frac{\boldsymbol{\omega}}{\boldsymbol{L}} \qquad \text{and} \qquad \boldsymbol{b} = \frac{\boldsymbol{D}}{\boldsymbol{a}}. \tag{B.7b}$$

5. Check that $\mathcal{Q}$ not is too large for the Gabor system to satisfy the bandwidth requirements $[\boldsymbol{\Omega_0}, \boldsymbol{\Omega_1}]$. Set $\boldsymbol{\Omega_c} = 0$ (which is required for real-valued windows) and choose $\mathcal{R}$ so that it satisfies the bandwidth requirements $[\boldsymbol{\Omega_0}, \boldsymbol{\Omega_1}]$. Compute $\boldsymbol{\Omega'_c}$ and $\boldsymbol{\Omega'}$ using (4.15a). Set $\boldsymbol{T'} = \frac{1}{\boldsymbol{\Omega'}}$ and $\boldsymbol{\Omega''_c} = \boldsymbol{\Omega'_c}$. Choose $\boldsymbol{\Omega''}$ to be large enough to obtain a desired minimum size (`P_min_size`) of $\mathcal{P}$ and such that $\boldsymbol{\Omega''} \geq 1.5\boldsymbol{\Omega'}$. Set $\boldsymbol{T_\gamma} = \frac{1}{\boldsymbol{\Omega}+\boldsymbol{\omega}}$ and $\boldsymbol{T''} = \frac{1}{\boldsymbol{\Omega''}}$.

6. Compute the samples of the restriction of g to its ϵ-essential support (as plotted in Figure 7 (a)). We get this support with the same arguments as in step 3 by noting that $g(x) = \epsilon$ if $\langle \boldsymbol{\alpha x}, \boldsymbol{x} \rangle = -\ln(\epsilon)$.

7. Choose $\boldsymbol{L_0}$ and $\boldsymbol{C_0}$ according to (4.15b) and large enough for the number $|\mathcal{N}|$ of samples $\boldsymbol{n\omega_0}$ to be at least `N_desired_length`.

8. Compute the samples $S^{\Omega''}_{\Omega''_c}(\boldsymbol{n\omega_0}, \boldsymbol{pT''})$ that appear as coefficients in (4.16). There are basically two ways to obtain these coefficients:

   (a) Compute them from measurements on a real channel.

   (b) Generate a random choice of the samples that satisfy the decay and support properties describes in sections 3 and 6, based on the underlying assumption that then, there is always some constellation of antennas and reflecting objects that correspond to this particular spreading function.

   For the results in Section 5 we have chosen approach 8b. The exponential decay is computed as follows, where the input parameters are chosen in step 2 above.

$$a(\boldsymbol{t}) = \begin{cases} \epsilon \cdot \left(\frac{A}{\epsilon}\right)^{\|\boldsymbol{t}\|_2 - t_0} & \text{if } t_0 < \|\boldsymbol{t}\|_2 \leq t_1, \\ A & \text{if } t_1 < \|\boldsymbol{t}\|_2 \leq t_2, \\ A \cdot \left(\frac{\epsilon}{A}\right)^{\|\boldsymbol{t}\|_2 - t_2} & \text{if } t_2 < \|\boldsymbol{t}\|_2 \leq t_3, \\ 0 & \text{otherwise.} \end{cases} \tag{B.8}$$

   The subexponential decay is computed in the same way, since the difference between the exponential and subexponential decay is entirely in the negligible (and neglected) tail.

