Monografías Matemáticas *García de Galdeano* **39**, 187–194 (2014)

The Continuous Wavelet Transform: Fast Implementation and Pianos

Tomas Sauer

Abstract. The (continuous) wavelet transform is a powerful tool for the analysis of signals that have to be judged according to their time–frequency content. To handle realistic instances of such signals, efficient numerical methods have to be developed that make use, for example, of the computing capabilities of modern GPUs. This paper describes the basic ideas of such computations and shows the results of some test computations.

Keywords: Wavelet transform, GPU computation, FFT, piano.
AMS classification: 65T60,65Y05.

§1. Introduction

The *continuous wavelet transform* (CWT) has been established as an important tool for the analysis of time–frequency data during the last decades, standard references still being [2] and the permanently updated and extended [5]. Given a *mother wavelet* $\psi \in L_2(\mathbb{R})$ which satisfies the *admissibility condition*

$$\int_{\mathbb{R}} |\xi|^{-1} \, |\widehat{\psi}(\xi)| \, d\xi < \infty, \tag{1.1}$$

the *wavelet transform* takes a function f to

$$W_\psi f(u,s) := \int_{\mathbb{R}} f(t) \, \frac{1}{\sqrt{|s|}} \overline{\psi\left(\frac{t-u}{s}\right)} \, dt, \qquad u, s \in \mathbb{R}, \tag{1.2}$$

where u is the *localization parameter* and s the *scale*, a value closely related to the reciprocal frequency. The normalization $|s|^{-1/2}$ takes care that the 2–norm of the *time–frequency atoms* $\psi_{s,u} := \psi\left(s^{-1}(\cdot - u)\right)$ remains invariant of s and is omitted in some definitions of the wavelet transform.

One advantage of the wavelet transform, especially in the context of the analysis of musical sound signals, lies in the fact that the so–called *Heisenberg boxes* of the time–frequency atoms take the form

$$H_\psi(u,s) := \left[u - s\sigma(\psi), u + s\sigma(\psi)\right] \times s^{-1} \left[\widehat{\mu}(\psi) - \widehat{\sigma}(\psi), \widehat{\mu}(\psi) + \widehat{\sigma}(\psi)\right], \tag{1.3}$$

where

$$\mu(f) := \frac{1}{\|f\|_2^2} \int_{\mathbb{R}} t \, |f(t)|^2 \, dt, \qquad \widehat{\mu}(f) := \frac{1}{2\pi \|f\|_2^2} \int_{\mathbb{R}} \xi \left|\widehat{f}(\xi)\right|^2 \, d\xi,$$

and

$$\sigma^2(f) := \frac{1}{\|f\|_2^2} \int_{\mathbb{R}} (t-\mu(f))^2 \, |f(t)|^2 \, dt, \qquad \widehat{\sigma}^2(f) := \frac{1}{2\pi\|f\|_2^2} \int_{\mathbb{R}} (\xi - \widehat{\mu}(f))^2 \left|\widehat{f}(\xi)\right|^2 \, d\xi,$$

are the first and second moments or mean value and variance of the function f and its Fourier transform $\widehat{f}$, respectively. The number $\mu(f)$ does not appear in (1.3) due to the simple fact that the admissibility condition (1.2) requires that $\mu(f) = 0$.

The "musical" interpretation of (1.3) is that both time and frequency are resolved with a certain *relative* accuracy, in contrast to the *absolute* accuracy provided by, for example, the Gabor transform. This is even in accordance with the musical terminology where the difference between tones is measured in *cents* which quantifies exactly the relative deviation in the associated frequencies. In addition, the larger time intervals related to low frequencies represent the fact that in order to perceive a tone its duration has to be longer than the reciprocal of the frequency, since otherwise this is not even one full period of oscillation.

