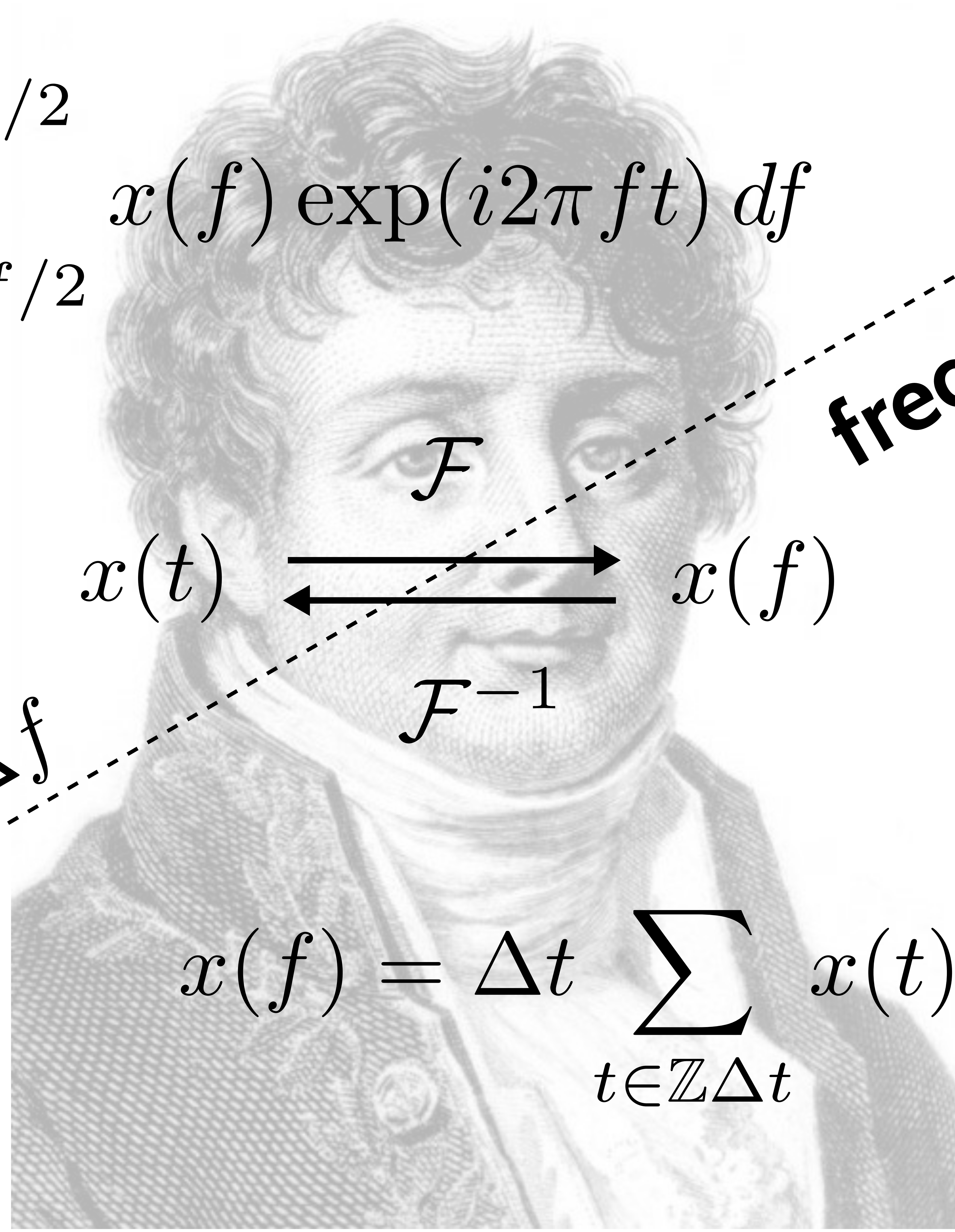
Spectral Analysis

Digital Audio Coding

SEBASTIEN.BOISGERAULT@MINES-PARISTECH.FR

Discrete-Time Fourier Transform



$$x(t) = \int_{-\Delta f/2}^{\Delta f/2} x(f)\exp(i2\pi ft)\,df$$

frequency domain

$f \in \mathbb{R}/\mathbb{Z}\Delta f$

$\Delta f = 1/\Delta t$

$$x(t) \underset{\mathcal{F}^{-1}}{\overset{\mathcal{F}}{\rightleftarrows}} x(f)$$

time domain

$t \in \mathbb{Z}\Delta t$

$\Delta t = 1/\Delta f$

$$x(f) = \Delta t \sum_{t\in\mathbb{Z}\Delta t} x(t)\exp(-i2\pi ft)$$

Spectral Decomposition Problem

Given $x(t) : \mathbb{Z}\Delta t \to \mathbb{R}$ **,**

find $\left| \begin{array}{l} a(f) : \mathbb{R} \to \mathbb{R}_+ \\ \phi(f) : \mathbb{R} \to [-\pi, \pi) \end{array} \right.$

such that $\forall\, t \in \mathbb{Z}\Delta t$**,**

$$x(t) = \int_0^{+\infty} a(f) \cos(2\pi f t + \phi(f))\, df$$

Complex Exponentials

Instead, search for $x(f) : \mathbb{R} \to \mathbb{C}$ **such that**

$$x(t) = \int_{-\infty}^{+\infty} x(f) \exp(i2\pi f t) \, df$$

(with $x(-f) = \overline{x(f)}$ **)**

then $\left| \begin{array}{l} a(f) = 2|x(f)| \\ \phi(f) = \angle x(f) \end{array} \right.$

Uniqueness of the Decomposition

Define $\Delta f = \dfrac{1}{\Delta t}$

If $x(f) : \mathbb{R} \to \mathbb{C}$ **is a solution of**

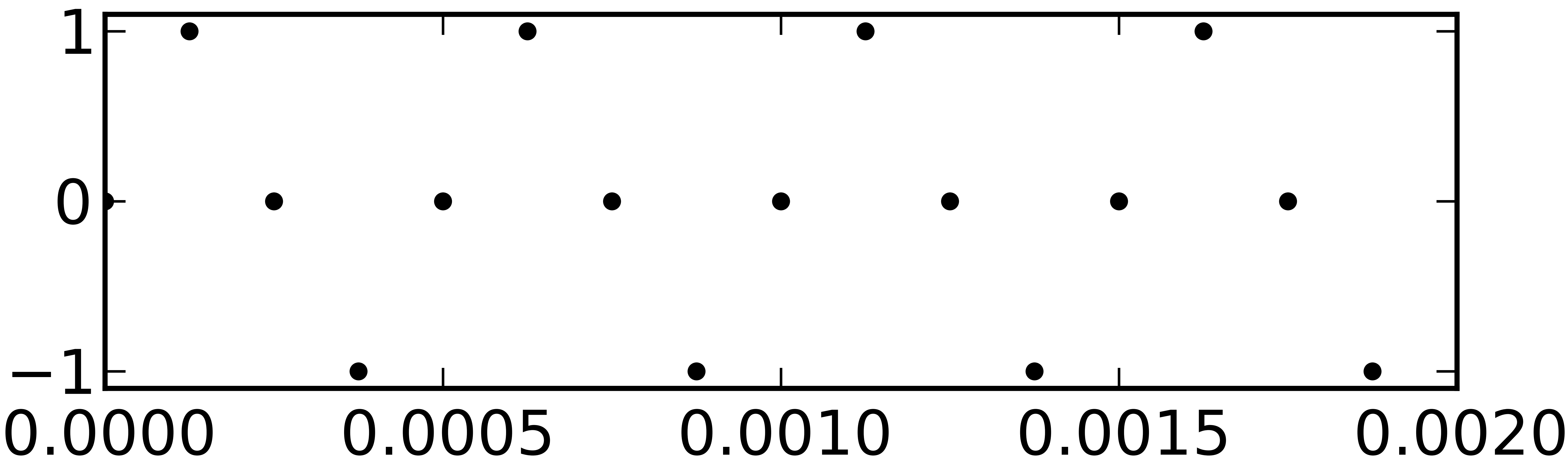
$$x(t) = \int_{-\infty}^{+\infty} x(f) \exp(i2\pi f t) \, df$$

... so is $x(f - k\Delta f)$ **for any** $k \in \mathbb{Z}$.

?

Frequency Ambiguity

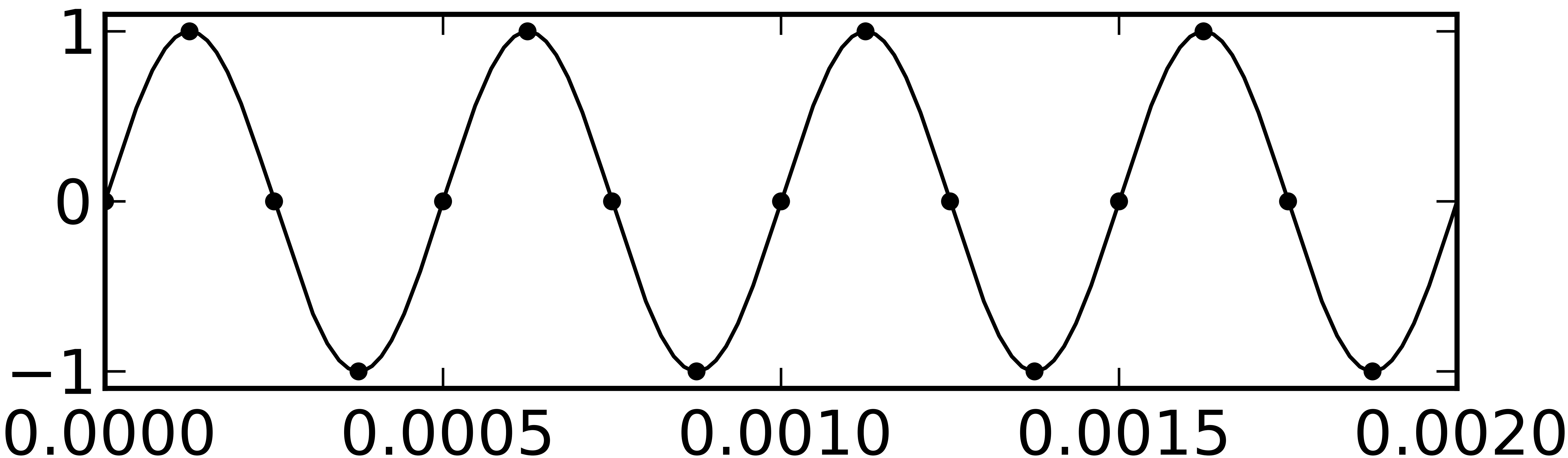
What is the frequency of this signal ?



$$\Delta f = 8000 \text{ Hz}$$

Frequency Ambiguity

What is the frequency of this signal ?