Source code

```
function [Omega_c,Omega,omega_c,omega,Omega_cBis,OmegaBis,...
          B_0,C_0,T_g,T_gamma,g,a,b,M,N,P,Q,R,S_np]=...
  SetParameters(channel,SprdFctTime,SprdFctType,sys_size,...
                P_min_size,window,D,epsilon,VERBOSE);
% USAGE:
% [Omega_c,Omega,omega_c,omega,Omega_cBis,OmegaBis,...
```

```
%                              B_0,C_0,T_g,T_gamma,g,a,b,M,N,P,Q,R,S_np] = ...
%     SetParameters(channel,SprdFctTime,SprdFctType,sys_size,...
%                   P_min_size,window,D,epsilon,VERBOSE);
%
% The following is a very brief description of the input and output
% parameters.  For a more comprehensive description, see the software
% documentation.
%
% INPUT:
% channel     = 'OFDM', 'satellite' or 'underwater'
% SprdFctTime = 'box' or 'exp'.  Tells wether output parameter
%                p_envelope shall be constant or exponential decaying
% SprdFctType = 'Kronecker delta', 'pseudo random', ...
%               'uniformly distributed random' or ...
%               'Normally distributed random'.
% sys_size        = 'small', medium or 'large'. Tells whether to do
%                quick computations for a small system, half-slow for
%                a larger system orreally big ones.
% P_min_size  = Minimum number of elements desired for the index set P.
% window      = 'Gaussian' (or all you'll get back is an error message =) )
% D           = TF-lattice unit cell area.
%                        D<1: oversampling
%                        D=1: critical sampling
%                        D>1: undersampling
% epsilon     = amplitude threshold for epsilon-essential supports.
% VERBOSE     = true or false.  Tells whether additional plots
%               and text output shall be generated.
%
% OUTPUT:
% Omega_c      = Center frequency (Hz) of synthesis Gabor window g.
% Omega        = Bandwidth (Hz) of synthesis Gabor window g.
% omega_c      = Center frequency (Hz) of spreading function eta.
% omega        = Bandwidth (Hz) of spreading function eta.
% OmegaBis     = Length of interval containing all basis
%                function frequency supports (see documentation).
%                Chosen so that the length of P is  equal to or
%                larger than the input parameter P_min_size.
% T_g          = 1/Omega. (See the software documentation.)
% T_gamma      = 1/(Omega+omega). (See the software documentation.)
% g            = Gabor window.
% a and b      : the Gabor system TF-lattice parameters are
%                a*T_g and b*Omega.
% Q            = We will consider modulations by Q(q)*b*Omega.
```

```
% R             = We will consider shifts by R(r)*a/Omega.
% M             = Vector containing all m for which g(mT_g)
%                 is nonzero.
% N             = Index set for the function n -> S_np(n,p).
% S_np          = Bandpass filtered spreading function samples.
% P             = Index set for the function p -> S_np(n,p).


%% 1: Choose Gabor System size and desired length of index set N =====
% For a transmitter Gabor wondow (OFDM pulse shape) g with bandwidth
% Omega, we will consider modulations by Q(j)*b*Omega and .
%          and shifts by R(l)*a*(sampling frequency of g or gamma).
%NlenDesired=0;%12;
NlenDesired=12;
% if     strcmp(sys_size,'small')
%   Q = [-2:2].';
%   R = [-2:2].';
% elseif strcmp(sys_size,'medium')
%   Q = [-8:7].';
%   R = [-8:7].';
% elseif strcmp(sys_size,'large')
%   Q = [-16:15].';
%   R = [-16:15].';
% end
% Set nr of frequecies/carriers (Qlen) and the
if     strcmp(sys_size,'small')
  Qlen = 16; % 0.19 minutes for OFDM
  R = [0:5].';
   if strcmp(channel,'Underwater')
     Qlen=47;
     R = [0:2].';
   end
elseif strcmp(sys_size,'medium')
  Qlen = 128; %
  if strcmp(channel,'Underwater')
    Qlen=66;
    %R = [0:16].';
  end
  R = [0:5].';
elseif strcmp(sys_size,'large')
  if strcmp(channel,'Underwater')
    %Qlen=47;
    %R = [0:20].';
```

```
    Qlen=47;
    R = [0:2].';
  else
    %Qlen = 128;
    %R = [0:15].';
    Qlen = 512;
    R = [0:2].';
  end
end
% if ( (sum( Q(:) == 0 )~=1) || (sum( R(:) == 0 )~=1))
%   warning('Both Q and R must contain zero for GaborDual to work.')
% end

%% 2: Set spreading function parameters depending on channel type ====
if     strcmp(channel(1:4),'OFDM')
  % OFDM channel with exponential time decay
  omega_c = 0;   % Hz.  Center frequency of doppler spread interval.
  omega   = 184*2; % Hz.  Length           of doppler spread interval.

  % This is, roughly, the
  % http://www.gsmworld.com/technology/spectrum/frequencies.shtml :
  Omega_0=1870e6;
  Omega_1=1990e6;
  if strcmp(SprdFctTime,'exp')
    t_exp_decay_par=[1 epsilon 1e-6*[0.8 1.3 2.5 3.5]];
    %t_exp_decay_par=[1 epsilon 1e-6*[1.5 1.5 1.7 1.7]];  % TEMP!
  elseif strcmp(SprdFctTime,'box')
    t_exp_decay_par=[1 epsilon 1e-6*[0.5 0.5 3.5 3.5]];
  else
    error([''''' SprdFctTime ...
           ''' is not a valid value of input parameter SprdFctTime.'])
  end
elseif strcmp(channel,'Satellite')
  % Example from p.47 in satellite communications book:
  omega_c = 0;    % Hz.  Center frequency of doppler spread interval.
  omega   = 18000*2; % Hz.  Length           of doppler spread interval.
  Omega_0=6e9;
  Omega_1=30e9;
  if strcmp(SprdFctTime,'exp')
    t_exp_decay_par=[1 epsilon 1e-6*[0.25 1 2.5 3.5]];
  elseif strcmp(SprdFctTime,'box')
    t_exp_decay_par=[1 epsilon 1e-6*[0.5 0.5 3.5 3.5]];
  else
```

```
    error([''''' SprdFctTime ...
           ''' is not a valid value of input parameter SprdFctTime.'])
  end
elseif strcmp(channel,'Underwater')
  % Example from p.47 in satellite communications book:
  %
  omega_c = 0;   % Hz.  Center frequency of doppler spread interval.
%   omega   = 100; % Hz.  Length           of doppler spread interval.
%   Omega_0=8000;  % Hz
%   Omega_1=11000; % Hz
  Omega_0=20000; % Hz
  Omega_1=35000; % Hz
  omega   = 30*0.51444/1531*Omega_1*2; % Hz.
                                       % Doppler spread interval length.
  if strcmp(SprdFctTime,'exp')
    %t_exp_decay_par=[1 epsilon  [0 0.4 0.6 1]];
    t_exp_decay_par=[1 epsilon  [0 0.3 0.8 1]*0.01];warning('Test...');
  elseif strcmp(SprdFctTime,'box')
    t_exp_decay_par=[1 epsilon 1e-6*[0 0 1 1]];
  else
    error([''''' SprdFctTime ...
           ''' is not a valid value of input parameter SprdFctTime.'])
  end
  warning('underwater channel parameters not carefully chosen yet')
else
  error([''''' channel
         ''' is not a valid value of input parameter ''channel''.']);
end
% Check later that the code is independent of time-shifts of
% the support (i.e. add +/- the below shift to Q' instead):
t_exp_decay_par(3:6)=t_exp_decay_par(3:6) ...
                     -(t_exp_decay_par(3)+t_exp_decay_par(6))/2;
%% 3: Compute bandwidth and sampling density for g ==================
tau=t_exp_decay_par(6)-t_exp_decay_par(3);
alpha_par=pi*omega/tau % =alpha in the documentation,
                       %  but that name is occupied
Omega = 2*sqrt(-alpha_par*log(epsilon))/pi
T_g=1/Omega;


%% 4: Compute lattice constants a and b ==============================
%  such that a is a positive integer
% We choose to round off towards zero whenever possible...
```

```
a=round(sqrt((D*tau)/(T_g^2*omega)));
if  a < sqrt((  tau)/(T_g^2*omega)) % would give oversampling
  a=a+1;
end
if a==0;
  %... and set a=1 otherwise:
  a=1;
end
D=a^2*T_g^2*omega/tau;
b=D/a;


%% 5: Compute Q, sampling period times and choose OmegaBis ===========
%  so that Tbis is small enough to fit at least P_min_size
% sample values with sampling period Tbis in an interval
% with length equal to the spreading function time support

% % Let's choose the nr of samples such that ...
% RelAmplitude=0.4;
% Qlen=floor(2*sqrt(-log(RelAmplitude)/alpha_par)/T_g+1)


Omega_c = 0;
Q1 = ceil( (Omega_0+Omega/2)/(b*Omega) );
Q = [Q1:Q1+(Qlen-1)].';
if ( Omega_1 < Omega_c+Q(end)*b*Omega -Omega/2 )
  error(['Your Gabor basis exceeds the allowed frequency band.  ' ...
         'Reducing |Q| from ' int2str(length(Q)) ' to ' ...
         int2str(floor((Omega_1-Omega_0)/(b*Omega))) ...
         ' would solve the problem.'])
end
OmegaPrime  =(length(Q)-1)*b*Omega+Omega+omega+omega_c;
Omega_cPrime=Omega_c+(Q(1)+Q(end))/2*b*Omega;
Tprime=1/OmegaPrime;
Omega_cBis=Omega_cPrime;
spr_fct_time_support_length=t_exp_decay_par(6)-t_exp_decay_par(3);
OversamplingFactor=1.5;
OmegaBis=max(OmegaPrime*OversamplingFactor,...
             ceil(P_min_size/spr_fct_time_support_length));
T_gamma=1/(Omega+omega);
Tbis=1/OmegaBis;

%% 6: Compute g and plot Gabor frame STFT supports ==================
```

```
if strcmp(window,'Gaussian')
  M_max=ceil(sqrt(-log(epsilon)/alpha_par)/T_g);
  M=[-M_max:M_max].'; % Vector containing all m for which
                      % |g(mT_g)|>=epsilon
  g=exp(-alpha_par.*((M.*T_g).^2));
  g=g./sqrt(sum(abs(g(:)).^2)*T_g); % Normalize to L^2-norm=1
  if VERBOSE
    plot_Gauss_frame(alpha_par,epsilon,Q,R,a,b,Omega_c,Omega, ...
                     g,M,channel);
  end
else
  error('Only Gaussian window implemented, so far')
end