This paper reports on some experiments which show that wavelet analysis helps to understand and visualize the complex acoustic behaviour of grand pianos. The computation of the wavelet transforms with a reasonable resolution of scale, i.e., resolution of frequency, within reasonable time requires a good implementation of wavelet transform. In particular, one can make use of the parallelization potential provided by modern graphics cards, the so–called *GPU computations*. It turns out that the fast wavelet transform is indeed perfectly suitable for such methods.

§2. The fast wavelet transform

A look at (1.2) shows that the wavelet transform can be written as a *convolution*

$$f * g := \int_{\mathbb{R}} f(t) \, \overline{g(\cdot - t)} \, dt,$$

namely as

$$W_\psi f(\cdot, s) = f * \psi_s, \qquad \psi_s := \frac{1}{\sqrt{|s|}} \, \psi\left(s^{-1}\cdot\right). \tag{2.1}$$

Ever since the invention of the *fast Fourier transform* (FFT) it is well–known that the Fourier transform is the most efficient way to compute any convolution as

$$f * g = \left(\widehat{f}\,\widehat{g}\right)^\vee.$$

In most applications, the function f is only given in uniformly sampled form as a vector

$$f_N := S_{N,h,t_0} f = \left[f(t_k) \;:\; k \in \mathbb{Z}_N\right], \qquad \mathbb{Z}_N := \mathbb{Z}/N\mathbb{Z} = \{0, \dots, N-1\},$$

with given *sampling distance* h, *initial time* t_0 and resulting *sampling points* $T_N := [t_k = t_0 + kh]$. The *discrete Fourier transform* of $f_N : \mathbb{Z}_N \to \mathbb{C}$,

$$\widehat{f_N} := \left[\sum_{k \in \mathbb{Z}_N} f_N(k) \, e^{2\pi i jk/N} \;:\; j \in \mathbb{Z}_N\right]$$

can be computed efficiently by means of the *fast Fourier transform* (FFT). However, it is well known, cf. [8], that the DFT $\widehat{f}$ is not a sampled version of the Fourier transform of the original function f. This relationship is built via a *quasi interpolant*

$$f_\varphi(t) := \sum_{k\in\mathbb{Z}_N} f_N(k)\,\varphi\left(h^{-1}t - k\right),$$

for which straightforward computations yield the relationship

$$\widehat{f_\varphi}(\xi_k) = h\,\widehat{\varphi}\left(\frac{2k\pi}{n}\right)\widehat{f_N}(k), \qquad \xi_k := \frac{2\pi k}{nh}, \qquad k \in \mathbb{Z}_N. \tag{2.2}$$

The function φ can be chosen for example as a centered B–spline, and the resulting smoothing effect consists of a damping of high frequencies since then $\widehat{\varphi}$ is essentially a power of the sinc function. On the other hand, in the spirit the Shannon–Whittaker Sampling Theorem, φ could be chosen as a sinc function which yields $\widehat{\varphi} = 1$ at the relevant values. However, it is well known that due to its slow decay the sinc function is not a good choice for the quasi interpolant, hence interpreting the DFT as samples of a Fourier transform at ξ_k is not a good idea from a numerical point of view, although in theory it seems an almost perfect way to go.

By means of (2.2), the Fourier transform of the wavelet transform is now easily computed for $k \in \mathbb{Z}_N$ as

$$\left(W_\psi f_\varphi(\cdot, s)\right)^\wedge(\xi_k) = \left(f_\phi * \psi_s\right)^\wedge(\xi_k) = \widehat{f_\varphi}(\xi_k)\,\widehat{\psi_s}(\xi_k) \tag{2.3}$$

$$= \sqrt{|s|}h\,\widehat{\varphi}\left(\frac{2k\pi}{n}\right)\widehat{f_N}(k)\,\widehat{\psi}(s\,\xi_k) \tag{2.4}$$