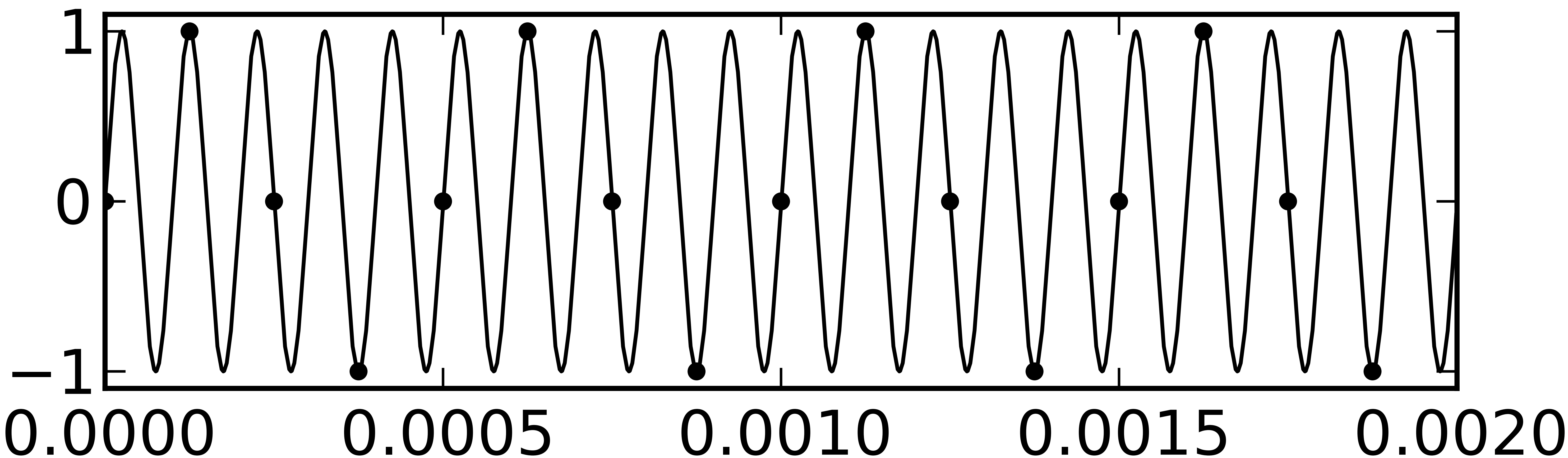


$$f = 2000 \text{ Hz}$$

$$\Delta f = 8000 \text{ Hz}$$

Frequency Ambiguity

What is the frequency of this signal ?



$$f = 10000 \text{ Hz}$$

$$\Delta f = 8000 \text{ Hz}$$

Spectral Decomposition - Conclusion

Search for $x(f) : \mathbb{R} \to \mathbb{C}$, solution of

$$x(t) = \int_{-\Delta f/2}^{+\Delta f/2} x(f) \exp(i2\pi f t) \, df$$

and enforce $\Delta f -$ periodicity of $x(f)$.

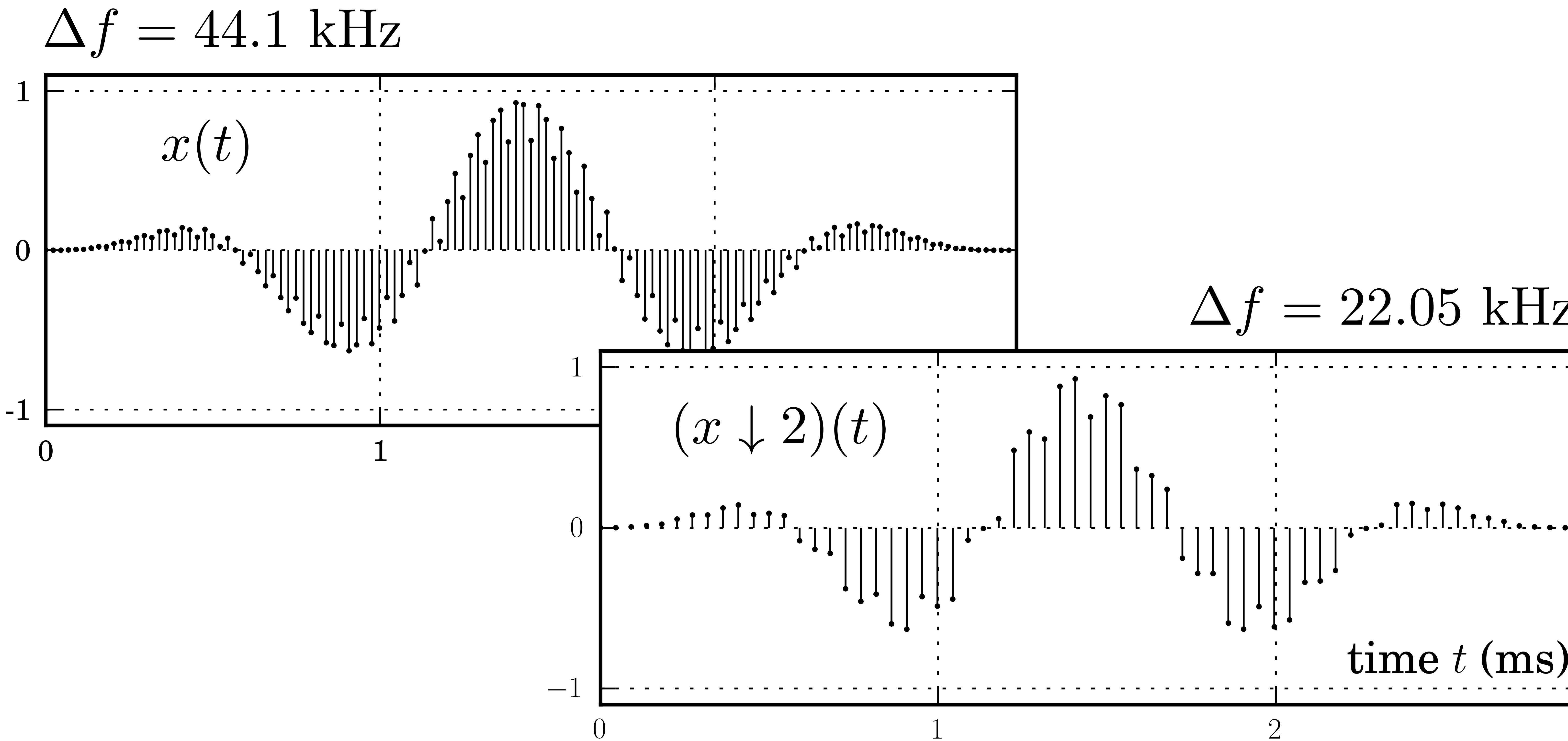
These choices yield uniqueness and:

$$x(f) = \Delta t \sum_{t \in \mathbb{Z}\Delta t} x(t) \exp(-i2\pi f t)$$

Decimation

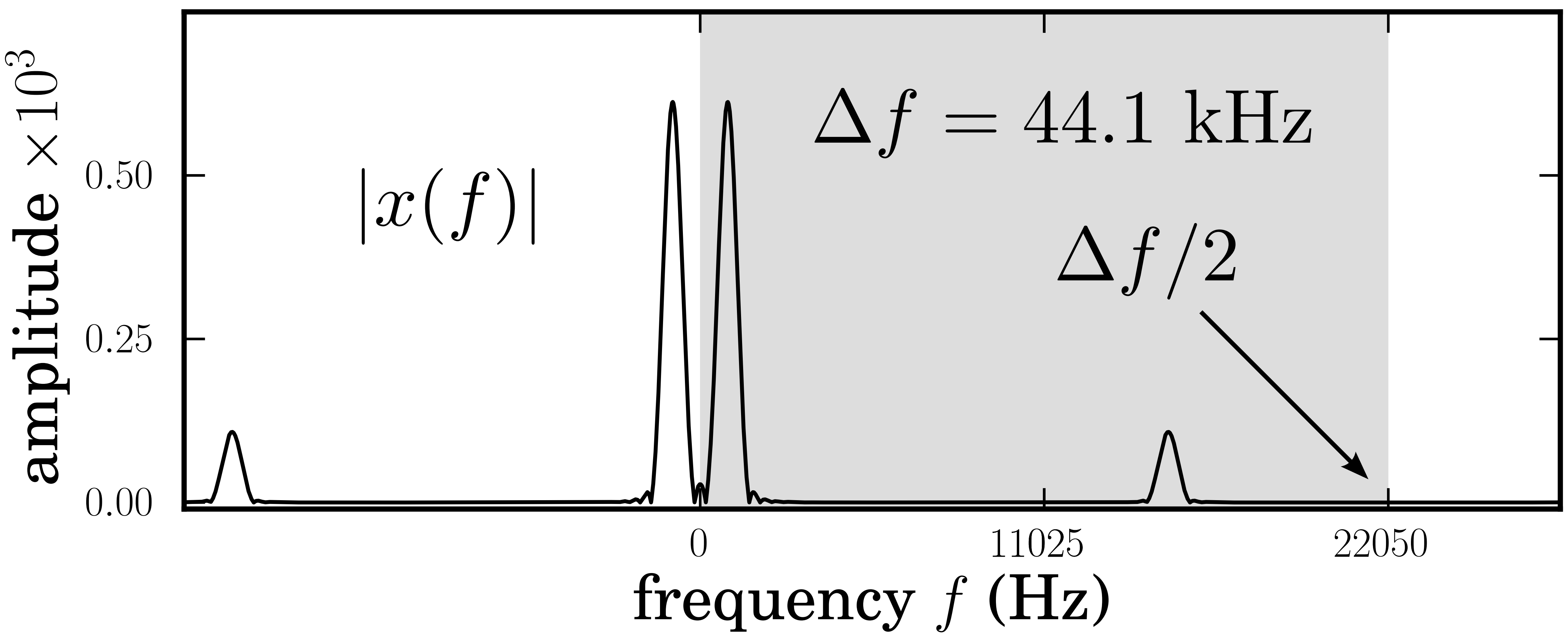
$$x : \mathbb{Z}\Delta t \to \mathbb{R} \quad \longrightarrow \quad x \downarrow M : \mathbb{Z}M\Delta t \to \mathbb{R}$$