%% 7: Choose omega_0 and B_0 ========================================
% Choose the support of t_0 -> h(t_0,t) so that it includes all
% points in time for which we possibly could be interested in
% computing H(g_{q,r})
t_start = floor(((M( 1 )+a*R( 1 ))*T_g+t_exp_decay_par(3)) ...
                /T_gamma-1)*T_gamma;
t_end   = ceil( ((M(end)+a*R(end))*T_g+t_exp_decay_par(6)) ...
                /T_gamma+1)*T_gamma;

C_0 = (t_start + t_end )/2;
B_0 =  t_end - t_start;
Nlen = ceil(max(NlenDesired,omega*B_0)); % since B_0=1/omega_0
B_0=Nlen/omega;
omega_0=1/B_0;



%% 8: Spreading function coefficients ===============================
% Compute envelope:
[p_envelope P] = ...
  spr_fct_time_decay(t_exp_decay_par,Tbis,VERBOSE,epsilon,...
                     SprdFctType,channel);
[n_envelope N] = ...
  spr_fct_nu_decay(omega_c,omega,omega_0,VERBOSE,epsilon, ...
                   SprdFctType,channel);
S_np = zeros(length(N),length(P));
for p=1:length(P)
  S_np(:,p)=p_envelope(p).*n_envelope;
```

```
end
% Normalize to L^2-norm 1
S_np=S_np/sqrt(sum(abs(S_np(:)).^2).*(omega_0*Tbis));
%surf(S_np)

if     strcmp(SprdFctType,'pseudo random');
  % Used for getting "seemingly random" coefficients
  % but in a way that can be exactly reproduced in any other
  % (for example C++) implementation.
  for P=1:length(P)
    t=p*TBis;
    S_np(:,p)=S_np(:,p).*cos(1000*10^6*N*t)+i*sin(12001*10^6*N*t);
  end
elseif strcmp(SprdFctType,'smooth');
  % Coefficients uniformly random distributedin [-1-,1]x[-i,i]:
  Plen=length(P)
  for n=1:Nlen
    for p=1:Plen
      S_np(n,p)=2*S_np(n,p).*(5+exp(i*2*pi*(n/Nlen+p/Plen)));
    end
  end
elseif strcmp(SprdFctType,'uniformly distributed random');
  % Coefficients uniformly random distributedin [-1-,1]x[-i,i]:
  S_np=2*S_np.*((rand(size(S_np))-0.5)+i*(rand(size(S_np))-0.5));
elseif strcmp(SprdFctType,'Normally distributed random');
  % Normally distributed random coefficients
  S_np=S_np.*randn(size(S_np));
else
  error('Input parameter SprdFctType has an incorrect value')
end

%% ======================================================================
%% ======================== INTERNAL FUNCTIONS ==========================
%% ======================================================================

%% Envelope function for Spr.Fct.exponential decay in t=p*Tbis =======
function [p_envelope P] = ...
  spr_fct_time_decay(t_exp_decay_par,Tbis,VERBOSE,epsilon, ...
                     SprdFctType,channel)
%  Function for computing the envelope function governing the time-decay
%  of the spreading function trigonometric polynomial coefficients:

A      =t_exp_decay_par(1);
```

```
epsilon=t_exp_decay_par(2);
t0     =t_exp_decay_par(3);
t1     =t_exp_decay_par(4);
t2     =t_exp_decay_par(5);
t3     =t_exp_decay_par(6);

Pstart=floor(t0/Tbis);
Pend  =ceil( t3/Tbis);
P=[Pstart:Pend].';
p_envelope=zeros(size(P));

for p_ind=1:length(P)
  t=P(p_ind)*Tbis;
  if      t <= t0
    p_envelope(p_ind)=0;
  elseif  t <= t1
    p_envelope(p_ind)=epsilon*(A/epsilon)^((t-t0)/(t1-t0));
  elseif  t <= t2
    p_envelope(p_ind)=A;
  elseif  t <= t3
    p_envelope(p_ind)=A*(epsilon/A)^((t-t2)/(t3-t2));
  else
    p_envelope(p_ind)=0;
  end
end
% We do not want Hf or system matrix entries to be overly small
%  (in worst case, close to eps), so we normalize to
%  coefficients of "L^1(R)-norm 1:
p_envelope = p_envelope./(Tbis*sum(abs(p_envelope)));
if VERBOSE
  figure
  stem(P*Tbis,p_envelope);
  axis tight
  xlabel('p{\cdot}T''''      (s)');
  ylabel('a(p{\cdot}T'''')');
  title(['Spreading function {\itp} with amplitude threshold ' ...
         '{\in}=' num2str(epsilon) '.'])
  PathExcludingExtension = ['Figures\' channel '\CoeffEnvelopes\p'];
  multiplot(PathExcludingExtension,'landscape')
end

%% Envelope function for Spr.Fct. exponential decay in nu=n*omega_0 ==
function [n_envelope N] = ...
```

```
  spr_fct_nu_decay(omega_c,omega,omega_0,VERBOSE,epsilon, ...
                   SprdFctType,channel)
% Here we are cheating slightly by making exponential rather
% than subexponential decay, but the difference between the two is
% all in the negligible (and here neglected) tail
Nstart=floor((omega_c-omega/2)/omega_0);
Nend  =ceil( (omega_c+omega/2)/omega_0);
N=[Nstart:Nend].';
n_envelope=zeros(size(N));

DecayingPartRelLen=0.7;
A       =1;
nu0     =omega_c-omega/2;
nu1     =omega_c-omega/2*(1-DecayingPartRelLen);
nu2     =omega_c+omega/2*(1-DecayingPartRelLen);
nu3     =omega_c+omega/2;

for n_ind=1:length(N)
  nu=N(n_ind)*omega_0;
  if      nu <= nu0
    n_envelope(n_ind)=0;
  elseif  nu <= nu1
    n_envelope(n_ind)=epsilon*(A/epsilon)^((nu-nu0)/(nu1-nu0));
  elseif  nu <= nu2
    n_envelope(n_ind)=A;
  elseif  nu <= nu3
    n_envelope(n_ind)=A*(epsilon/A)^((nu-nu2)/(nu3-nu2));
  else
    n_envelope(n_ind)=0;
  end
end
%% We do not want Hf or system matrix entries to be overly small
%  (in worst case, close to eps), so we normalize to
%  coefficients of "L^1(R)-norm 1:
n_envelope = n_envelope./(omega_0*sum(abs(n_envelope)));
if VERBOSE
  figure
  stem(N*omega_0,n_envelope);
  axis tight
  xlabel('n{\cdot}\omega_0      (s)');
  ylabel('a(p{\cdot}T'''')');
  title(['Spreading function {\nu} with amplitude threshold ' ...
         '{\in}=' num2str(epsilon) '.'])
```

```
  PathExcludingExtension = ['Figures\' channel '\CoeffEnvelopes\nu'];
  multiplot(PathExcludingExtension,'landscape')
end


%% Function for plotting TF-localization of the chosen frame =========
function plot_Gauss_frame(alpha_par,epsilon,Q,R,a,b,Omega_c,...
                          Omega,g_sampled,M,channel)
Omega_epsilon=Omega; % TMP!!!
T_g=1/Omega;
Q=Q(:).'; % Make row vector
R=R(:).'; % Make row vector

%% Plot TF-localization of the chosen Gaussian window Gabor frame:
SetPlot(2)
figure

line_colours = {'k','r','b','g','m','c'};
line_styles  = {'-',':','-.','--',':.'};
NrOfStyles=length(line_styles);
NrOfColours=length(line_colours);
dot_colours ={'k.'};
nr_of_colours=length(line_colours);
phi=linspace(0,2*pi,100);
time_axis=sqrt(-2*log(epsilon)/alpha_par);
freq_axis=sqrt(-2*alpha_par*log(epsilon))/pi;
g_ellipse_t=time_axis*cos(phi);
g_ellipse_f=freq_axis*sin(phi);
for r=R;
  for q=Q;
    line_style=[line_colours{mod(q-Q(1),NrOfColours)+1} ...
                line_styles{mod(r-R(1),NrOfStyles)+1}];
    %col_nr=mod(q+r-1,nr_of_colours)+1;
    %col_nr=mod(q+r-1,nr_of_colours)+1;
    plot(r*a*T_g,Omega_c+q*b*Omega,dot_colours{1});
    hold on;
    plot(g_ellipse_t+r*a*T_g,g_ellipse_f+Omega_c+q*b*Omega,line_style);
  end
end
hold off
% Zoom in one black ellipse
r=R(1);
q=Q(1);
```

```
axis([r*a*T_g-time_axis r*a*T_g+time_axis ...
      q*b*Omega+Omega_c-freq_axis q*b*Omega+Omega_c+freq_axis]);
xlabel('Time (s)');
ylabel('Frequency (Hz)');
% title([num2str(length(R)) 'x' num2str(length(Q)) ...
%        ' Gaussian Gabor system essential supports. ' ...
%        'Lattice unit cell area a\cdot b=' num2str(a*b) '.']);
 multiplot(['Figures/' channel '/GaborSystem/EssentialSupports'], ...
            'landscape')

%% Plot the window and its sample values
figure
SetPlot(1)
subplot(1,2,1)
N=ceil(time_axis/T_g);
Tcont=T_g/1000;
t=[-2*M(end)*T_g:Tcont:2*M(end)*T_g].';
g=resample(Omega_c,Omega,M,g_sampled,t);
g_max=max(g_sampled);
% CutToEssSupp(epsilon,OversampFact,g_centerfreq, ...
%                            g_bandwidth,g_sample_ind,g_samples)
semilogy(t,abs(g),'b-',M.*T_g,g_sampled,'ro', ...
     [t(1);t(end)],g_max*[epsilon;epsilon],'r:');
axis([t(1) t(end) 1e-13 g_max]);
legend('|g(x)|','{g}({\itmT_g})','Location','South')
%%
text(-0.27*10^(-4),10^(-3.5),'$\epsilon g(0)$','Interpreter','latex')
%title(['Transmitter window {\it g} and its nonzero Nyquist freq. '...
%       'samples. Amplitude threshold \in=' num2str(epsilon) '.'])
xlabel('t (s)');

subplot(1,2,2)
Nstep=20;
step=Omega/2/Nstep;
xi=[-1.2*Nstep:1.2*Nstep].'.*step;
Fg=zeros(size(xi));
Fgsupport=find(abs(xi)<=Omega/2);
for mind=1:length(M)
  m=M(mind);
  Fg(Fgsupport)=Fg(Fgsupport) ...
                +g_sampled(mind)*exp(-i*2*pi*xi(Fgsupport)*m*T_g);
end
Fg=real(Fg)*T_g;
```

```
gauss=sqrt(pi/alpha_par)*exp(-pi^2*xi.^2./alpha_par)*g_max;
y_min=min(gauss);
y_max=max([gauss(:);Fg(:)]);
semilogy(xi,abs(Fg),'b',xi,gauss,'r:.',...
         [xi(1);xi(end)], ...
         sqrt(pi/alpha_par)*g_max*[epsilon;epsilon],'r-.', ...
         [-Omega_epsilon/2;-Omega_epsilon/2],[y_min y_max],'b:',...
         [ Omega_epsilon/2; Omega_epsilon/2],[y_min y_max],'b:' )
xlabel('{\xi} (Hz)');
axis tight
%set('Interpreter','latex');
legend('|{\itg}^{\wedge}(\xi)|', ...
       '{\itg}_0^{\wedge}(\xi)', ...
       'Location','South')
text(-0.2*10^4,10^(-7.5),'$\epsilon\hat{g}(0)$','Interpreter','latex')
text(-1.9*10^4,10^(-11),'$-\frac{\Omega}{2}$','Interpreter','latex')
text( 1.65*10^4,10^(-11),'$\frac{\Omega}{2}$','Interpreter','latex')
%title(['Comparison of {\itg}^{\wedge}(\xi) with the ' ...
%       'approximated Gaussian.'])
multiplot(['Figures/' channel ...
           '/GaborSystem/GaborWindowAndCriticalSampleValues'], ...
          'a4HalfLandscape')
```