Assuming that the Fourier transform above has been generated from a data vector $w_N := \left[W_\psi f_\varphi(t_k, s) \;:\; k \in \mathbb{Z}_N\right]$, corresponding to a sampled wavelet transform, by the same quasi-interpolant, i.e., by

$$\left(\left(W_\psi f_\varphi\right)(\cdot, s)_\varphi\right)^\wedge(\xi_k) = h\,\widehat{\varphi}\left(\frac{2k\pi}{n}\right)\widehat{w_N}(k), \qquad k \in \mathbb{Z}_N,$$

we get that

$$\widehat{w_N}(k) = \sqrt{|s|}\,\widehat{f_N}(k)\,\widehat{\psi}(s\,\xi_k), \qquad k \in \mathbb{Z}_N,$$

and therefore

$$\left[W_\psi f_\varphi(t_k, s) \;:\; k \in \mathbb{Z}_N\right] = w_N = \left[\sqrt{|s|}\,\widehat{f_N}(k)\,\widehat{\psi}(s\,\xi_k) \;:\; k \in \mathbb{Z}_N\right]^\vee = \left(\widehat{f_N} \odot \widehat{\psi}_{N,s}\right)^\vee, \tag{2.5}$$

where $\odot$ denotes the componentwise product or *Hadamard product* between the vectors and

$$\widehat{\psi}_{N,s} := \left[\sqrt{|s|}\,\widehat{\psi}\left(\frac{k}{N}\frac{2\pi s}{h}\right) \;:\; k \in \mathbb{Z}_N\right].$$

The computation of (2.5) is done for a finite set $S_M := \left[s_j \;:\; j \in \mathbb{Z}_M\right]$ of scales which are best selected in an *equally tempered scale* as $s_k = s_0\sigma^k$, $k \in \mathbb{Z}_M$, for some $\sigma > 1$, as this in accordance with the shape of the Heisenberg boxes once more.

Arranging the wavelet data into a matrix

$$\widehat{\Psi}_{M,N} := \left[\sqrt{|s_j|}\,\widehat{\psi}\left(\frac{k}{N}\frac{2\pi s_j}{h}\right) \;:\; \begin{matrix} j \in \mathbb{Z}_M \\ k \in \mathbb{Z}_N \end{matrix} \right] \in \mathbb{C}^{M\times N},$$

the numerical computation of the wavelet transform can be written in the convenient form

$$\left[W_\psi f_\phi\left(t_k, s_j\right) \;:\; \begin{matrix} j \in \mathbb{Z}_M \\ k \in \mathbb{Z}_N \end{matrix} \right] = \left(\left(1_M \otimes \widehat{f_N}^T\right) \odot \Psi_{M,N}\right)^\vee . \tag{2.6}$$

Even if this appears to be a trivial reformulation (and actually is one), it is the key to developing a fast and efficient way to compute the wavelet transform in an efficient way.

The first step of such an algorithm consists, if needed, of the (pre)computation of the matrix $\Psi_{M,N}$, which is either done by simply sampling an explicitly given Fourier transform of Ψ, like in the case of the *Morlet wavelet*, or by computing M arrays of FFTs. The latter requires a resampling of the function ψ for each scale which leads to a $O(MN)$ complexity on CPUs but since the samplings are essentially independent, this job could at least be parallelized nicely on GPUs. Either way, as long as the wavelet and the scales remain fixed, the matrix $\Psi_{M,N}$ needs to be computed only once for any number of wavelet transforms of different functions f.

The next and crucial step computes the FFT of the samples, $\widehat{f_N}$, and then spreads them over Ψ by generating a matrix

$$\widehat{W} := \left[\left(\widehat{f_N}\right)_k \left(\Psi_{M,N}\right)_{jk} \;:\; \begin{matrix} j \in \mathbb{Z}_M \\ k \in \mathbb{Z}_N \end{matrix} \right] = \begin{bmatrix} \left(\widehat{f_N}\right)_1 \left(\Psi_{M,N}\right)_{11} & \cdots & \left(\widehat{f_N}\right)_N \left(\Psi_{M,N}\right)_{1N} \\ \vdots & \ddots & \vdots \\ \left(\widehat{f_N}\right)_1 \left(\Psi_{M,N}\right)_{M1} & \cdots & \left(\widehat{f_N}\right)_N \left(\Psi_{M,N}\right)_{MN} \end{bmatrix} \tag{2.7}$$

