Audio Denoising Using Image Processing Methods

Jan Hendrik Dithmar
dithmar@mia.uni-saarland.de

Outline:

1. Motivation
2. Wav-File Structure
3. Denoising Method, Method Noise
4. NL-means Algorithm
5. Ideas, Inpainting
6. Problems / Questions
7. Applications

Motivation

- We have a noisy audio signal and want to denoise it.
- How does, e.g. Gaussian noise look like in images?
- How does noise **sound**?

Left: Original image. **Right:** Noisy version with Gaussian noise ($\sigma = 80$).

Wav-File Structure

- A Wav-file is a container for audio data.
- Invented by Microsoft and IBM.
- It has no unique structure. For example it can have a lot of chunks.
- PCM is uncompressed audio. ADPCM and MP3 are examples for compressed audio.

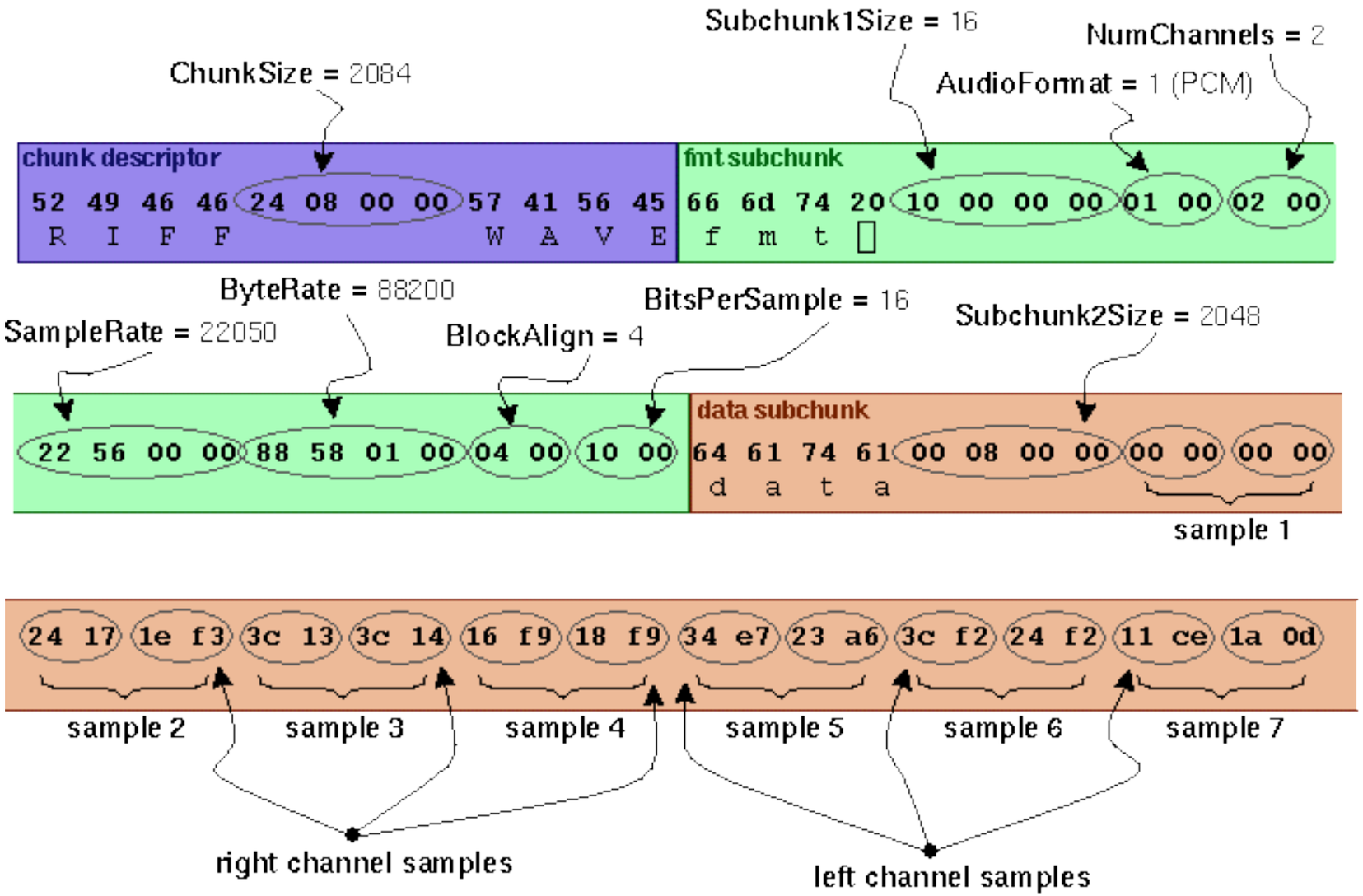
I will concentrate on the format specified on the next slide.

Wav-File Structure

Position (in bytes)	Field Name	Field Size (in bytes)	Description
0	Chunk ID	4	"RIFF"
4	File Size	4	Entire File Size - 8
8	File Format	4	"WAVE"
12	SubChunk1 ID	4	"fmt " (header signature)
16	SubChunk1 Size	4	For PCM files, this will be 16.
20	Audio Format	2	Should be 1 for uncompressed audio.
22	Number of Channels	2	Could be 1 = mono, 2 = stereo, ...
24	Sample Rate	4	44100 Hz for CD quality
28	Byte Rate	4	Sample Rate $\times$ Block Alignment
32	Block Alignment	2	Channels $\times$ Bits Per Sample $\div$ 8
34	Bits Per Sample	2	8, 16 or 24
36	SubChunk2 ID	4	"data" (header signature)
40	Data Size	4	Number of bytes following the header.
DATA			

Adapted from: `http://de.wikipedia.org/wiki/Wav`

Overview



First 72 bytes of an example WAV file.
Source: `http://ccrma.stanford.edu/courses/422/projects/WaveFormat/`

Denoising Method

- Formally, a denoising method D_h is defined as a decomposition

$$v = D_h v + n(D_h, v)$$

  where v is the noisy image, h is the filtering parameter which usually depends on the standard deviation of the noise.

- $D_h v$ is smoother than v.

- $n(D_h, v)$ looks like the realisation of white noise.

We know how a smooth image looks like. But how does a smooth audio signal sound? Tinny (blechern)? Flannelly (dumpf)?