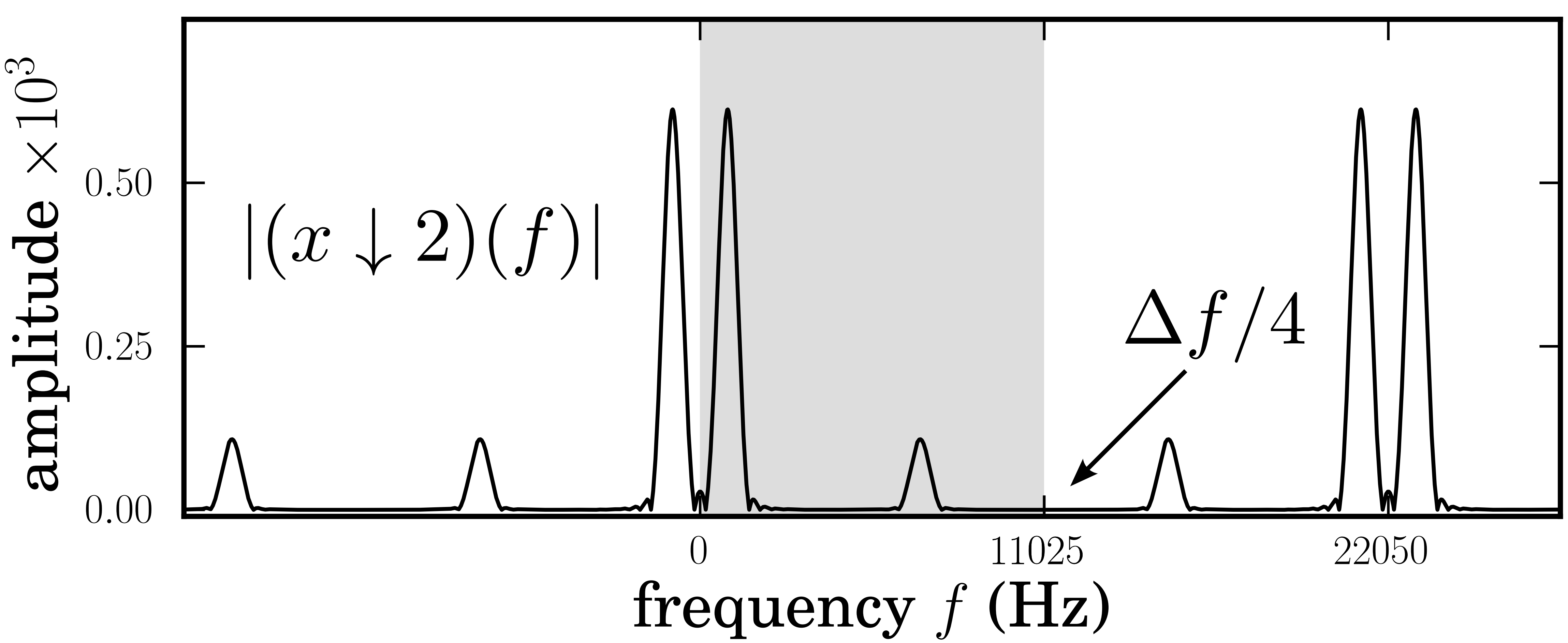
$$(x \downarrow M)(t) = x(t),\ t \in \mathbb{Z}M\Delta t$$



Decimation - Spectrum

$$(x \downarrow 2)(f) = x(f) + x(f + \Delta f/2)$$





Filtering

It's multiplication in the frequency domain:

$$y(f) = h(f)u(f)$$

or signal **convolution** in the time domain:

$$y(t) = (h * u)(t) = \Delta t \sum_{t' \in \mathbb{Z}\Delta t} h(t')u(t - t')$$

$h(t)$ is the filter **impulse response**:

$$y(t) = h(t) \;\text{ if }\; u(t) = \begin{cases} 1/\Delta t & \text{if } \; t = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Low-Pass Filter

Defined by the **frequency response**:

$$h(f) = \left| \begin{array}{ll} 1 & \text{if } \ |f| \leq f_c \\ 0 & \text{otherwise} \end{array} \right. \quad \text{for } \ |f| \leq \Delta f/2$$

where f_c is the **cutoff frequency**.

In the time domain:

$$h(t) = 2f_c \operatorname{sinc} 2f_c t \quad \text{with} \quad \operatorname{sinc} x = \frac{\sin \pi x}{\pi x}$$

The impulse response is **acausal** and **infinite**.

Low-Pass Filter

Concrete implementations are **causal** and **finite**:

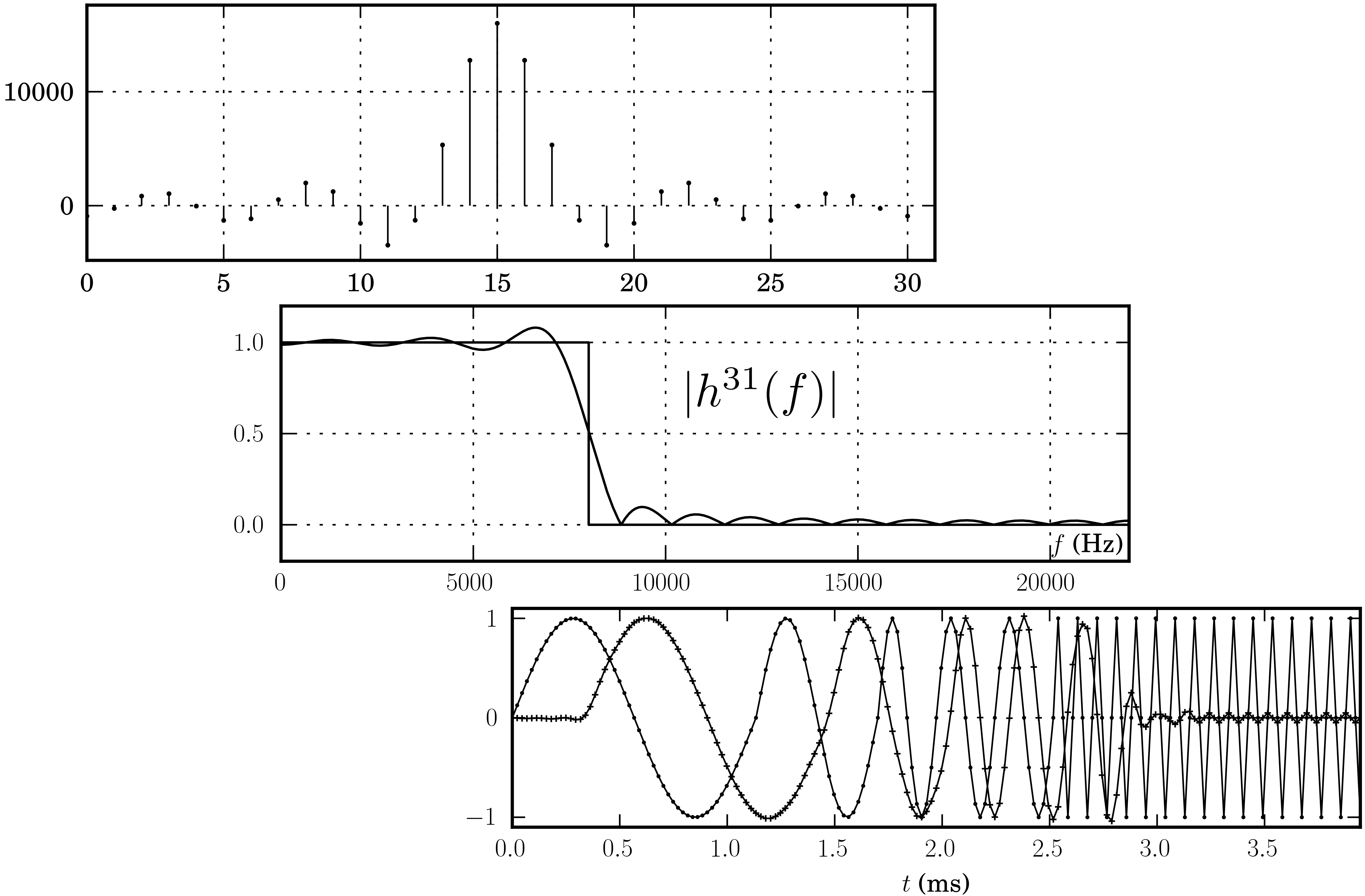
```
def low_pass(fc, dt=1.0, window=ones):
    def h(n):
        t = arange(-0.5 * (n-1), 0.5 * (n-1) + 1) * dt
        return 2 * fc * sinc(2 * fc * t) * window(n)
    return h
```

Filter the (finite, causal) signal $u(t)$:

```
>>> N = 31
>>> h = low_pass(fc=8000.0, dt=1.0/44100.0)(N)
>>> y = dt * convolve(h, u)
```