B.14 SetPlot: Changes of line thickness, font size etc in plots

File path: Matlab/ChannelMatrix/SetPlot.m
Short description: Changes of line thickness, font size etc in plots.
Additional explanation: Thicker lines, smaller fonts etc is good for smaller plots and for slides.

Source code

```
function SetPlot(mode);

switch (mode)

case 0
   disp('  Thin-line plots selected');
   pos = [678 36 600 400];
   font_size = 11;
   line_width = 1;

case 1
   disp('  Thick-line plots selected');
   % pos = [400 36 800 600];
   pos = [200 36 1000 700];
   font_size = 16;
   line_width = 2;

case 2
   disp('  Thick-line plots with even larger font selected');
   % pos = [400 36 800 600];
   pos = [200 36 1000 700];
   font_size = 30;
   line_width = 2;
end

disp(' ');

set(0,'defaultaxesfontsize',font_size);
set(0,'defaulttextfontsize',font_size);
set(0,'DefaultAxesLineWidth',line_width);
set(0,'DefaultlineLineWidth',line_width);
set(0,'DefaultLineMarkerSize',12);
set(0,'DefaultLineMarkerSize',18);
set(0,'DefaultLineMarkerSize',16);
```

```
set(0,'DefaultLineMarkerSize',14);


% MATLAB defines the figure Position property as a vector.
%
% [left bottom width height]

set(0,'defaultfigurepos',pos);
```

B.15 SincOmega: Computes the function $\mathrm{sinc}_{\boldsymbol{\omega}}(\boldsymbol{x})$

File path: Matlab/ChannelMatrix/SincOmega.m
Short description: Computes the function $\mathrm{sinc}_{\boldsymbol{\omega}}(\boldsymbol{x})$.
Additional explanation: Computes the function $\mathrm{sinc}_{\boldsymbol{\omega}}(\boldsymbol{x})$, which is defined before (2.1).