Method Noise

- Idea:
  Taking the difference of the original image and its denoised version

$$u - D_h u$$

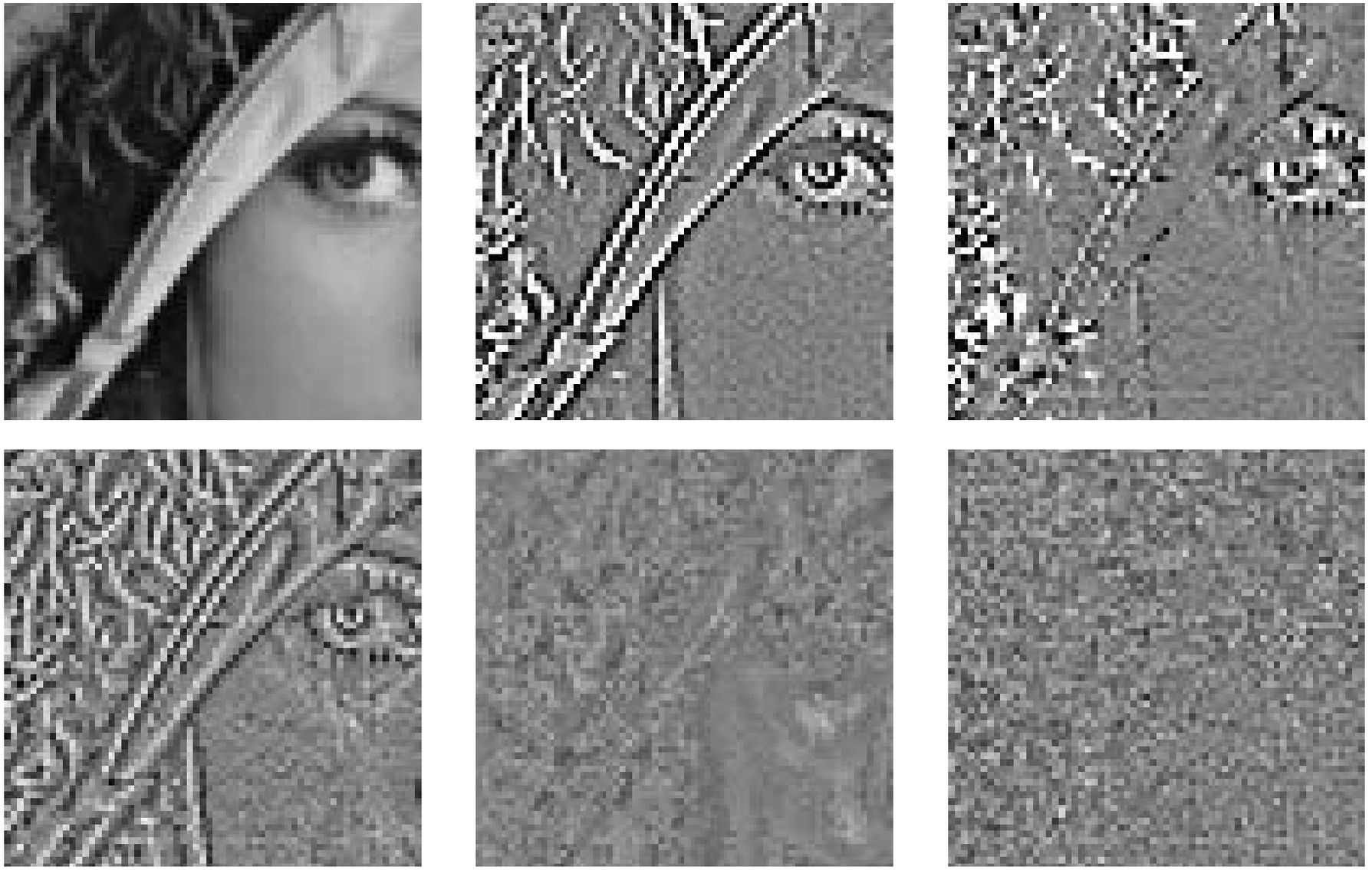
  where u is an image and D_h is a denoising operator with filtering parameter h should only give you noise.

- Problem:
  In general, you get structures of the image, which give you a good measurement whether your denoising destroys too much of the original data.

- Goal:
  Port that to audio signals since it is a good quality measurement for a filter.

- Open:
  How does noise sound? Intuitively, it should not contain any structures like rhythm, for example.

Method Noise



Part of the image Lena. **Top left:** Original image. **Others:** Displaying the image difference $u - D_h u$ after applying different filtering methods. h is fixed in order to remove noise of standard deviation 2.5. **Authors:** A. Buades, B. Coll, J.-M. Morel.

NL-means Algorithm

- Given a discrete noisy image $v = \{v(i) \,|\, i \in I\}$, the estimated value $NL[v](i)$ for a pixel i is computed as a weighted average of all the pixels in the image

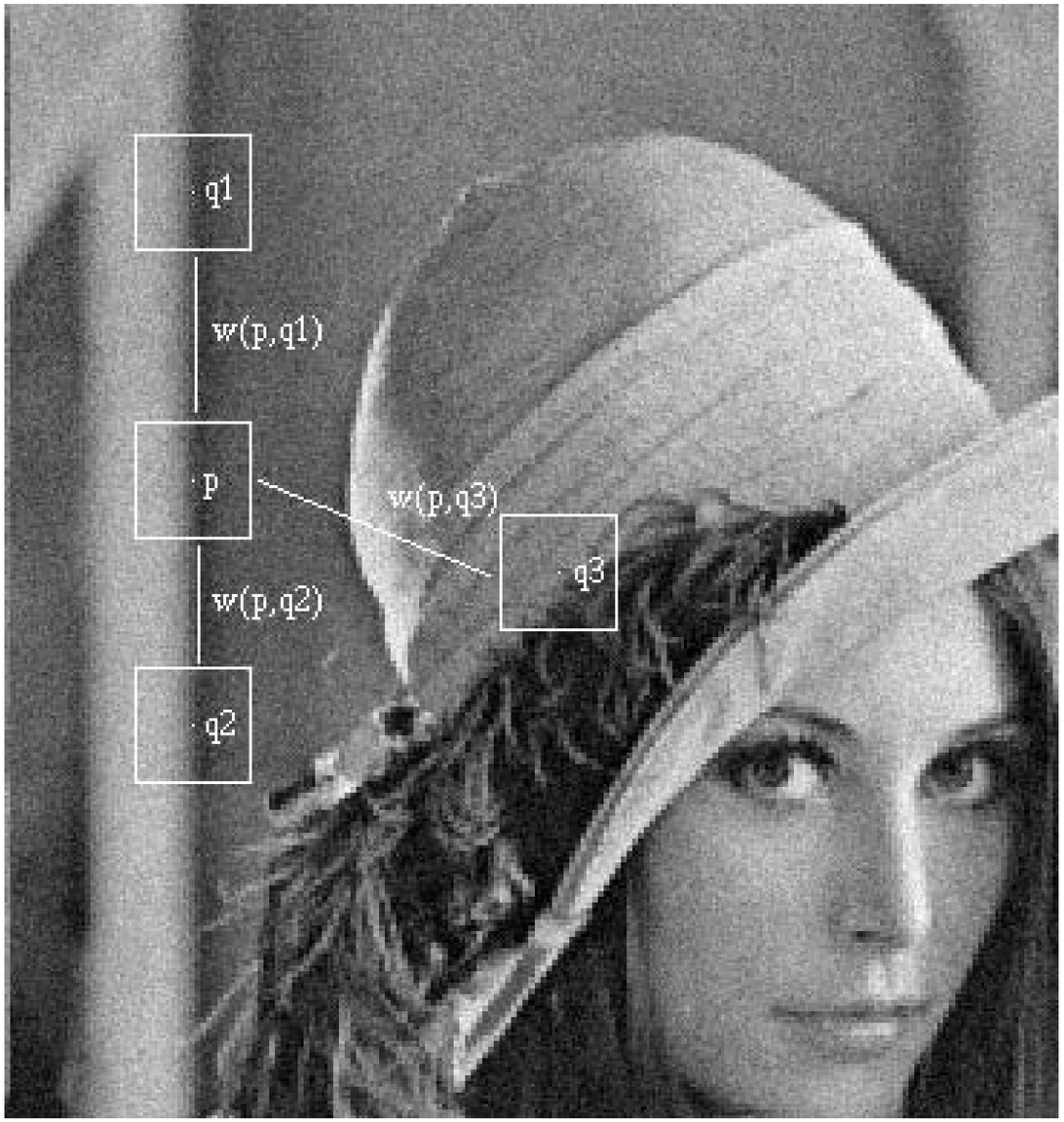
$$NL[v](i) = \sum_{j \in I} w(i,j) v(j).$$

- The family of weights $\{w(i,j)\}_j$ depends on the similarity between the pixels i and j. It satisfies the conditions $0 \leq w(i,j) \leq 1$ and $\sum_j w(i,j) = 1$.