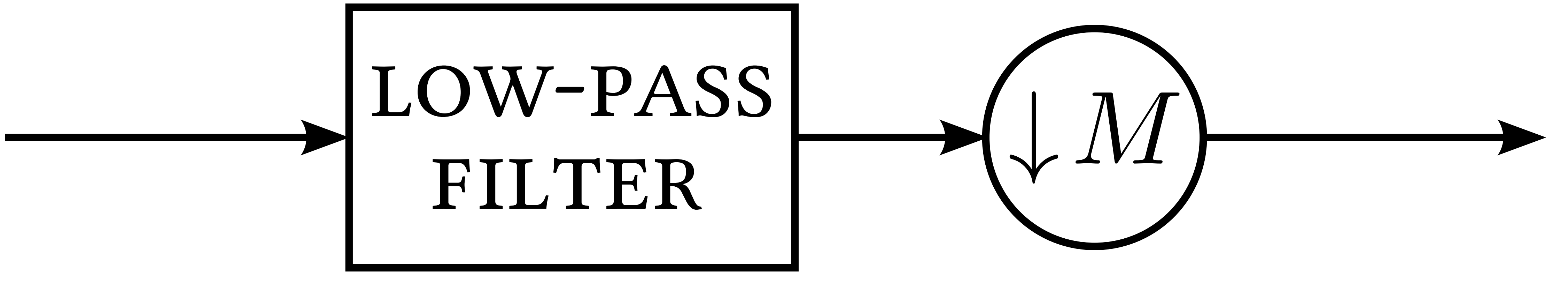
Low-Pass Filter

$$h_n^{31} = 16000\,\mathrm{sinc}\,16000(n-15),\ n \in \{0, \cdots, 30\}$$



Downsampling

$$f_c = \frac{\Delta f}{2M}$$



F.T. & Signal Energy

Parseval theorem for Fourier series yields:

$$\Delta t \sum_{t \in \mathbb{Z}\Delta t} |x(t)|^2 = \int_{-\Delta f/2}^{\Delta f/2} |x(f)|^2 \, df$$

and proves that $\mathcal{F}$ is an isomorphism:

$$L^2(\mathbb{Z}\Delta t) \longleftrightarrow L^2(\mathbb{R}/\mathbb{Z}\Delta f)$$

F.T. & Measures

Some signals are not of finite energy but may be represented in the Fourier domain as **measures***:

periodic

$$x(f) = x_1(f) + \sum_k a_k \delta(f - f_k)$$

$L^2(\mathbb{R}/\mathbb{Z}\Delta f)$

$\sum_k |a_k| < +\infty$

They correspond in the time domain to:

$$x(t) = x_1(t) + \sum_k a_k \exp(i2\pi f_k t)$$

*: periodic, complex-valued, with no singular continous part.

Spectrum Computation

Finite causal signal: all values are zero but :

$$x(0),\, x(\Delta t), \cdots, x((N-1)\Delta t)$$

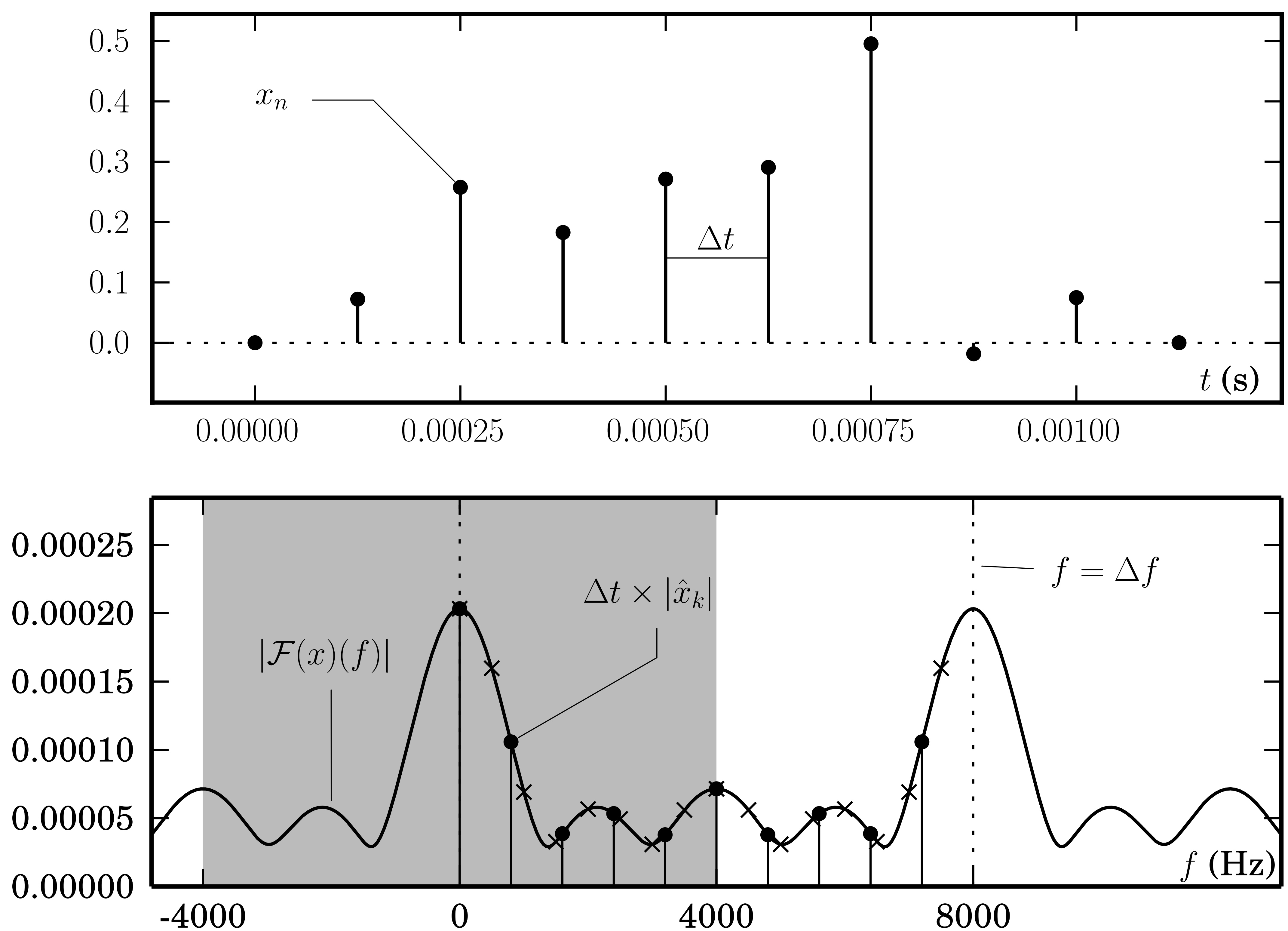
$$[\mathcal{F}x](f) = \Delta t \sum_{n=0}^{N-1} x(n\Delta t) \exp(-i 2\pi f n \Delta t)$$

```
def F(x, dt=1.0):
    n = reshape(r_[0:len(x)], (-1, 1))
    def Fx(f):
        f = reshape(f, (1, -1))
        return dt * dot(x, exp(-1j * 2 * pi * dt * n * f))
    return Fx
```

Discrete Fourier Transform - DFT

$$x_n = x(t = n\Delta t) \longrightarrow \hat{x}_k = \Delta f \times x(f = k\Delta f / N)$$

$$n = 0, 1, \cdots, N-1 \qquad\qquad k = 0, 1, \cdots, N-1$$



zero-padding increases frequency resolution

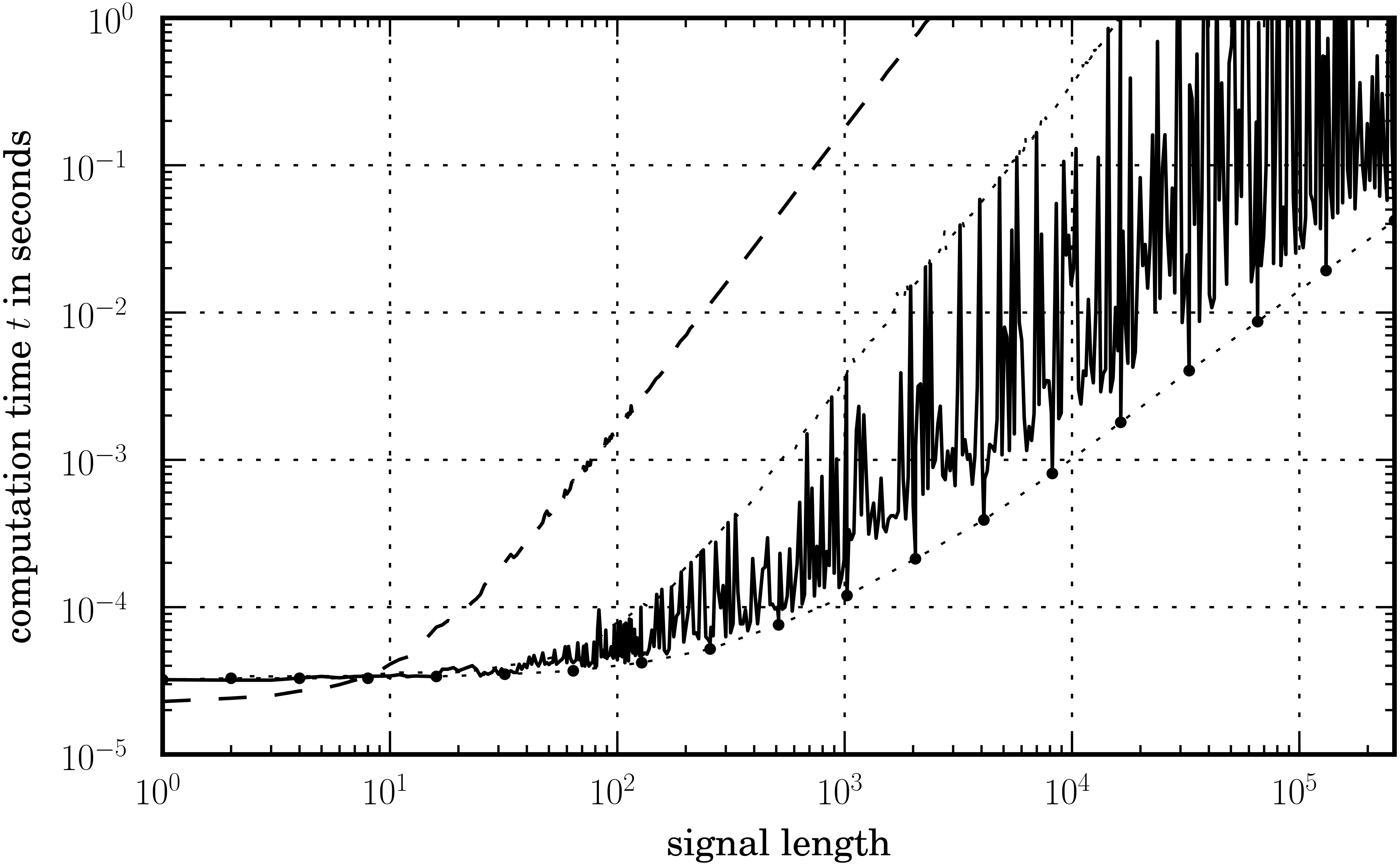
Fast Fourier Transform - FFT

The DFT is a **function**, not an algorithm, that may be used to compute some spectrum (a.k.a. (DT)FT) values, for finite and causal signals.