Source code

```
function y=SincOmega(omega,x)
% USAGE: y=SincOmega(omega,x)
%
% INPUT:
% omega = vector of length d
% x     = vector of length d
%
% OUTPUT
% y=sin(pi*omega*x)./(pi*x)

  y=omega.*sinc(omega.*x);
```

B.16 `TFshift`: Time-frequency shifts (integer time-shifts, fast)

File path: Matlab/ChannelMatrix/TFshift.m
Short description: Time-frequency shifts (integer time-shifts, fast).
Additional explanation: Compute the time-frequency shifts (4.6b) in the simplest case when $\boldsymbol{T_g} = \boldsymbol{T_f}$, that is

$$f_{\boldsymbol{q},\boldsymbol{r}}(\boldsymbol{k}\boldsymbol{T_g}) = f((\boldsymbol{k} - \boldsymbol{r}\boldsymbol{a})\boldsymbol{T_g})\mathrm{e}^{\mathrm{i}2\pi\langle \boldsymbol{k}-\boldsymbol{r}\boldsymbol{a},\boldsymbol{q}\boldsymbol{b}\rangle}.$$

Source code

```
function [TFg_samples TFg_sample_ind TFg_center_freq ] ...
          = TFshift(g_samples,g_sample_ind,g_center_freq, ...
                    g_bandwidth,t_shift_ind,nu_shift_ind)
% Applies a time-frequency shift to a function g that is completely
% determined by a finite number of Nyquist frequency input samples.
%
% USAGE: [TFg_samples TFg_sample_ind TFg_center_freq ] ...
%          = TFshift(g_samples,g_sample_ind,g_center_freq, ...
%                    g_bandwidth,t_shift_ind,nu_shift_ind)
%
% INPUT:
% g_samples     = Samples of g such that the mth sample
%                 is sampled at "time" g_sample_ind(m)*T
%                 with T = 1/g_bandwidth, where ...
% g_sample_ind  = Integer sample indices.
% g_center_freq = Center frequency of g.
% g_bandwidth   = Bandwidth of g.
% t_shift_ind   = INTEGER!
%                  The time-shift to compute is t_shift_ind*T.
% nu_shift_ind  = The freq-shift to compute is nu_shift_ind*g_bandwidth.
%
% OUTPUT
% TFg_samples     = Sample values of TF-shifted g
% TFg_sample_ind  = Corresponding new sample indices.
% TFg_center_freq = Corresponding new sample indices.
%
% SE ALSO: TFshiftAndResample for noninteger t_shift_ind.
if isintegers(t_shift_ind)
  T=1/g_bandwidth;
  TFg_sample_ind=g_sample_ind+t_shift_ind;
  TFg_samples=g_samples.*exp(i*2*pi*g_sample_ind.*nu_shift_ind);
```

```
  TFg_center_freq=g_center_freq+nu_shift_ind*g_bandwidth;
else
  error(['Fifth input ' num2str(t_shift_ind) 'is no integer.  ' ...
         'Use TFshiftAndResample instead.'])
end

%% Internal function ================================================
function b=isintegers(N)
% INPUT
% N = number, vector or matrix
%
% OUTPUT
% b = 1 if all elements in N are integers
% b = 0 otherwise.
b = min(mod(N(:),1) == 0);
```

B.17 `TFshiftAndResample`: Time-frequency shifts

File path: Matlab/ChannelMatrix/TFshiftAndResample.m
Short description: Time-frequency shifts.
Additional explanation: Compute the time-frequency shifts (4.6b) in the when possibly $\boldsymbol{T_g} \neq \boldsymbol{T_f}$, that is

$$f_{\boldsymbol{q},\boldsymbol{r}}(\boldsymbol{kT_f}) = f(\boldsymbol{kT_f} - \boldsymbol{raT_g})\mathrm{e}^{\mathrm{i}2\pi\left\langle \boldsymbol{kT_f}-\boldsymbol{raT_g},\boldsymbol{qb\Omega}\right\rangle}.$$

with the samples $f(\boldsymbol{kT_f} - \boldsymbol{raT_g})$ computed by applying (4.5b), which will be a finite sum formula for those f that we will consider.

Source code

```
function [TFg_samples TFg_sample_ind TFg_center_freq ] ...
       = TFshiftAndResample(g_samples,g_sample_ind, ...
                            g_center_freq,g_bandwidth,t_shift,nu_shift)
% USAGE:
% [TFg_samples TFg_sample_ind TFg_center_freq ] ...
%      = TFshiftAndResample(g_samples,g_sample_ind, ...
%                           g_center_freq,g_bandwidth,t_shift,nu_shift)
%
% Applies a time-frequency shift to a function g that is completely
% determined by a finite number of input Nyquist frequency samples.
%
% INPUT:
% g_samples     = Samples of g such that the mth sample
%                 is sampled at "time" g_sample_ind(m)*T
%                 with T = 1/g_bandwidth, where ...
% g_sample_ind  = Integer sample indices.
% g_center_freq = Center frequency of g.
% g_bandwidth   = Bandwidth of g.
% t_shift       = Desired time-shift
% nu_shift      = Desired frequency-shift
%
% OUTPUT
% TFg_samples     = Sample values of TF-shifted g
% TFg_sample_ind  = Corresponding new sample indices.
% TFg_center_freq = Corresponding new sample indices.
%
% SE ALSO: TFshift (does the same faster when t_shift/T is an integer).
T = 1/g_bandwidth;
TFg_sample_ind_start=g_sample_ind( 1 ) + floor(t_shift/T);
TFg_sample_ind_end  =g_sample_ind(end) + ceil( t_shift/T);
TFg_sample_ind  = [TFg_sample_ind_start:TFg_sample_ind_end].';
```

```
TFg_center_freq = g_center_freq + nu_shift;
t = TFg_sample_ind.*T;
TFg_samples     = exp(i*2*pi*nu_shift.*(t-t_shift)) ...
                  .*resample(g_center_freq,g_bandwidth,g_sample_ind, ...
                             g_samples,t-t_shift);
```

C Functions for two of the plots

This appendix contains two additional MATLAB functions, that were used to make some other plots in this report and not are related to the functions presented in Appendix B.