- The similarity between two pixels i and j depends on the similarity of the intensity grey level vectors $v(\mathcal{N}_i)$ and $v(\mathcal{N}_j)$, where $\mathcal{N}_k$ denotes a square neighbourhood of fixed size and centered at a pixel k.

- The similarity is measured as a decreasing function of the weighted Euclidean distance $||v(\mathcal{N}_i) - v(\mathcal{N}_j)||^2_{2,a}, a > 0$.

NL-means Algorithm



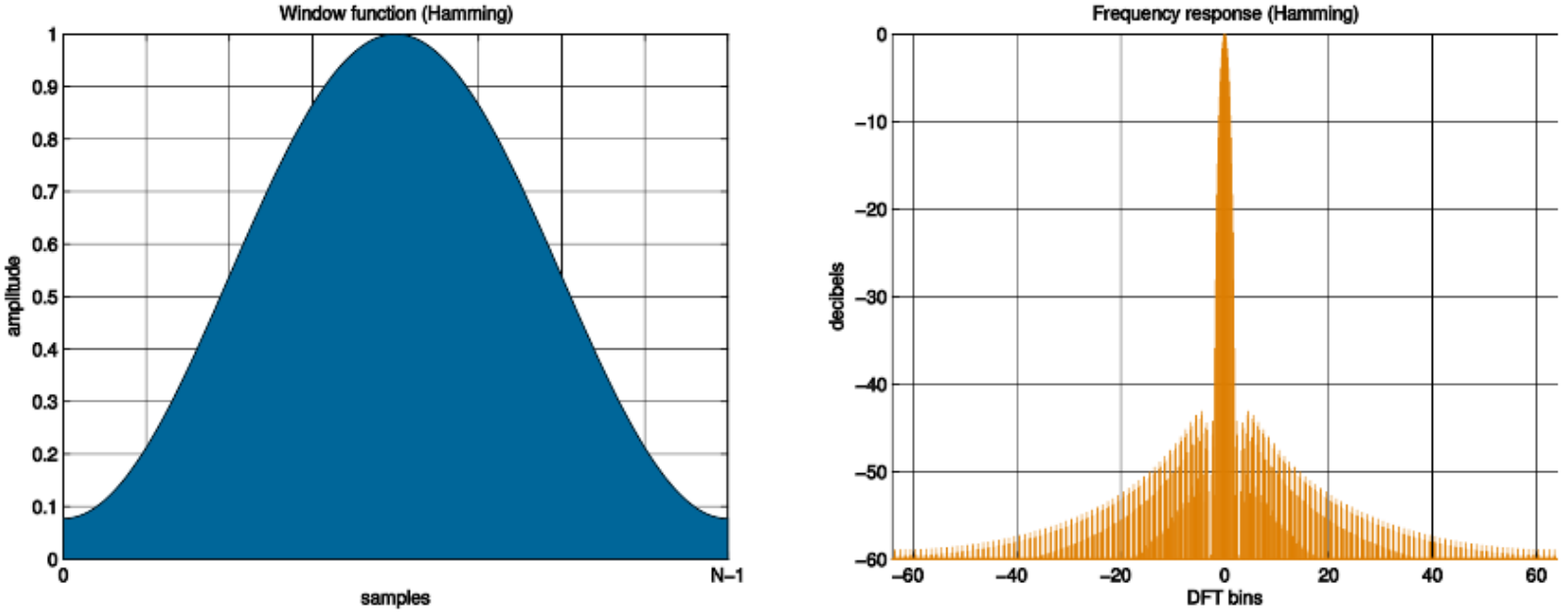
Scheme of NL-means strategy. Similar pixel neighbourhoods give a large weight, $w(p, q_1)$ and $w(p, q_2)$, while much different neighbourhoods give a small weight $w(p, q_3)$. **Authors:** A. Buades, B. Coll, J.-M. Morel.

Ideas

- Frequency analysis using the Short-Time-Fourier-Transform (STFT) with a Hamming window

$$w(n) = 0.53836 - 0.46164 \cos\left(\frac{2\pi n}{N-1}\right)$$

where N is the width of a discrete-time window function in samples (typically a power of 2) and n an integer value with $0 \leq n \leq N-1$.



Hamming window. Source: `http://en.wikipedia.org/wiki/Hamming_window`

Ideas

- Apply different filtering methods, for example
  - Gaussian filtering,
  - anisotropic filtering,
  - total variation minimisation,
  - neighbourhood filtering.
- Synthesize audio information by using the neighbourhood.
- Calculate the method noise of the different filtering methods to judge whether a method is good or not. But this would only work if the intuition about the method noise is true.

Inpainting

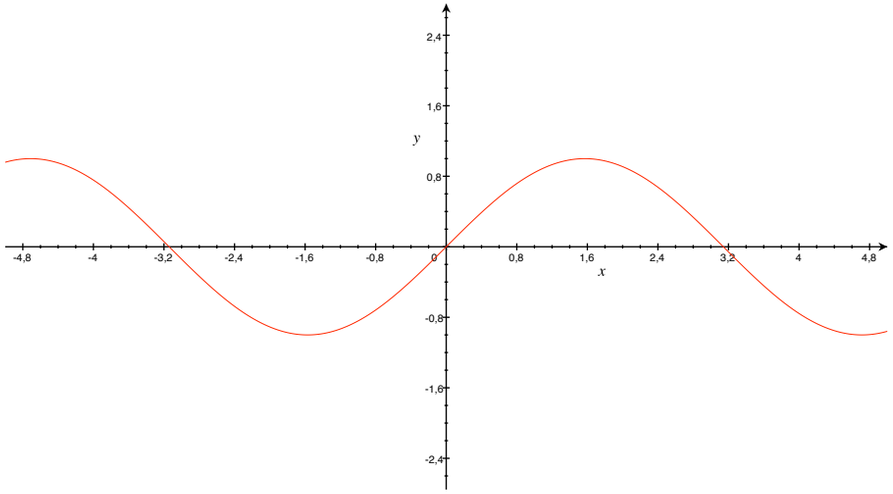
- First step:
  Try to find parts of the sound file that fit in the hole to be filled.