The naive algorithm (as a matrix-vector product) to compute the DFT has a $\mathcal{O}(N^2)$ complexity.

FFT refer to a **family of algorithms** that perform the DFT with a $\mathcal{O}(N \log N)$ complexity.

FFT Benchmarks

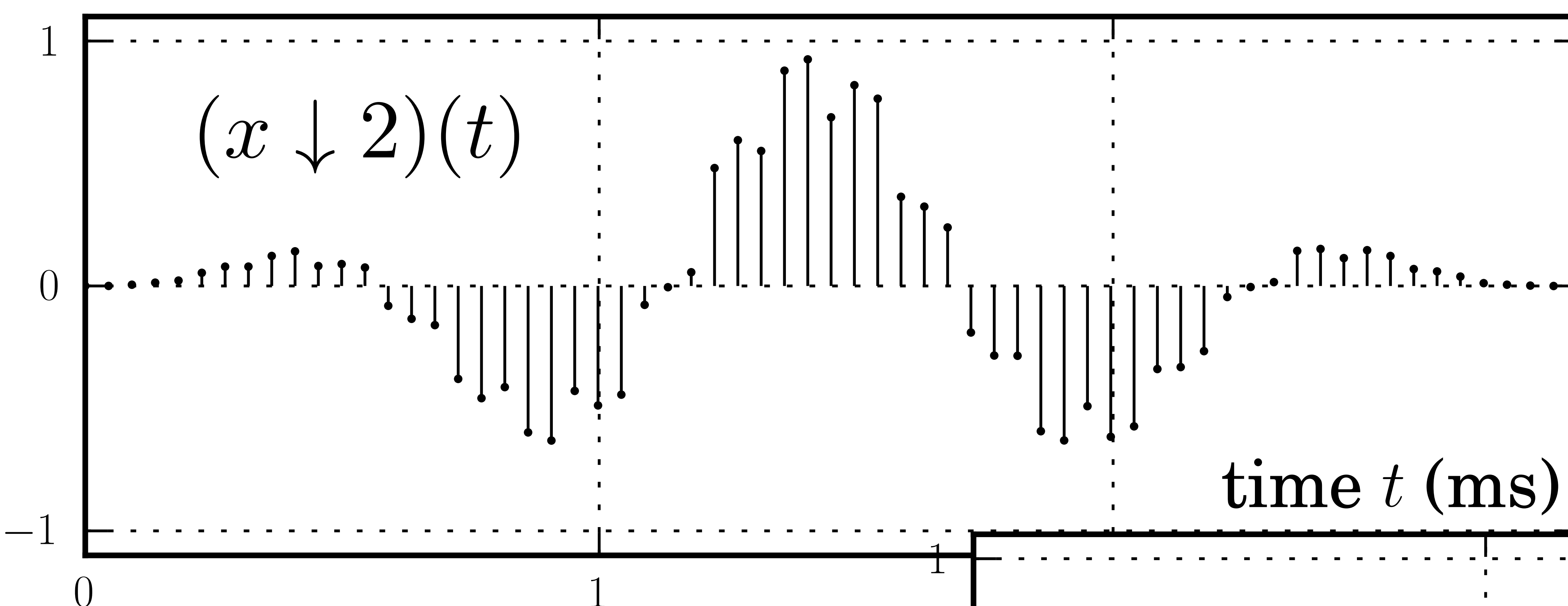


Expansion

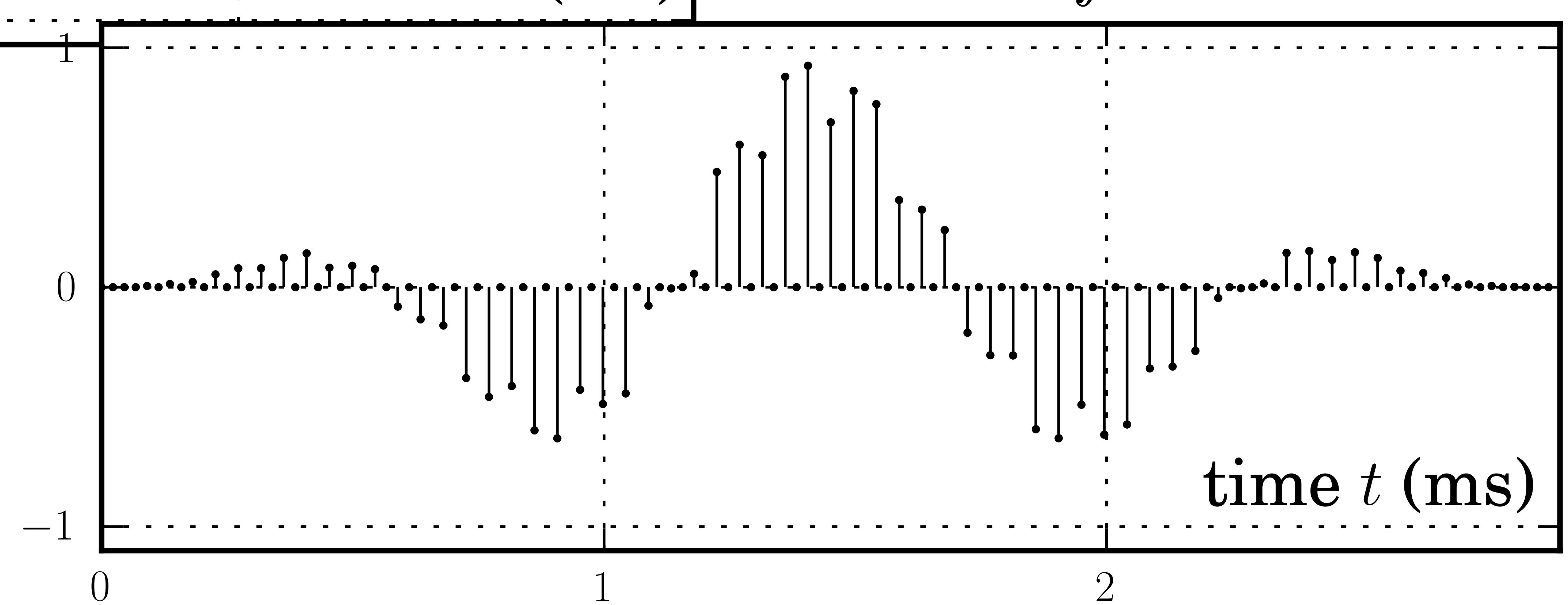
$$x : \mathbb{Z}\Delta t \to \mathbb{R} \longrightarrow x \uparrow M : \mathbb{Z}\Delta t / M \to \mathbb{R}$$

$$(x \uparrow M)(t) = x(t), \text{ if } t \in \mathbb{Z}\Delta t, \ 0 \text{ otherwise.}$$