C.1 FourierTranformDecay: Matlab-file for producing Figure 1, p. 8

File path: Matlab/FourierTransformDecays/FourierTranformDecay.m
Short description: Matlab-file for producing Figure 1, p. 8.
Additional explanation:

Source code

```
function F = FourierTranformDecay(xi)

global xi_global cos_degree_global

tic

if nargin == 0
  xi_min=0;
  xi_max=50;
  xi = [xi_min:0.001:xi_max].';
else
  xi_min=min(xi(:));
  xi_max=max(xi(:));
end

y_min=1e-16; % = where numerical precision seem to start suffering

SetPlot(2)

%NumericalPrecision = 1e-11;  % Precision in integration
%NumericalPrecision = 1e-1;  % Precision in integration
NumericalPrecision = 5e-12;  % Precision in integration


x=[-1.5:0.01:1.5].';

f_exp=f(x);
aint=ApproxIntegral(f_exp,x);
f_exp=f_exp./aint;
```

```
[r,c] = size(xi);
F=zeros(r,c);
for n = 1:length(xi(:))
  xi_global=xi(n);
  F(n)=2*quadl(@integrand,0,1,NumericalPrecision);
end
absF=abs(F)./aint;


figure(1);
plot(x,f_exp);
axis tight
xlabel('{\itx}');
ylabel('{\itf}({\itx})');
print('-depsc2','..\..\Figures\DecayComparison\BumpFunction')

figure(2);
semilogy(xi,absF);
axis([xi_min xi_max y_min max(absF)])
xlabel('{\it\xi}');
ylabel('{}_{ \wedge}\newline|{\itf} ({\it\xi})|');
grid on
print('-depsc2', ...
  '..\..\Figures\DecayComparison\FourierTransformOfBumpFunction')

for cos_power=1:5

  cos_degree_global=cos_power;
  %% Old slow version kept for error-checking
  %
  %for n = 1:length(xi(:))
  %  xi_global=xi(n);
  %  F(n)=2*quadl(@integrand_RaisedCos,0,1,NumericalPrecision);
  %end

  fRC=f_RaisedCos(x);
  aint=ApproxIntegral(fRC,x);
  fRC=fRC./aint;

  absF = abs(Four_raised_cos(xi,cos_power))./aint;;

  figure(2+2*cos_power-1);
```

```
  plot(x,fRC);
  axis tight
  xlabel('{\itx}');
  ylabel(['{\itf}({\itx})']);
  print('-depsc2',['..\..\Figures\DecayComparison\RaisedCos' ...
                  int2str(cos_power)])

  figure(2+2*cos_power);
  semilogy(xi,absF);
  axis([xi_min xi_max y_min max(absF)])
  %axis tight
  xlabel('{\it\xi}');
  ylabel('{}_{ \wedge}\newline|{\itg} ({\it\xi})|');
  grid on
  print('-depsc2', ...
    ['..\..\Figures\DecayComparison\FourierTransformOfRaisedCos'...
     int2str(cos_power)])
end

disp(['Execution finished in ' num2str(toc/3600) 'hours, hurrah!  :-D'])

%% ------------------------------------------------------------------------
function aif=ApproxIntegral(f,x);
dx=x(2)-x(1);
aif=sum(f)*dx;

%% Candidate function 1 ---------------------------------------------------
function y = f(x)

[r,c] = size(x);
y=zeros(r,c);

pos_out_ind=find( (x>-1) & (x<1) ); % Indices giving positive output

y(pos_out_ind)=exp(-1./(1-x(pos_out_ind).^2));

%% Candidate function 2 ---------------------------------------------------
function y = f_RaisedCos(x)

global cos_degree_global

[r,c] = size(x);
y=zeros(r,c);
```

```
pos_out_ind=find( (x>-1) & (x<1) ); % Indices giving positive output

y(pos_out_ind)=(1+cos(pi*x(pos_out_ind))).^cos_degree_global;



%% Integrand 1 ------------------------------------------------------------
function y = integrand(x)
global xi_global

y=f(x).*cos(2*pi*xi_global.*x);

%% Integrand 2 ------------------------------------------------------------
function y = integrand_RaisedCos(x)
global xi_global

y=f_RaisedCos(x).*cos(2*pi*xi_global.*x);

%% Fourier transform of cos-window raised to power n ----------------------

function y = Four_raised_cos(xi,n);

n=round(n); % First make sure that we rase to an integer power

if n==0
  y=2*sinc(2*xi); % Four. tr. of "box with support in  [-1,1]";
else
  y =    Four_raised_cos(xi,    n-1) ...
      +( Four_raised_cos(xi+1/2,n-1)   ...
        +Four_raised_cos(xi-1/2,n-1))./2;
end

%% ----------------------------------------------------------------------
function SetPlot(mode);

switch (mode)

case 0
   disp('  Thin-line plots selected');
   pos = [678 36 600 400];
   font_size = 11;
   line_width = 1;
```

```
case 1
   disp('  Thick-line plots selected');
   % pos = [400 36 800 600];
   pos = [200 36 1000 700];
   font_size = 16;
   line_width = 2;

case 2
   disp('  Thick-line plots with even larger font selected');
   % pos = [400 36 800 600];
   pos = [200 36 1000 700];
   font_size = 30;
   line_width = 2;
end

disp(' ');

set(0,'defaultaxesfontsize',font_size);
set(0,'defaulttextfontsize',font_size);
set(0,'DefaultAxesLineWidth',line_width);
set(0,'DefaultlineLineWidth',line_width);
set(0,'DefaultLineMarkerSize',12);
set(0,'DefaultLineMarkerSize',18);
set(0,'DefaultLineMarkerSize',16);
set(0,'DefaultLineMarkerSize',14);


% MATLAB defines the figure Position property as a vector.
%
% [left bottom width height]

set(0,'defaultfigurepos',pos);
```

C.2 NrOfNonzeroNyquistFreqSamples: Produce Figure 6, p. 33

File path: Matlab/ChannelMatrix/NrOfNonzeroNyquistFreqSamples.m
Short description: Produce Figure 6, p. 33.
Additional explanation:

Source code

```
function NrOfNonzeroNyquistFreqSamples()
Mmax=100;
close all hidden
setplot(2)

figure(1)

M=zeros(Mmax,1);

n=[1:Mmax].';
M=ceil(2*10.^(6./(n+1)).*(n+1)/pi)+1;

semilogy(n,M);
xlabel('\itn');
ylabel('\itM')
axis tight

minM=min(M)
m=find(M==minM)
grid on

PathExcludingExtension = ...
  ['..\..\Figures\OffDiagDecay-plots\SplineM'];
multiplot(PathExcludingExtension,'landscape')
```

References

[BAB+01] Ernst Bonek, henrik Asplund, Conor Brennan, Christian Bergljung, Peter Cullen, Dirk Didascalou, Patrick C.F. Eggers, José Fernandes, Cristophe Grangeat, Ralf Heddergott, Peter Karlsson, Ralf Kattenbach, Mikael B. Knudsen, Preben E. Mogensen, Andreas F. Molisch, Bo Olsson, Jörg Pamp, Gert Frølund Pedersen, I. Pedersen, Martin Steinbauer, martin Weckerle, and Thomas Zwick. *Antennas and Propagation*, chapter 3, pages 77–306. John Wiley & Sons, 2001.

[Bel63] Philip A. Bello. Characterization of randomly time-variant linear channels. *IEEE T. Commun. Syst.*, 11:360–393, December 1963. WWW: `http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=1088793`.