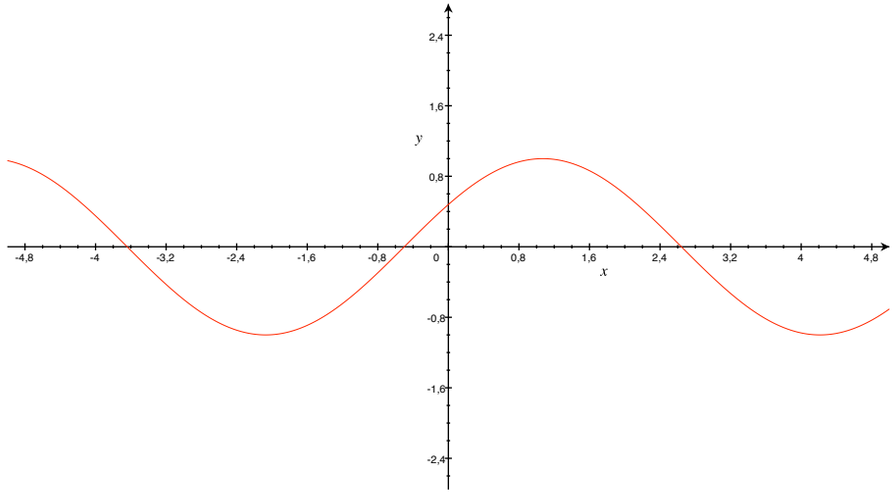
- Problem:
  You hear a digital click if the sound file is not cut properly. In general, this results in discontinuities in the audio signal.
  This means, one can not fill the hole by simply plugging in a sound file which has the same size. Care must be taken at the boundaries!

- Remedy:
  Use another line/channel where you can fade in the block to be filled-in and fade out afterwards.

Problems / Questions

- How does a smooth signal sound?
- How does noise sound?
- Not everything that sounds the same is actually the same.
  $\longrightarrow$ need a suitable threshold!





- What does averaging do with the signal? What is removed? What is introduced? How does the signal sound afterwards?
- How much audio data can one synthesize based on the neighbourhood?
- Background sound has definitely to be kept. Removing that results in an unnatural listening experience!

Applications

- Remove local distortions like a sneeze (Niesen) or a cough (Husten).
- Remove distortions caused by the playing technique of records (Schallplatten).
- Automatic cutting of music.