$\Delta f = 22.05$ kHz

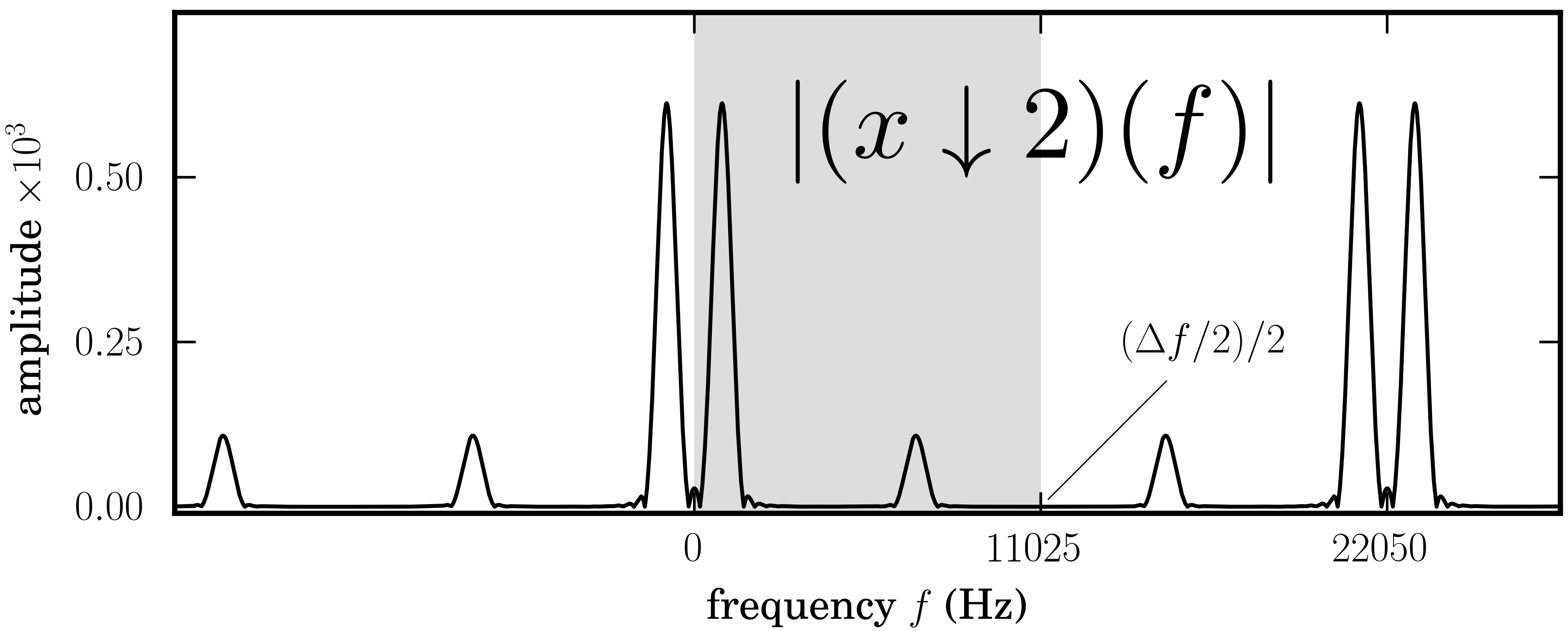


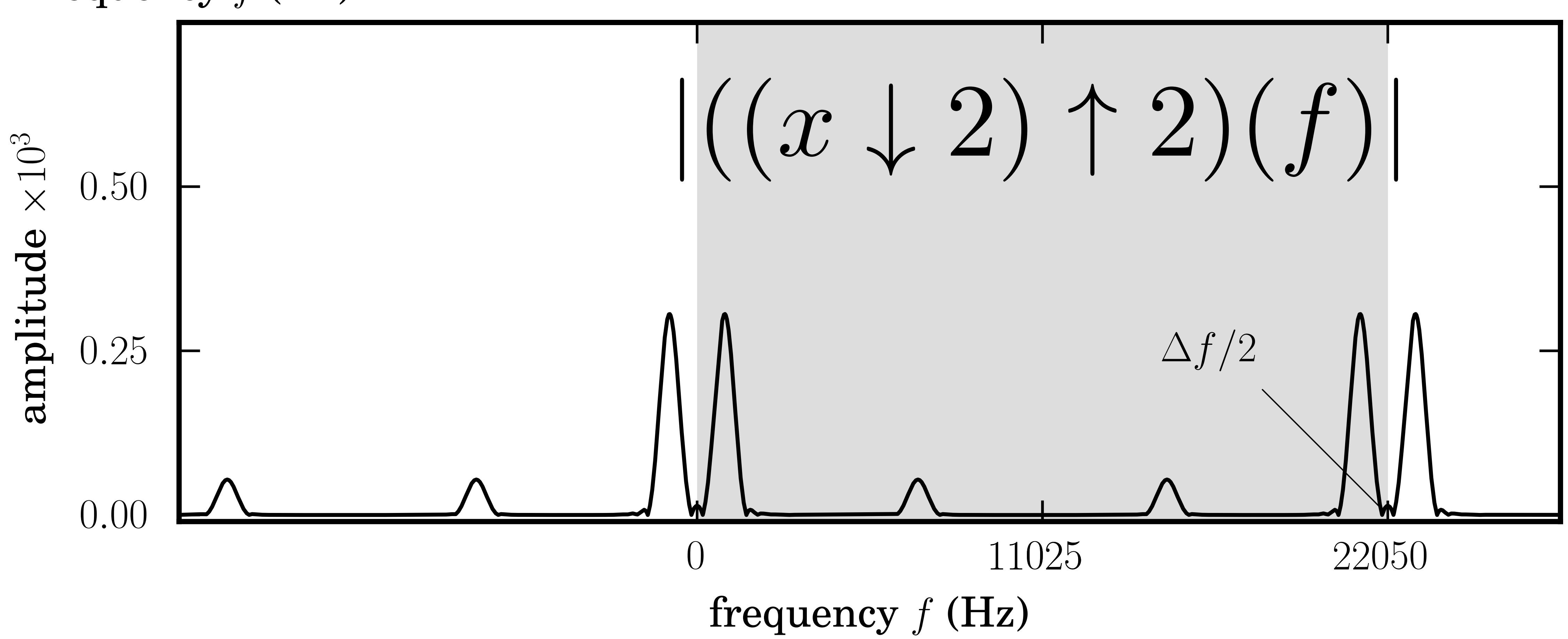
$\Delta f = 44.1$ kHz



Expansion - Spectrum

$$(x \uparrow M)(f) = \frac{1}{M} x(f)$$





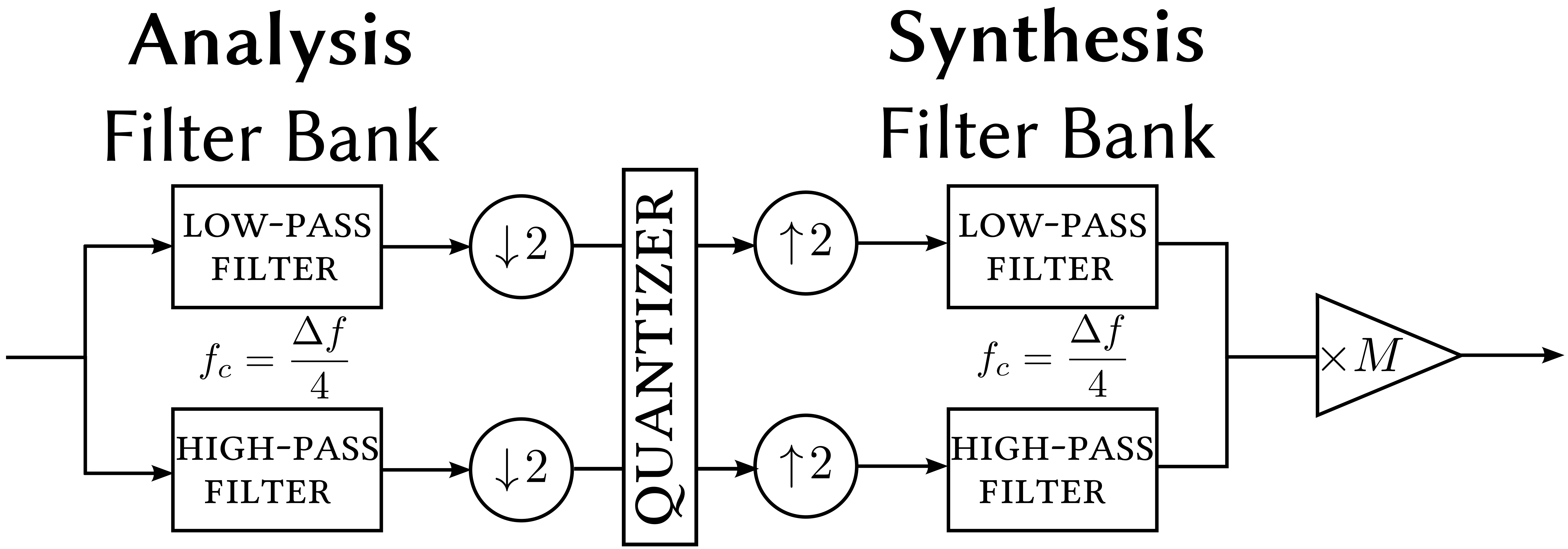
Upsampling

$$f_c = \frac{\Delta f}{2M}$$

Downsampling

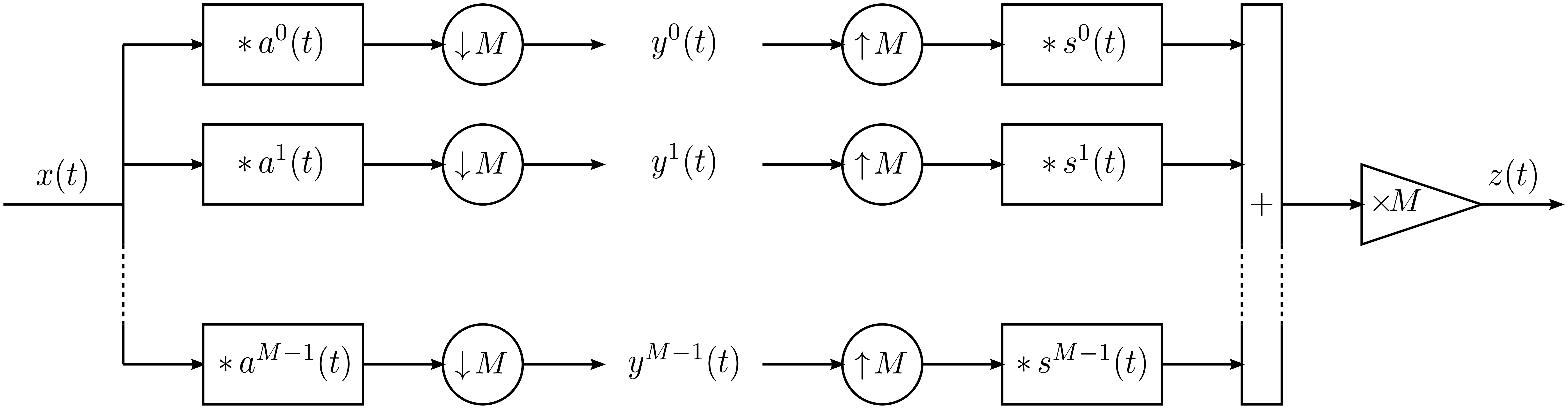
$$f_c = \frac{\Delta f}{2M}$$

Perfect Reconstruction



If the filters are perfect and there is no quantization,
the reconstruction is perfect.