[Bel64] P. Bello. Time-frequency duality. *IEEE Trans. Inform. Theory*, 10(1):18–33, 1964. WWW: `http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=1053640`.

[Beu38] A. Beurling. Sur les intégrales de Fourier absolument convergentes et leur application à une transformation fonctionnelle. In *Neuvième congrès des mathématiciens Scandinaves*, Helsingfors, 1938. Reprinted as [Beu89, p. 39–60].

[Beu89] Arne Beurling. *The Collected Works of Arne Beurling. Volume 2, Harmonic Analysis*, volume 2. Birkhäuser, Boston ; Basel, 1989.

[BLM04] Imad Barhumi, Geert Leus, and Marc Moonen. Time-domain and frequency-domain per-tone equalization for OFDM over doubly selective channels. *Signal Process.*, 84(11):2055–2066, November 2004. WWW: `http://www.sciencedirect.com/science/journal/01651684`.

[BLM05] Imad Barhumi, Geert Leus, and Marc Moonen. Time-varying FIR equalization for doubly selective channels. *IEEE T. Wirel. Commun.*, 4(1):202–214, January 2005. WWW: `http://ieeexplore.ieee.org/xpl/abs_free.jsp?arnumber=1381438`.

[BS01] Scott Beaver and Thomas Strohmer. Optimal OFDM pulse and lattice design for doubly dispersive channels. In *Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers*, pages 1763–1767, Pacific Grove, CA, USA, 2001. WWW: `http://www.math.ucdavis.edu/~strohmer`.

[Chr02] Ole Christensen. *An Introduction to Frames and Riesz Bases*. Birkhäuser, 2002.

[Con00] John B. Conway. *A course in operator theory*. 2000.

[DH98] Jacek Dziubański and Eugenio Hernández. Band-limited wavelets with subexponential decay. *Canad. Math. Bull.*, 41(4):398–403, 1998. WWW: http://www.journals.cms.math.ca/CMB/v41n4/index.en.html.

[Dom56] Y. Domar. Harmonic analysis based on certain commutative Banach algebras. *Acta Math.*, 96:1–66, 1956.

[EG04] Stefan Ericsson and Niklas Grip. Efficient wavelet prefilters with optimal time-shifts. *IEEE Trans. Signal Process.*, 53(7):2451–2461, July 2004. WWW: http://www.sm.luth.se/∼grip/Research/publications.php.

[EG05] Stefan Ericsson and Niklas Grip. An analysis method for sampling in shift-invariant spaces. *Int. J. Wavelets Multiresolut. Inf. Process.*, 3(3):301–319, September 2005. WWW: http://www.sm.luth.se/∼grip/Research/publications.php.

[FK98] Hans G. Feichtinger and Werner Kozek. Quantization of TF lattice–invariant operators on elementary LCA groups. In Hans G. Feichtinger and Thomas Strohmer, editors, *Gabor Analysis and Algorithms*, chapter 7, pages 233–266. Birkhäuser, Boston, MA, USA, 1998.

[Fol95] Gerald B Folland. *A Course in Abstract Harmonic Analysis*. Studies in advanced mathematics. CRC Press, Boca Raton, FL, 1995.

[Fol99] Gerald B. Folland. *Real analysis. Modern techniques and their applications*. Pure and Applied Mathematics. A Wiley-Interscience Series of Texts, Monographs, and Tracts., NY, second edition, 1999.

[FZ98] Hans G. Feichtinger and Georg Zimmermann. A Banach space of test functions for Gabor analysis. In Hans G. Feichtinger and Thomas Strohmer, editors, *Gabor Analysis and Algorithms*, chapter 3, pages 123–170. Birkhäuser, Boston, MA, USA, 1998.

[GHS+01] Calle Gustavsson, Christian Henriksson, Mårten Sander, Peter Sidén, Roland Standert, and Andreas Vedin. Full duplex OFDM modem over a frequency selective channel. Project course report, Department of Signals, Sensors and Systems, KTH, Stockholm, Sweden, May 2001. WWW: http://www.s3.kth.se/signal/edu/projekt/students/01/lightbrown/www/final_report.PDF.

[GP07] Niklas Grip and Götz Pfander. A discrete model for the efficient analysis of time-varying narrowband communication channels. *Multidim. Syst. Sign. Process.*, To appear:38 pages, 2007. WWW: http://www.sm.luth.se/∼grip/Research/publications.php#GrPf07a.

[Gri02] Niklas Grip. *Wavelet and Gabor Frames and Bases: Approximation, Sampling and Applications.* Doctoral thesis 2002:49, Luleå University of Technology, SE-971 87 Luleå, 2002. WWW: `http://www.sm.luth.se/~grip/Research/publications.php`.

[Grö00] Karlheinz Gröchenig. *Foundations of Time-Frequency Analysis.* Birkhäuser, 2000.

[Hör03] Lars Hörmander. *The Analysis of Linear Partial Differential Operators I; Distribution Theory and Fourier Analysis.* Classics in mathematics. Springer, second edition, 2003.

[Jan98] A. J. E. M. Janssen. The duality condition for Weyl-Heisenberg frames. In Hans G. Feichtinger and Thomas Strohmer, editors, *Gabor Analysis and Algorithms*, chapter 1, pages 33–84. Birkhäuser, Boston, MA, USA, 1998.

[JYGR74] W. C. Jakes, Y. S. Yeh, M. J. Gans, and D. O. Reudnik. Large-scale variations of the average signal. In William C. Jakes, editor, *Microwave Mobile Communications*, chapter 5, pages 309–387. John Wiley & Sons, NY, 1974. IEEE 1994 reprint with corrections.

[Kat04] Yitzhak Katznelson. *An Introduction to Harmonic Analysis.* Cambridge University Press, third corrected edition, 2004.

[KM98] Werner Kozek and Andreas F. Molisch. Nonorthogonal pulse shapes for multicarrier communications in doubly dispersive channels. *IEEE J. Sel. Area. Comm.*, 16(8):1579–1589, October 1998. WWW: `http://ieeexplore.ieee.org/iel4/49/15739/00730463.pdf`.

[Koz97] Werner Kozek. *Matched Weyl-Heisenberg Expansions of Nonstationary Environments.* Ph.D. thesis, Technical University of Vienna, March 1997. WWW: `http://www.mat.univie.ac.at/ nuhag/papers/1997/koz0897.html`.

[KP06] W. Kozek and G.E. Pfander. Identification of operators with bandlimited symbols. *SIAM J. Math. Anal.*, 37(3):867–888, 2006 2006.

[KPZ02] Werner Kozek, Götz Pfander, and Georg Zimmermann. Perturbation stability of coherent Riesz systems under convolution operators. *Appl. Comput. Harmon. Anal.*, 12(3):286–308, May 2002. WWW: `http://www.math.iu-bremen.de/pfander/publications.php`.

[LM04] Geert Leus and Marc Moonen. Equalization techniques for fading channels. In Mohamed Ibnkahla, editor, *Handbook on Signal Processing for Communications*, chapter 16, pages 16.1–16.30. CRC Press, 2004.