whose inverse Fourier transform yields the sampled wavelet transform in the final step. Here, it is absolutely necessary to organize the data in the sequential order $\widehat{f_N} \odot \left(\Psi_{M,N}\right)_{j,:}$, $j \in \mathbb{Z}_M$, of the scales since all professional implementations of the FFT, like `FFTW` and `CUFFT` provide a special mode for the fast computations of M identical transformations of size N as long as they are aligned in the aforementioned way.

§3. GPU implementation

The GPU implementation is using the streaming parallelization capabilities of modern graphics processing units, aka *GPU*s. As described in [1], this a simple way of parallelization which consists of *blocks* and *threads*. Essentially, a block is processor core on the GPU and a thread is one job running on this core. Threads on the same block can share memory, while jobs on different blocks are performed completely independently of each other; for example it is not possible to prescribe any order of different blocks or to have any communication between different blocks.

A GPU implementation starts with transferring copies of the data $\widehat{f_N}$ and $\Psi_{M,N}$ to the GPU memory and then the computation of $\widehat{W}$ from (2.7) can either use j as block index and k as a thread index or vice versa. This means that either the columns or the rows of (2.7) are computed as different threads on the same core. As an example, he kernel versions of the latter approach take the (much simplified) `C`–form

```
fwt<<<<M,N>>>>( f,Psi,M,N );

__global__ void
fwt1( double complex *f, double complex *Psi, unsigned M, unsigned N )
{
  Psi[ N*blockIdx.x + threadIdx.x ] *= f ( blockIdx.x );
}
```

Both approaches make sense:

1. If the thread index varies over the scale k, the number to be multiplied, $\left(\widehat{f_N}\right)_j$, can be kept in the shared memory of the block which allows for faster access to this number.
2. If the block index varies over the scale, then all scales are computed independently and the time behaviour can be expected to be almost independent of the number of scales to be computed.

It is not yet fully clear which of the two approaches is to be preferred and what can be gained from more elaborate memory access techniques, but a first test implementation verified the claim for the second approach. In fact, the increase in the number of scales even resulted in a memory overflow before showing any increase in runtime. Finally, after the multiplication which overwrites $\Psi_{M,N}$, the modified data is transferred back to GPU memory from where the inverse Fourier transform based on `CUFFT`, a fast parallelized version of the FFT with a syntax very similar to `FFTW`, is computed.

Of course, this is only a much simplified description of the GPU computation of the fast wavelet transform (FWT), in a real implementation there are a lot of subtle programming issues to be dealt with, which cannot be discussed here.

§4. Wavelet design

One of the big advantages of the continuous wavelet transform is that there are very few restrictions to the wavelet function, or *mother wavelet*, ψ which has to satisfy only the admissibility condition (1.1). This allows for the adaption of the wavelet to a specific problem or to a specific shape to be recognized in a given signal by means of correlation. And even if there is a big zoo of wavelets like the ubiquitous *Morlet wavelet*, the *Mexican hat* and many more, there is a simple way to create even more general and problem adapted wavelets. The approach is based on the following observation.

Lemma 1. *If ψ is admissible function, then so is*

$$\theta = \sum_{k\in\mathbb{Z}_P} a_k \psi\left(\cdot - t_k\right), \qquad t_k \in \mathbb{R},\ a_k \in \mathbb{C}, \quad k \in \mathbb{Z}_P,$$

for any $P > 0$.