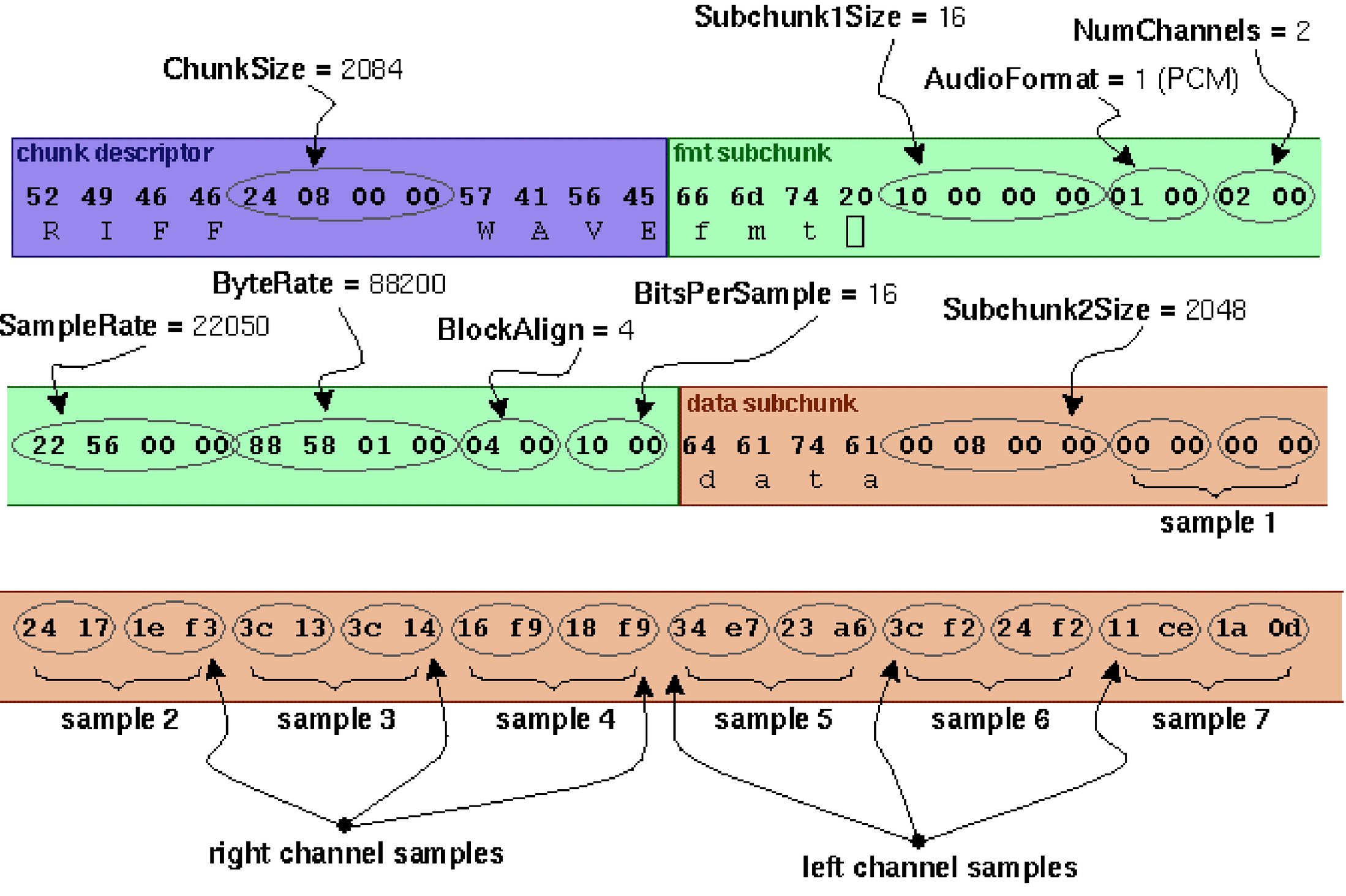
Summary

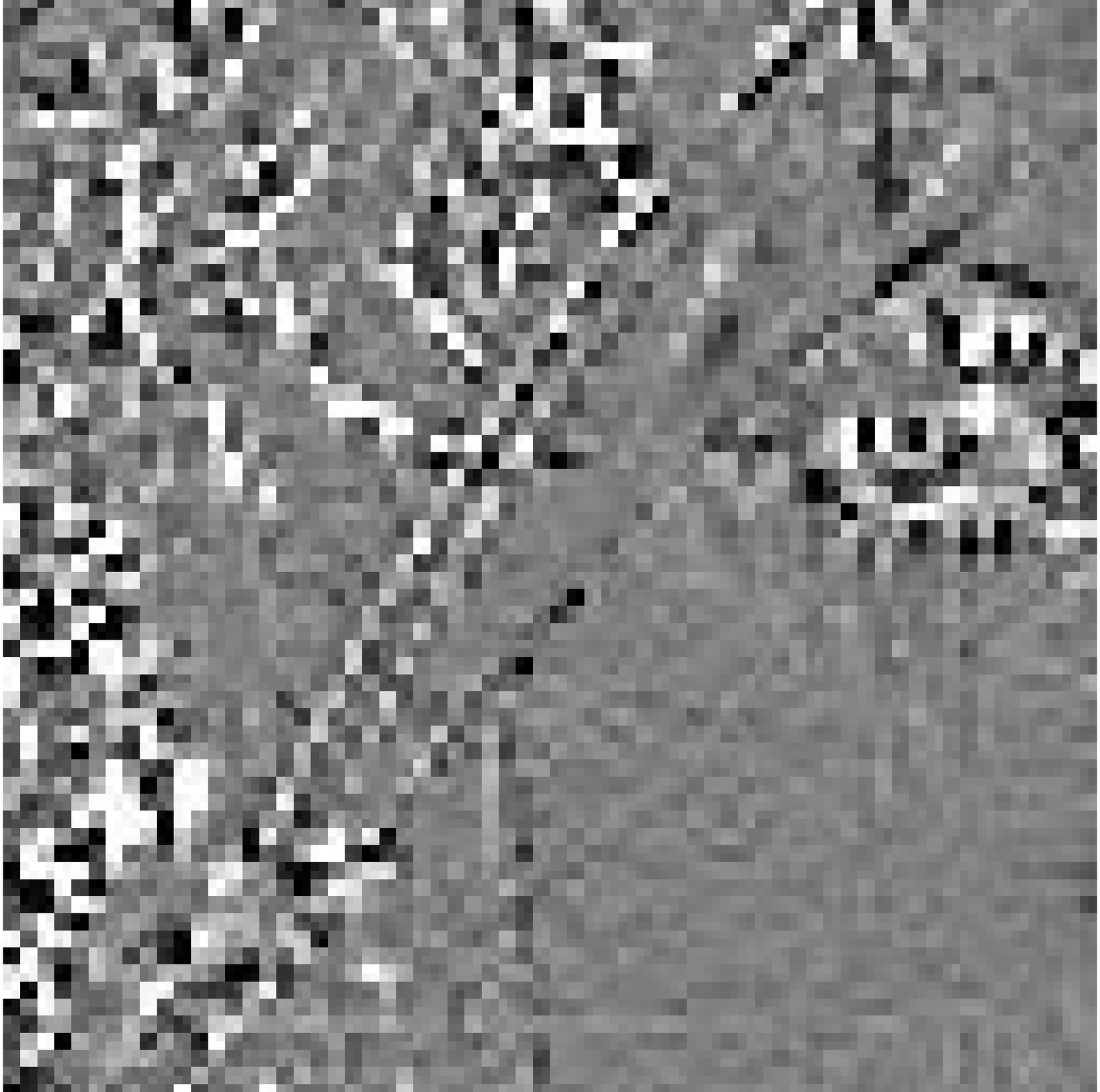
- It looks in theory that the ideas of the NL-means algorithm work also for audio signals.
- There are a lot of open questions especially when it comes to smoothing or replacing parts of the signal.
- Some things only work because music is highly redundant.