[LO97] W. K. Lam and R. F. Ormondroyd. A coherent COFDM modulation system for a time-varying frequency-selective underwater acoustic channel. In *Seventh International Conference on Electronic Engineering in Oceanography : technology transfer from research to industry*, pages 198–203, Southampton, UK, June 1997. IEEE. WWW: http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=612669.

[LPW05] James Lawrence, Götz Pfander, and David Walnut. Linear independence of Gabor systems in finite dimensional vector spaces. *J. Fourier Anal. Appl.*, 11(6):715–726, 2005. To appear. WWW: http://www.math.iu-bremen.de/pfander/publications.php.

[LZGk03] Geert Leus, Shengli Zhou, and Georgios B. Giannakis. Orthogonal multiple access over time- and frequency-selective channels. *IEEE Trans. Inform. Theory*, 49(8):1942–1950, August 2003. WWW: http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=1214073.

[Mat00] Gerald Matz. *A time-frequency calculus for time-varying systems and nonstationary processes with applications.* Ph.d. thesis, E 389, Vienna University of Technology, November 2000. WWW: http://www.lss.supelec.fr/perso/matz_gerald/GM_other.html.

[MB02] Gérard Maral and Michel Bousquet. *Satellite Communications Systems: Systems, Techniques and Technology.* John Wiley & Sons, fourth edition, 2002.

[MGk02] Xiaoli Ma and Georgios B. Giannakis. Maximum-diversity transmissions over time-selective wireless channels. In *Wireless Communications and Networking Conference, 2002. (WCNC2002)*, volume 1, pages 497–501, March 2002. WWW: http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=993547.

[MGk03a] Xiaoli Ma and Georgios B. Giannakis. Full-diversity full-rate complex-field space-time coding. *IEEE Trans. Signal Process.*, 51(11):2917–2930, November 2003. WWW: http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=1237423.

[MGk03b] Xiaoli Ma and Georgios B. Giannakis. Maximum-diversity transmissions over doubly selective wireless channels. *IEEE Trans. Inform. Theory*, 49(7):1832–1840, July 2003. WWW: http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=1207384.

[Mid87] D. Middleton. Channel modeling and threshold signal processing in underwater acoustics: An analytical overview. *IEEE J. Oceanic Eng.*, 12(1):4–28, January 1987. WWW: http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=1145225.

[MLGk05] Xiaoli Ma, Geert Leus, and Georgios B. Giannakis. Space-time-Doppler block coding for correlated time-selective fading channels. *IEEE Trans. Signal Process.*, 53(6):2167–2181, June 2005. WWW: http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=1433146.

[MMH+02] Gerald Matz, Andreas F. Molisch, Franz Hlawatsch, Martin Steinbauer, and Ingo Gaspard. On the systematic measurement errors of correlative mobile radio channel sounders. *IEEE T. Commun.*, 50(5):808–821, May 2002. WWW: http://www.lss.supelec.fr/perso/matz_gerald/GM_jrnl.html.

[MSG+05] G. Matz, D. Schafhuber, K. Gröchenig, M. Hartmann, and F. Hlawatsch. Analysis, optimization, and implementation of low-interference wireless multicarrier systems. *preprint (submitted to IEEE Transactions on Wireless Communications)*, 2005.

[PW06] Götz E. Pfander and David F. Walnut. Measurement of time-variant linear channels. *IEEE Trans. Inform. Theory*, 52(11):4808–4820, November 2006. WWW: http://ieeexplore.ieee.org/iel5/18/36107/01715527.pdf?isnumber=36107&arnumber=1715527.

[Rap02] Theodore S. Rappaport. *Wireless communications: principles and practice*. Prentice Hall PTR, second edition, 2002.

[Reu74] D. O. Reudnik. Large-scale variations of the average signal. In William C. Jakes, editor, *Microwave Mobile Communications*, chapter 2, pages 79–131. John Wiley & Sons, NY, 1974. IEEE 1994 reprint with corrections.

[Ric03] Scott Rickard. *Time-frequency and time-scale representations of doubly spread channels*. Ph.D. dissertation, Princeton University, November 2003.

[RS80] Michael Reed and Barry Simon. *Methods of modern mathematical physics; I: Functional analysis*. Academic Press, NY, revised and enlarged edition, 1980.

[RS00] Hans Reiter and Jan D. Stegeman. *Classical Harmonic Analysis and Locally Compact Groups*. Number 22 in London Mathematical Society monographs; New Series. Oxford University Press, second edition, 2000.

[Rud87] Walter Rudin. *Real and Complex Analysis*. McGraw-Hill, third edition, 1987.

[SA99] Akbar M. Sayeed and Behnaam Aazhang. Joint multipath-Doppler diversity in mobile wireless communications. *IEEE*

T. Commun., 47(1):123–132, January 1999. WWW: http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=00747819.

[Sto96] Milica Stojanovic. Recent advances in high-speed underwater acoustic communications. *IEEE J. Oceanic Eng.*, 21(2):125–136, April 1996. WWW: http://intl.ieeexplore.ieee.org/xpl/tocresult.jsp?isNumber=27330&puNumber=26.

[Sto99] Milica Stojanovic. *Underwater Acoustic Communications*, volume 22, pages 688–698. John Wiley & Sons, 1999. WWW: http://www.mit.edu/people/millitsa/publications.html.

[Str06] Thomas Strohmer. Pseudodifferential operators and Banach algebras in mobile communications. *Appl. Comput. Harmon. Anal.*, 20(2):237–249, March 2006. WWW: http://www.math.ucdavis.edu/∼strohmer.

[TL04] Jitendra K. Tugnait and Weilin Luo. Blind identification of time-varying channels using multistep linear predictors. *IEEE Trans. Signal Process.*, 52(6):1739–1749, June 2004. WWW: http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=01299106.

[Zad50] Lofti A. Zadeh. Frequency analysis of variable networks. *Proc. IRE*, 38(3):291–299, March 1950.

[ZK00] Y.V. Zakharov and V.P. Kodanev. Multipath-Doppler diversity of OFDM signals in an underwater acoustic channel. In *Proceedings of ICASSP'2000*, volume 5, pages 2941–2944. IEEE, June 2000. WWW: http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=861150.

[ZT02] M. Zatman and B. Tracey. Underwater acoustic mimo channel capacity. In *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1364–1368. IEEE, November 2002. WWW: http://intl.ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=1197002.

Niklas Grip received his Ph.D. degree in Mathematics from the Luleå University of Technology, in Sweden 1997, where he now has a junior research fellow position. His main scientific interests are in the fields of wavelets, Fourier and Gabor analysis with applications to signal processing and multicarrier communications.
Götz E. Pfander received his Ph.D. degree in Mathematics from the University of Maryland in 1999. Since 2002 he is an Assistant Professor of Mathematics at Jacobs University Bremen, Germany. His interests include harmonic analysis, wavelet and Gabor theory, and signal processing.