Filter Banks



$$z(f) = \sum_{k=0}^{M-1} \left[\sum_{i=0}^{M-1} s_i(f) a_i(f + k\Delta f/M) \right] x(f + k\Delta f/M)$$

Distortion

The filter banks output satisfies:

$$z(f) = x(f) + \sum_{k=0}^{M-1} D_k(f) x(f + k\Delta f/M)$$

where the k-th distortion function D_k is

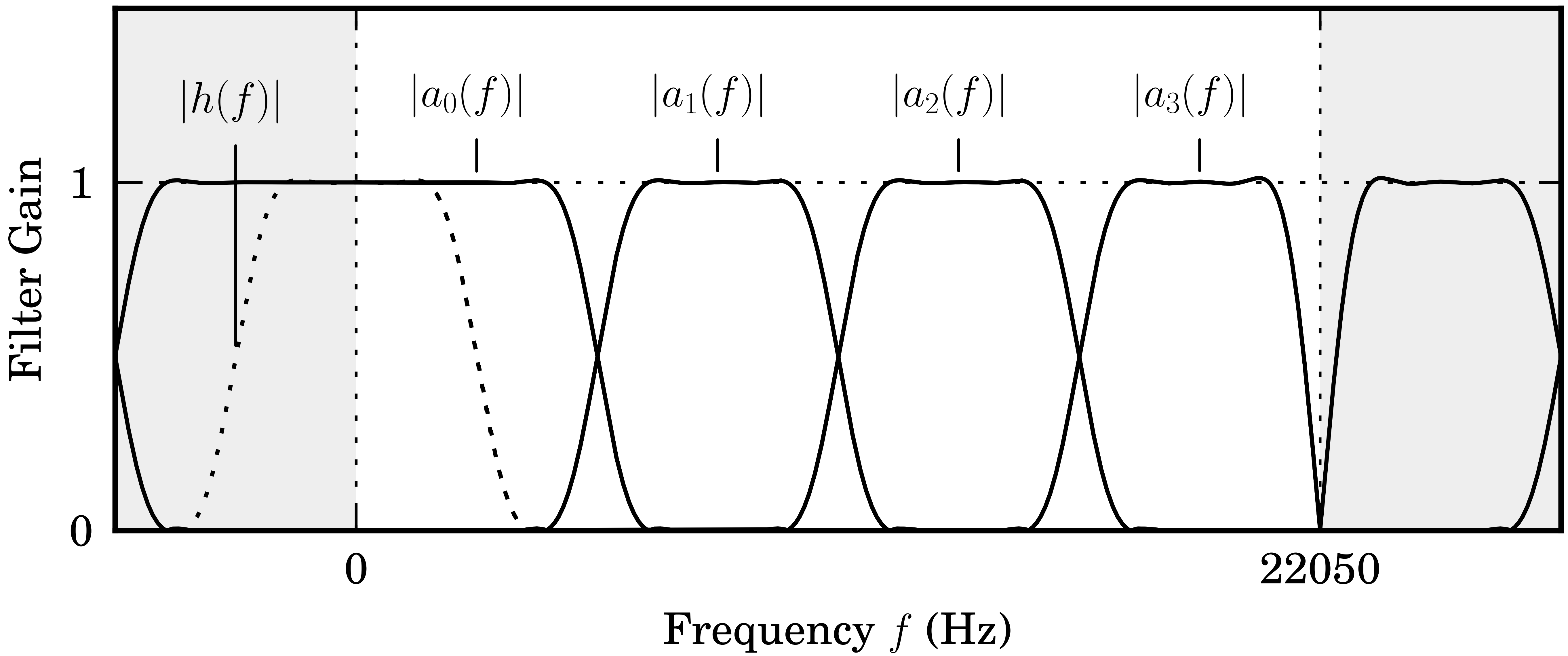
$$D_k(f) = \sum_{i=0}^{M-1} s_i(f) a_i(f + k\Delta f/M) - \delta_k$$

Perfect reconstruction:

$$\forall\, k \in \{0, \cdots M-1\},\ \forall\, f \in \mathbb{R},\ D_k(f) = 0$$

Modulated Filter Banks

Pick a lowpass filter with $f_c = \Delta f / 4M$, and a frequency response $h(f)$.



Modulate the prototype filter by

$$\Delta f_k = (k + 0.5) \times \Delta f / 2M$$

Modulated Filter Banks

Example: select a prototype truncated from

$$h(n\Delta t) = \frac{\Delta f}{2M} \operatorname{sinc} \frac{n}{2M}$$

and as

$$\exp(i2\pi\Delta f_k \times n\Delta t) = \exp(i\pi(k+0.5)n/M)$$

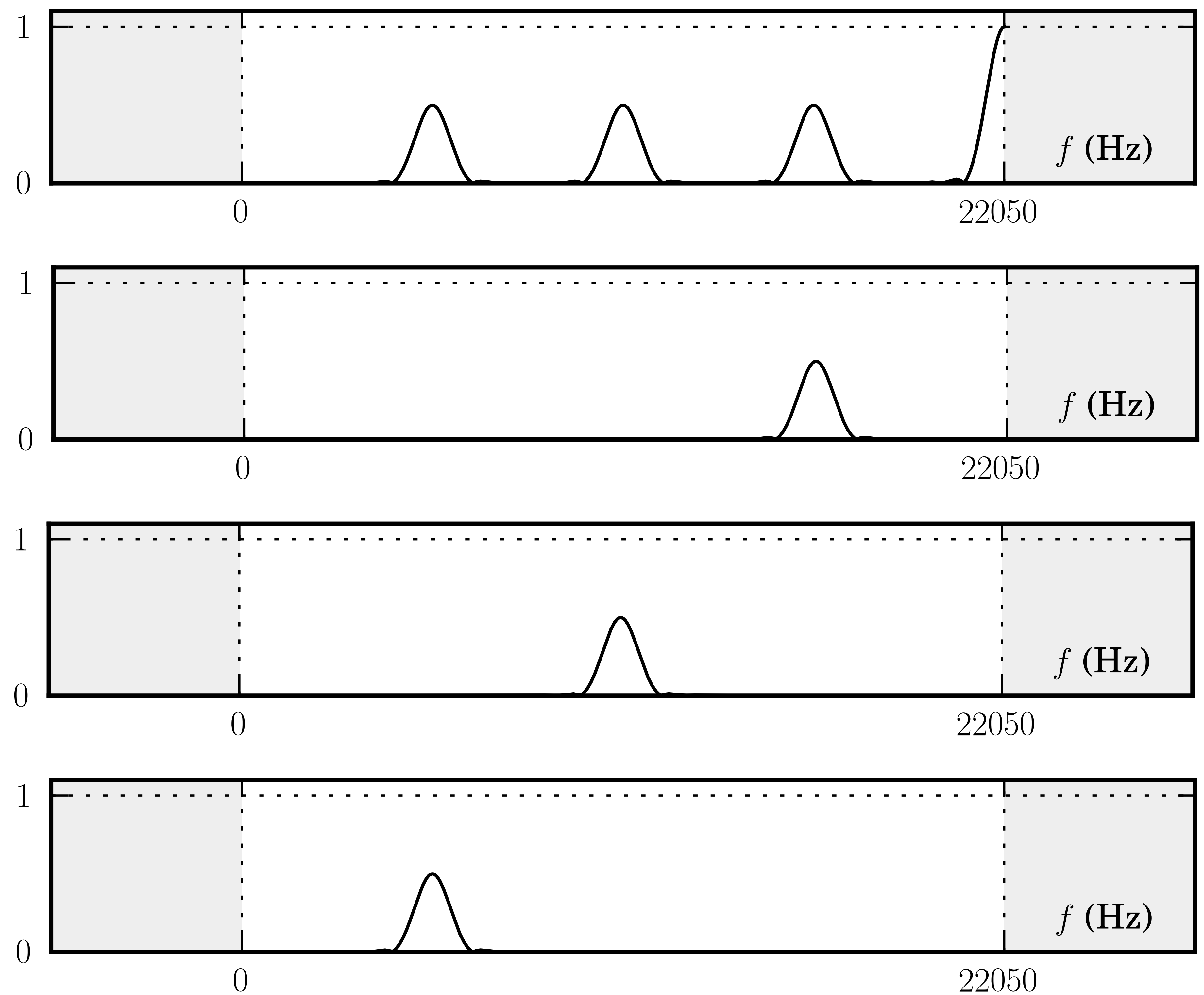
select

$$a^k(n\Delta t) = h(n\Delta t) \times 2\cos\left(\pi(k+0.5)n/M\right)$$

Modulated Filter Banks

Distortions

same analysis and synthesis filters



Pseudo-Quadrature Mirror Filters

1. Use the phase in the modulation:

$$a^k(n\Delta t) = h(n\Delta t) \times 2\cos\left(\pi(k+0.5)n/M + \phi_k\right)$$
$$s^k(n\Delta t) = h(n\Delta t) \times 2\cos\left(\pi(k+0.5)n/M - \phi_k\right)$$

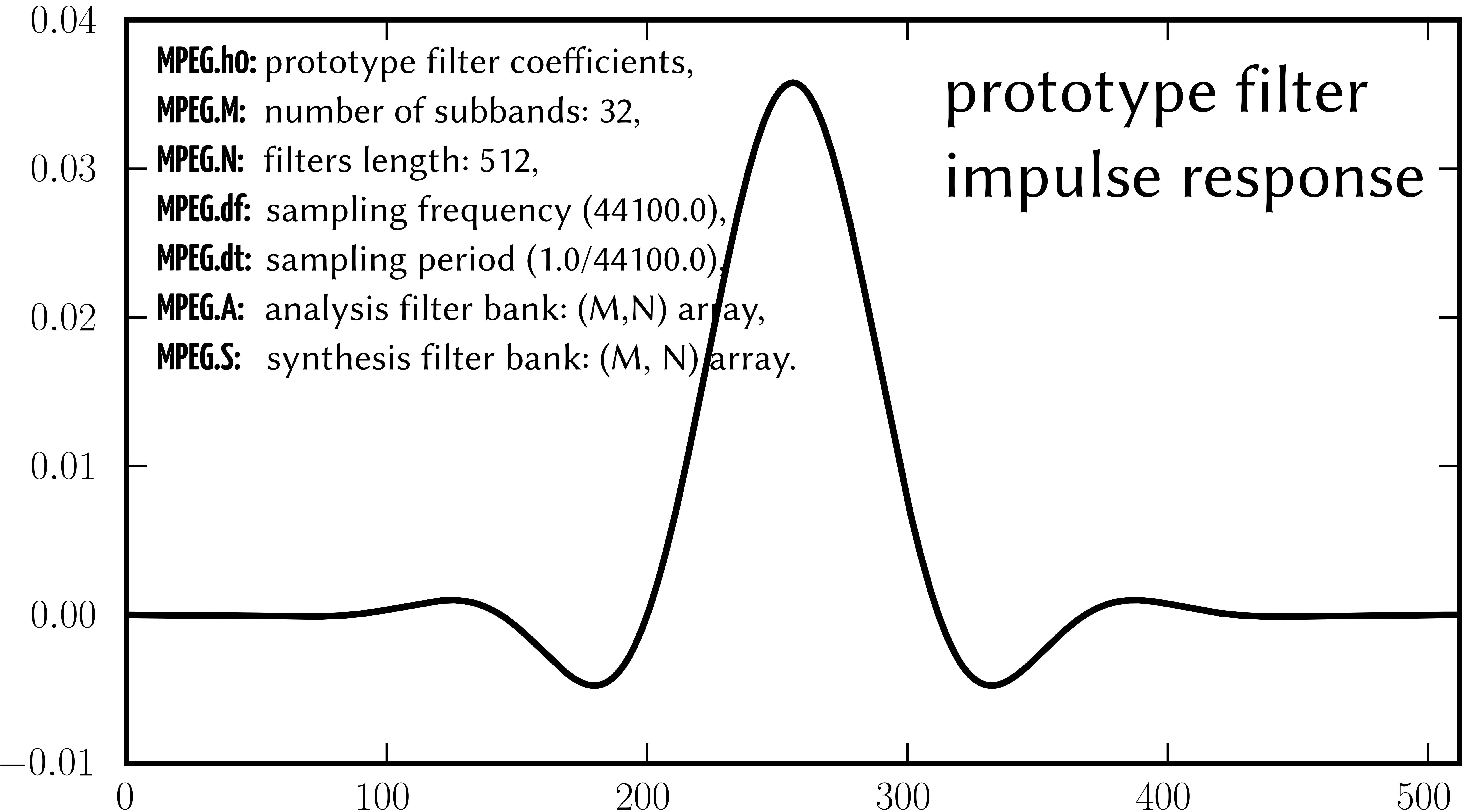
$$\phi_k = \frac{\pi}{2}\left(\frac{N-1}{M} - 1\right)(k+0.5)$$

where N is the prototype filter length.