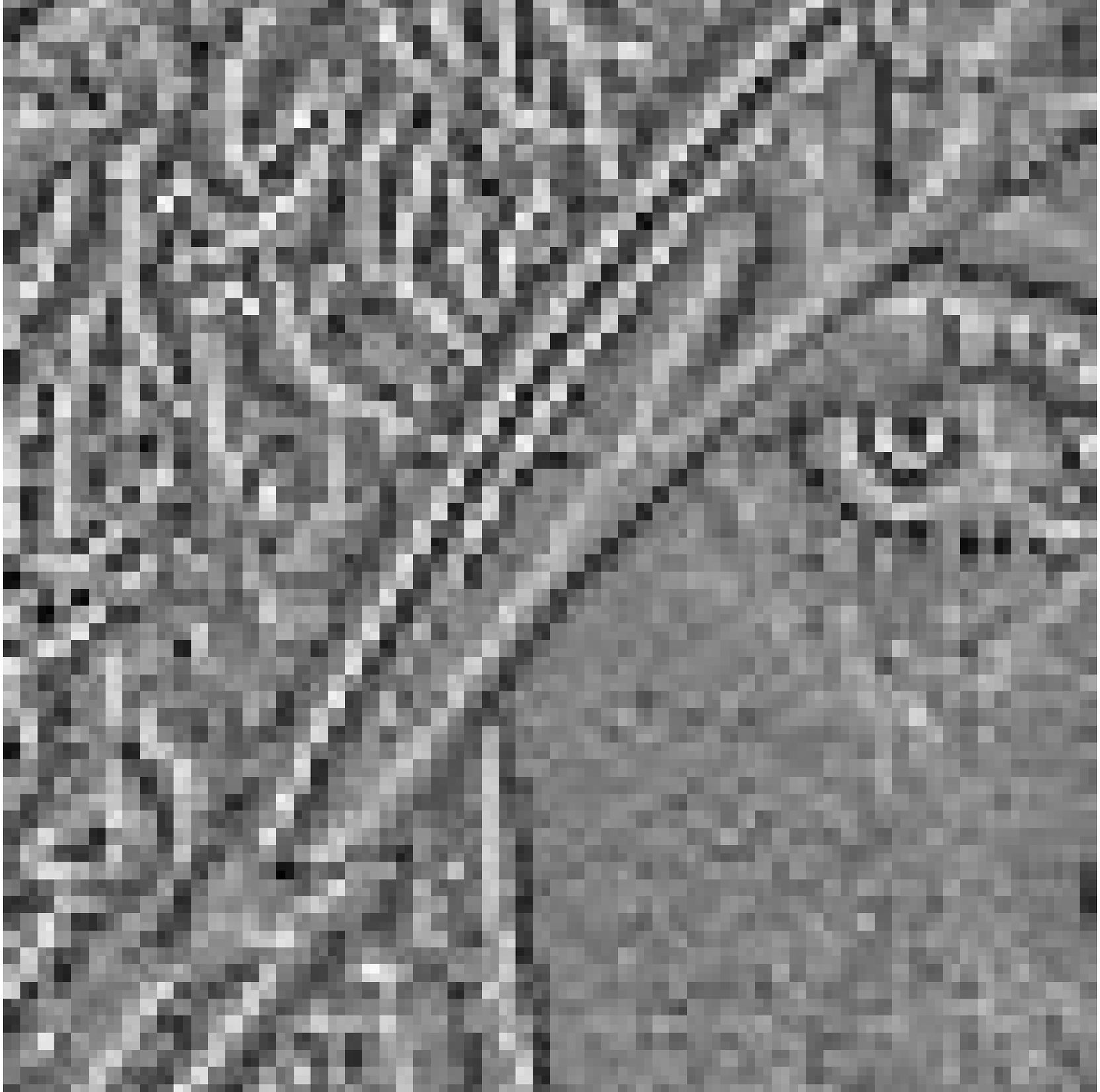
References

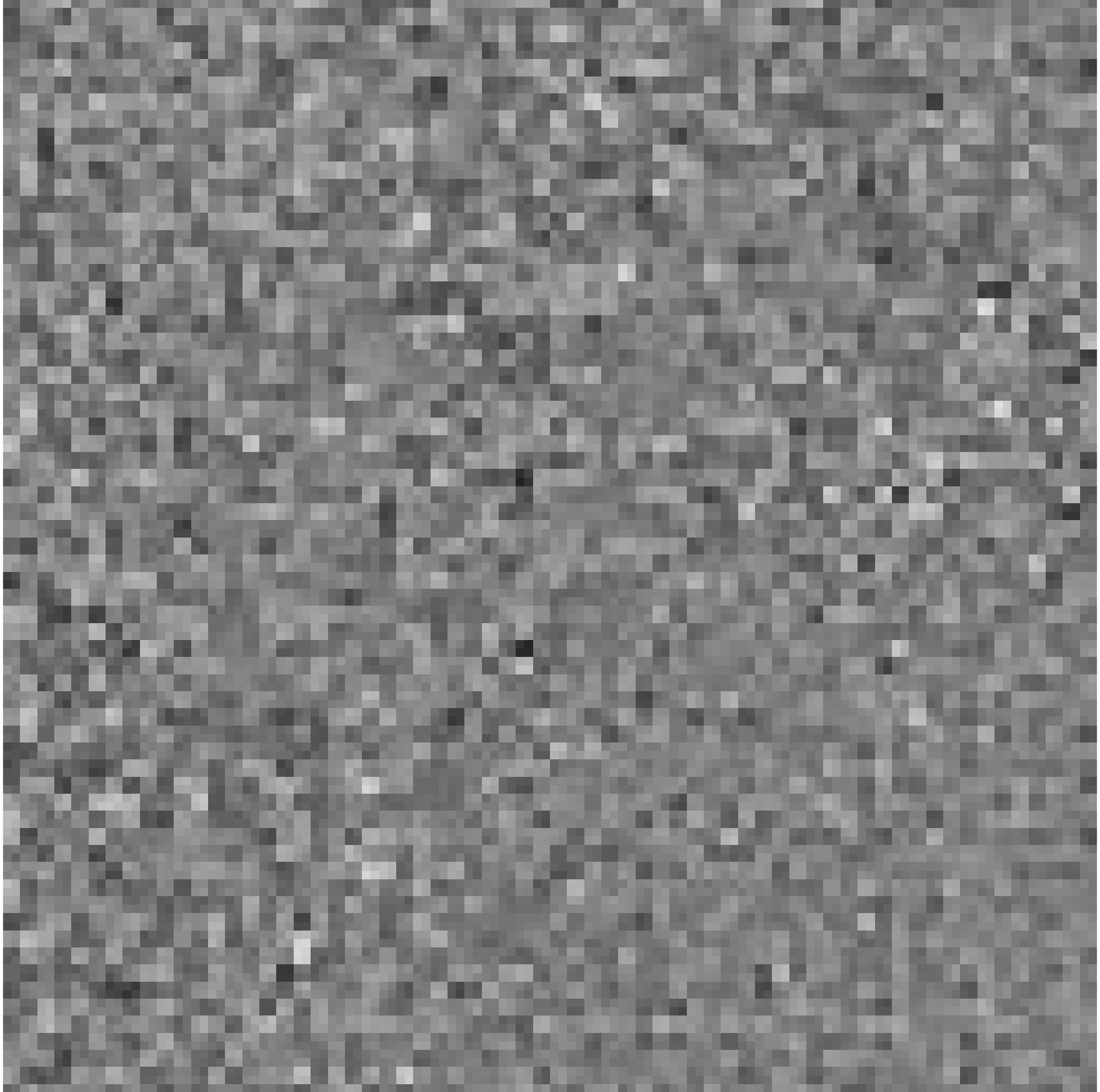
- A. Buades, B. Coll, J.-M. Morel: A non-local algorithm for image denoising. *IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 2, p. 60-65, 2005.
- A. A. Erfos, T. K. Leung: Texture Synthesis by Non-parametric Sampling. *IEEE International Conference on Computer Vision*, vol. 2, p. 1033-1038, 1999.
- `http://de.wikipedia.org/wiki/Wav`.
- `http://en.wikipedia.org/wiki/Window_function`.

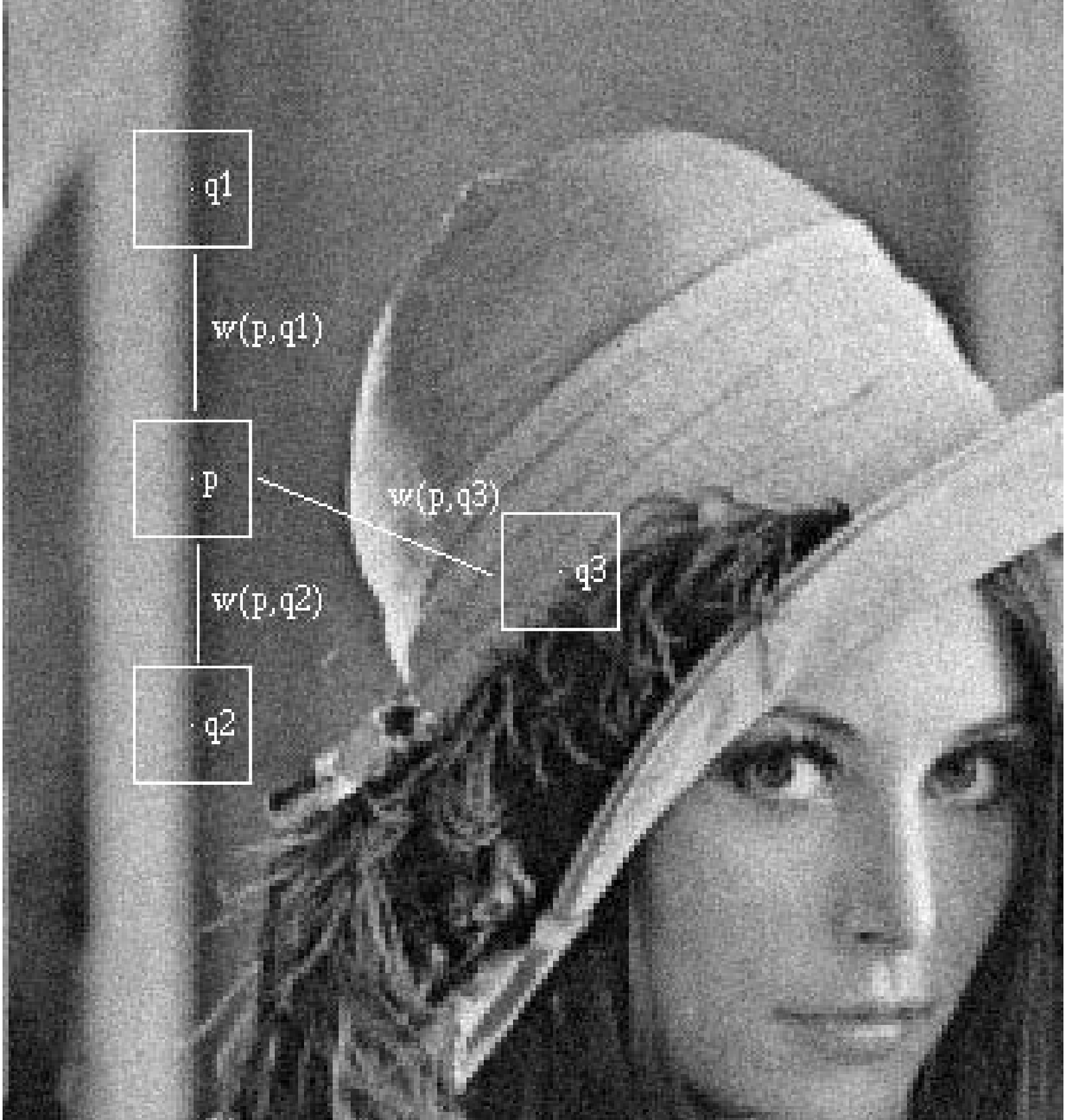
Thank you for your attention!