Proof. Since

$$\widehat{\theta}(\xi) = \widehat{\psi}(\xi) \sum_{k\in\mathbb{Z}_P} a_k\, e^{it_k\xi} =: \widehat{\psi}(\xi)\, a(\xi)$$

with $a(\xi)$ uniformly continuous and uniformly bounded on $\mathbb{R}$, it follows that

$$\int_{\mathbb{R}} |\xi|^{-1} \, |\widehat{\theta}(\xi)| \, d\xi \le \|a\|_\infty \int_{\mathbb{R}} |\xi|^{-1} \, |\widehat{\psi}(\xi)| \, d\xi < \infty,$$

hence θ is admissible. □

Usually, the coefficients a_k can be found by matching θ discretely against a template g in the least squares sense,

$$\min_{a_k} \sum_{j \in \mathbb{Z}_Q} \left| g(x_j) - \theta(x_j) \right|^2, \qquad x_j \in \mathbb{R},$$

whereas the shifts t_j can be be obtained by a greedy dictionary search over a large number of fine samples like in the Karhunen–Loeve method in [5]. Such an approach was used in [3] to construct a "spindle like" wavelet that relates to typical activity patterns in EEG measurements, cf. [6].

Such adapted wavelets fit very well into the concept of the fast wavelet transform since

$$\Theta_{M,N} = \Psi_{M,N} \odot \left[\widehat{a}\left(\frac{k}{N} \frac{2\pi s_j}{h} \right) \; : \; \begin{matrix} j \in \mathbb{Z}_M \\ k \in \mathbb{Z}_N \end{matrix} \right],$$

so that all our preceding remarks about efficient computations immediately carry over to adapted wavelets.

§5. Pianos

A particularly nice application of time–frequency methods is the analysis of the sound of musical instruments. In contrast to the basic theory, almost no real instrument produces a tone which does not vary over time and thus can be understood by means of a Fourier transform. Pianos are a particular example of such a situation as the tone is generated by an overlay of the strings' original sound, and resonance of the body of the instrument as well as of other strings. Moreover, the partials of the tone decay with different speed, hence the tone even changes its "quality" or "character" over time. Moreover, due to the high stress of the strings especially in grand pianos, even the partials of a single string show a highly nonlinear behaviour and are not just multiples of the the base tone, cf. [4]. For such an analysis the Morlet wavelet as a modulated Gaussian is particularly useful as its shape is already close to the relevant signal components. The data used in the following experiments were recorded as `wav` files in 44kHz CD quality with a ZOOM H2 field recorder located in a central position below the piano. All computations were performed in practically real time, i.e., in about 5s of user time, on a Dell 7500 workstation with an NVIDIA Quadro GPU.

Fig. 1 shows the Morlet scalogram distribution of the main partial tones of a low A played for five seconds on a Bösendorfer 200 grand piano. The numbers just indicate the indices of the respective time and scale where larger scale indices correspond to lower frequencies. It can be seen that the second partial tone is more dominant than the base tone, the first partial, but that all higher partials decay more rapidly than the first one. This time–frequency behaviour gives the impression of a mellow, low tone with a clear accent in the beginning.

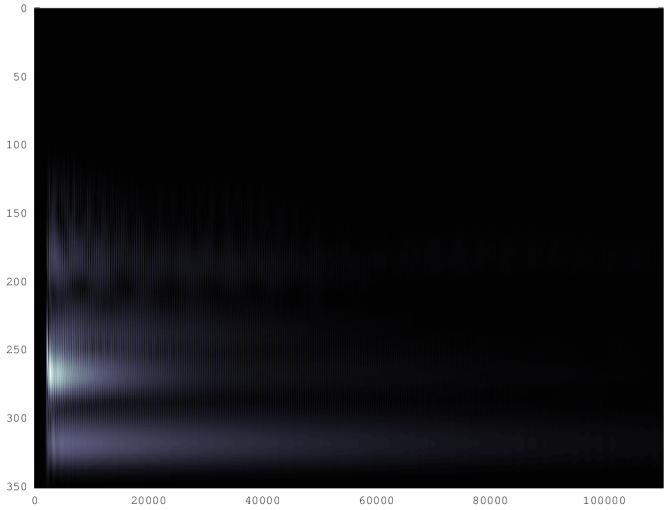

Figure 1: Low A on a grand piano.