2. Optimize the prototype filter w.r.t. the distortion.

Pseudo-Quadrature Mirror Filters

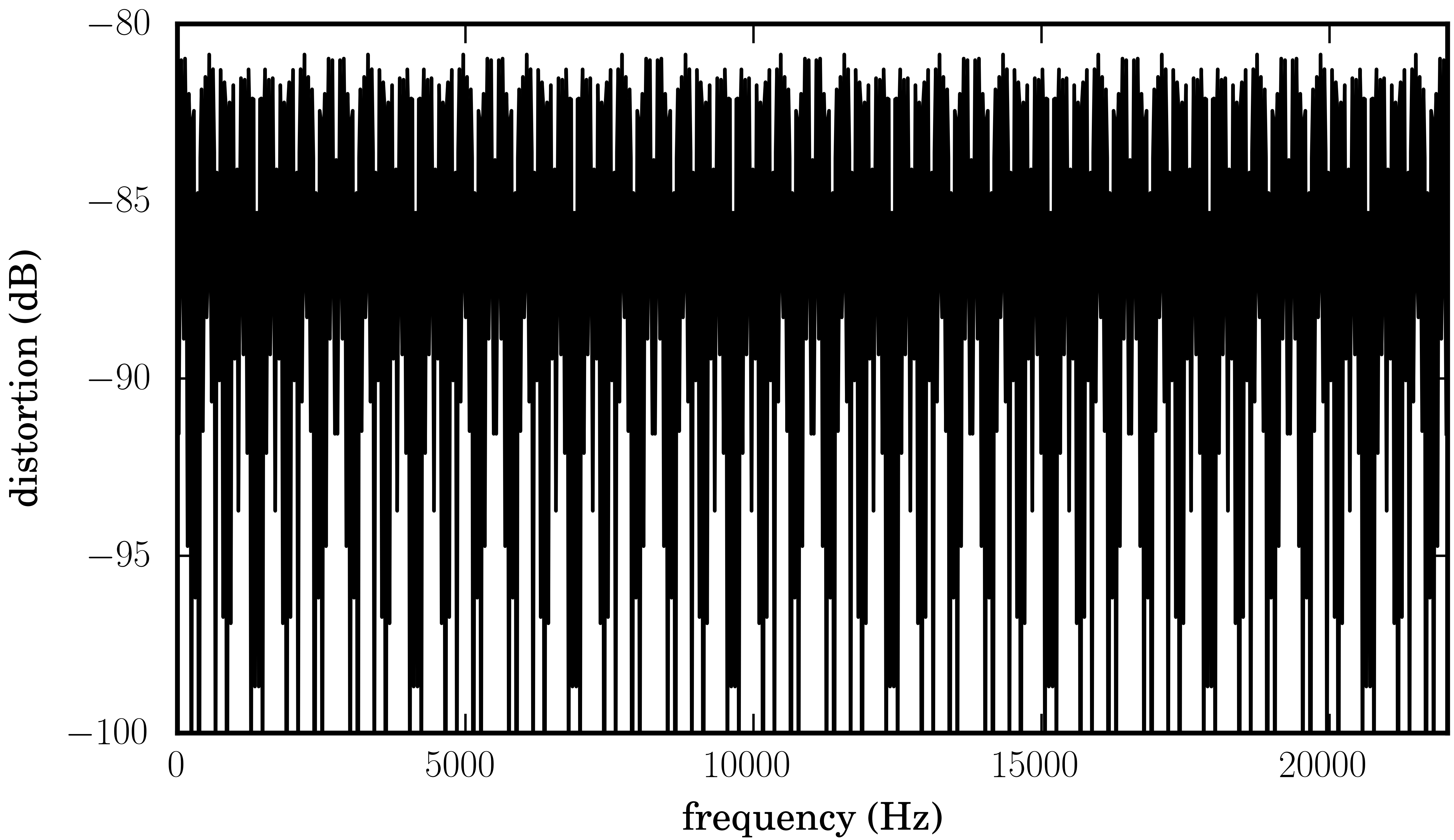
Example: MPEG Layer I + II



Pseudo-Quadrature Mirror Filters

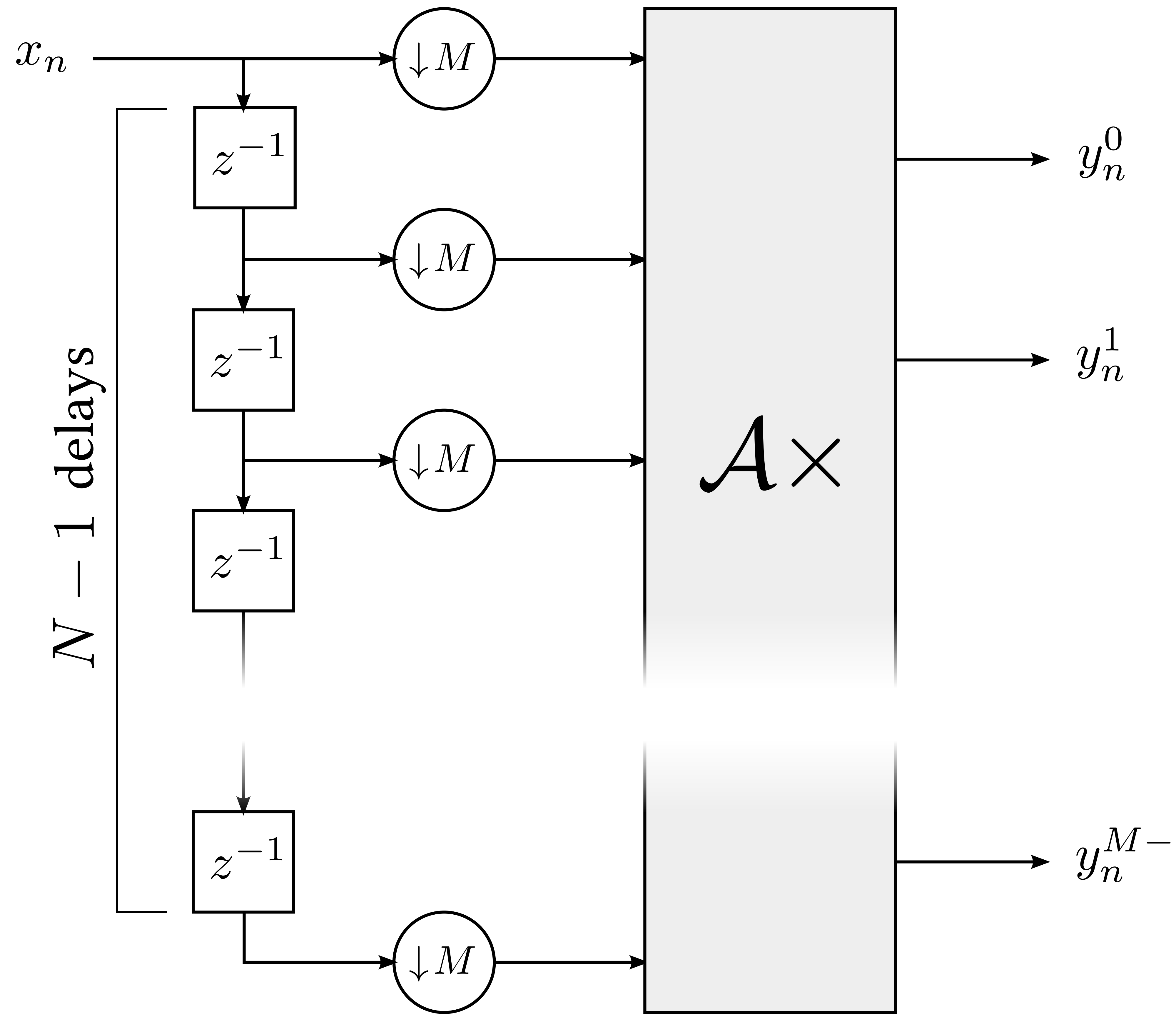
Example: MPEG Layer I + II : distortion.

$$20 \log_{10} |D_0(f)|$$



Polyphase Filter Banks

```
class Analysis(object):
  def __init__(self, a, dt=1.0):
    self.M, self.N = shape(a)
    self.A = a * dt
    self.buffer = zeros(self.N)
  def __call__(self, frame):
    buffer = self.buffer
    buffer[self.M:] = buffer[:-self.M]
    buffer[:self.M] = frame
    return dot(self.A, buffer)
```



Polyphase Filter Banks

```
class Synthesis(object):
  def __init__(self, s, dt=1.0)
    self.M, self.N = shape(s)
    self.S = self.M * dt * s
    self.buffer = zeros(self.N)
  def __call__(self, frame):
    buffer = self.buffer
    buffer[:] += dot(frame, self.S)
    output = buffer[-self.M:].copy()
    buffer[self.M:] = buffer[:-self.M]
    buffer[:self.M] = zeros(self.M)
    return output
```