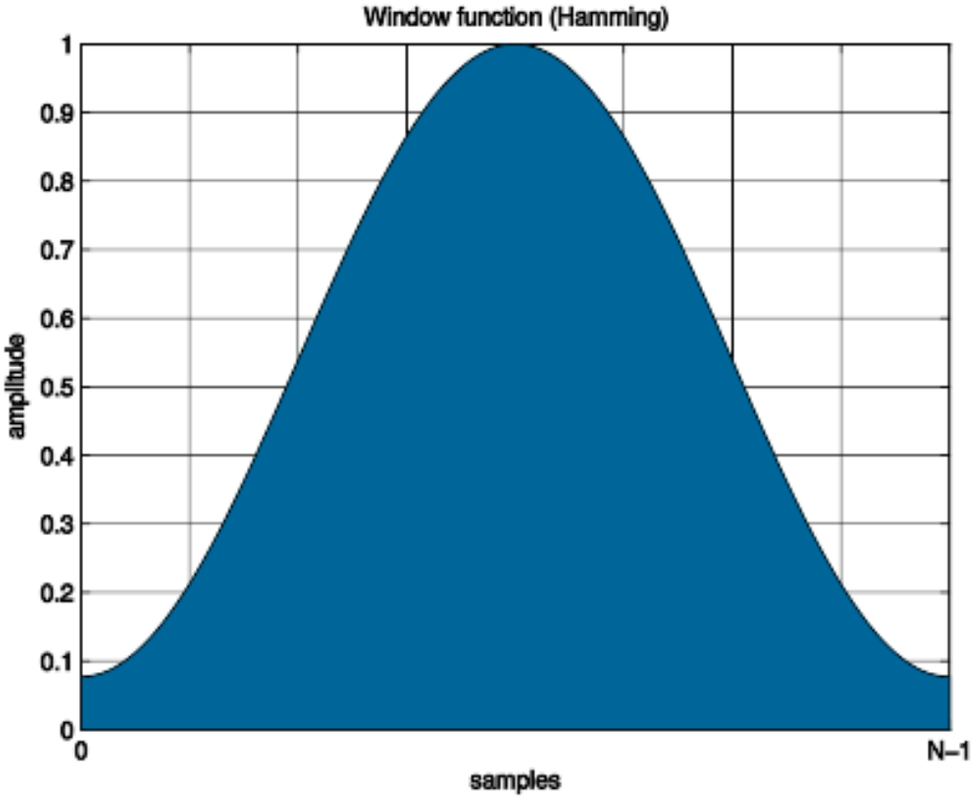

Subchunk1Size = 16
NumChannels = 2
ChunkSize = 2084
AudioFormat = 1 (PCM)
chunk descriptor
fmt subchunk
52 49 46 46 24 08 00 00 57 41 56 45
R I F F W A V E
66 6d 74 20 10 00 00 00 01 00 02 00
f m t
ByteRate = 88200
BitsPerSample = 16
SampleRate = 22050
BlockAlign = 4
Subchunk2Size = 2048
22 56 00 00 88 58 01 00 04 00 10 00
data subchunk
64 61 74 61 00 08 00 00 00 00 00 00
d a t a
sample 1
24 17 1e f3 3c 13 3c 14 16 f9 18 f9 34 e7 23 a6 3c f2 24 f2 11 ce 1a 0d
sample 2
sample 3
sample 4
sample 5
sample 6
sample 7
right channel samples
left channel samples