A careful look a the y–range between 150 and 200 also shows a periodic change of the amplitude of that partial, a phenomenon known as *beats*, cf. [4, 7]. Such phenomena are particularly well reproduced by the continuous wavelet transform while Fourier and Gabor transform usually detect them as different differences, see [7].

The wavelet scalogram of different pianos, depicting the respective partials and their temporal behaviour, can also be used to distinguish between the pianos. Fig. 2 shows the low a on four different grand pianos, namely (from left to right) a Steingräber 168 S and Steingräber 192 N on the top row and our "good old" Bösendorfer 200 sharing the bottom row with a Kawai RX–5. It is not accidental that the "cheapest" among those pianos has the weakest contribution of the first partial, as this partial tone requires the highest effort in constructions.

Obviously, the sound characteristics of the respective instruments, even for single tones, are well reflected in the wavelet scalogram which makes wavelets a useful tool also for the making and testing of instruments; however, in this case also more complicated structures like chords and short pieces of melody have to be considered.

Acknowledgements

I want to thank Johannes Jurgovsky for his assistance in realizing the CUDA implementation of the FCWT and Carlos Mora from the "Pianohaus Mora" in Passau for providing the sound samples of the pianos and giving me an introduction to the stunning physics of this fascinating instrument.

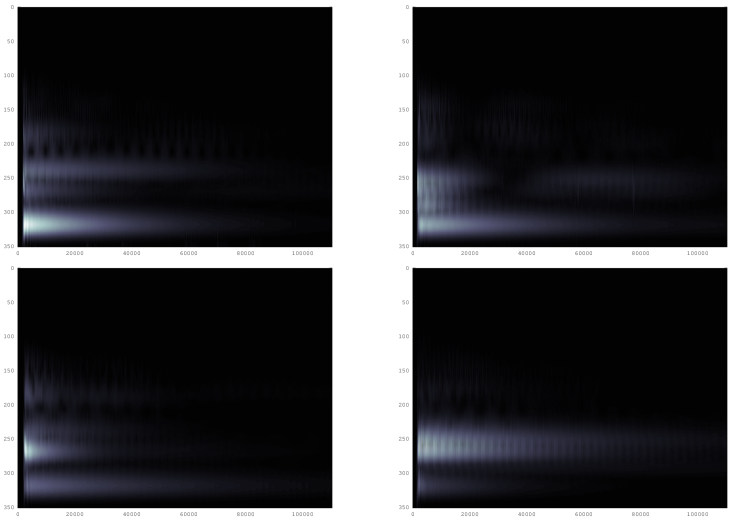

Figure 2: Four different pianos and their wavelet "fingerprint". The two pianos in the upper row are from the same maker.

References

[1] Corporation, N. *NVIDIA CUDA C Programming Guide 4.2*, 2012.

[2] Daubechies, I. *Ten Lectures on Wavelets*. SIAM, 1992.

[3] Flüggen, S. A wavelet approach to monitor brain activities during mental computations. Master's thesis, University of Gießen, 2013.

[4] Helmholtz, H. *On the Sensations of Tone*. Longmans & Co, 1885.

[5] Mallat, S. *A wavelet tour of signalprocessing*. Academic Press, 2008.

[6] Samar, V. J., Bopardikar, A., Raghuveer, M. K., and Swartz, K. Wavelet analysis of neuroelectric waveforms: A conceptual tutorial. *Brain and Language 66* (1999), 7–60.

[7] Sauer, T. Time–frequency analysis, wavelets and why things (can) go wrong. *Human Congnitive Neurophysiology 4* (2011), 38–64.

[8] Schüssler, H. W. *Digitale Signalverarbeitung*. Springer, 1992.

Tomas Sauer
Lehrstuhl für Mathematik mit Schwerpunkt Digitale Bildverarbeitung
Universität Passau
Innstr. 43, D–94032 Passau, Germany
Tomas.Sauer@uni-passau.de