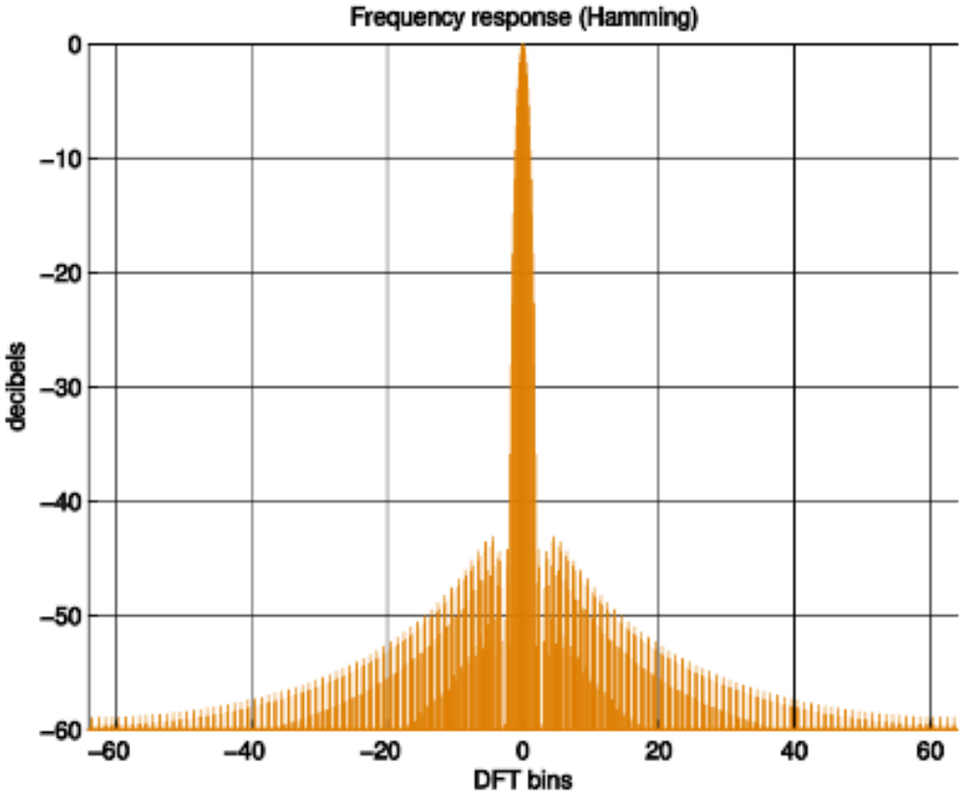


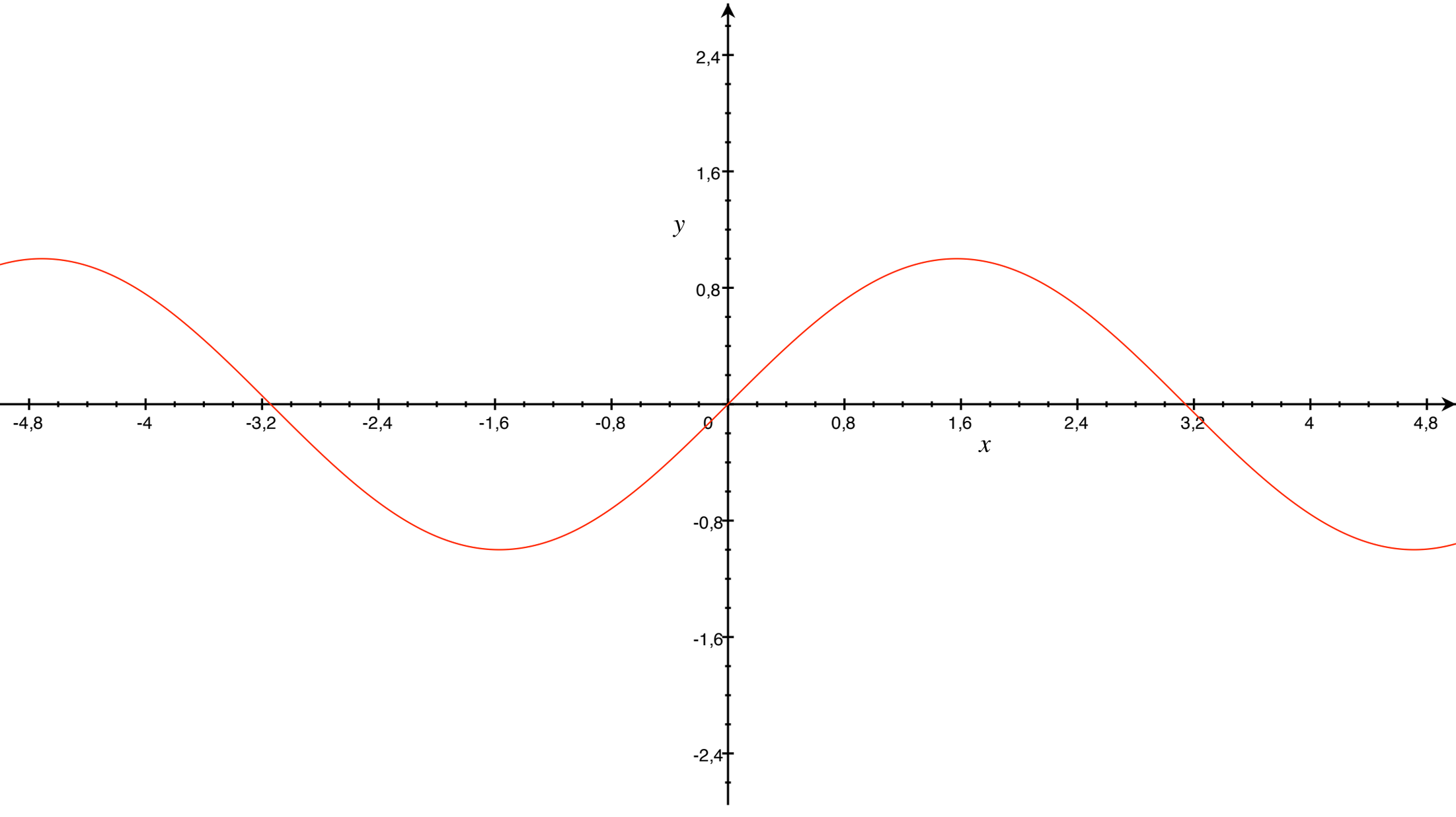


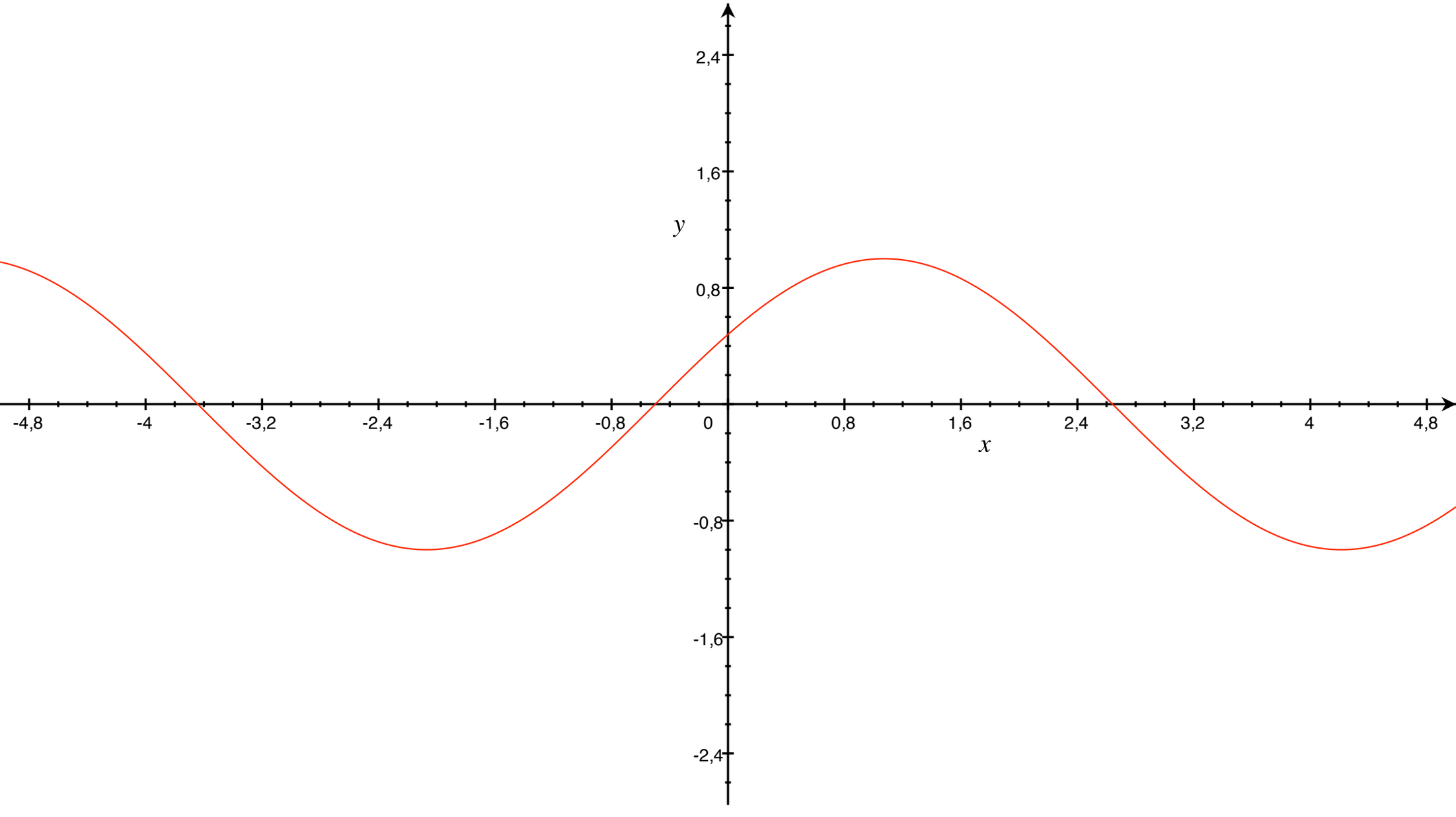



q1
w(p,q1)
p
w(p,q3)
q3
w(p,q2)
q2


Window function (Hamming)
amplitude
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
N–1
samples


Frequency response (Hamming)
decibels
0
–10
–20
–30
–40
–50
–60
–60
–40
–20
0
20
40
60
DFT bins


2,4
1,6
y
0,8
-4,8
-4
-3,2
-2,4
-1,6
-0,8
0
0,8
1,6
2,4
3,2
4
4,8
x
-0,8
-1,6
-2,4


2,4
1,6
y
0,8
-4,8
-4
-3,2
-2,4
-1,6
-0,8
0
0,8
1,6
2,4
3,2
4
4,8
x
-0,8
-1,6
-2,4