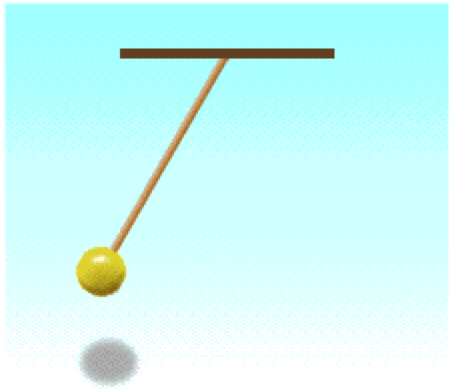
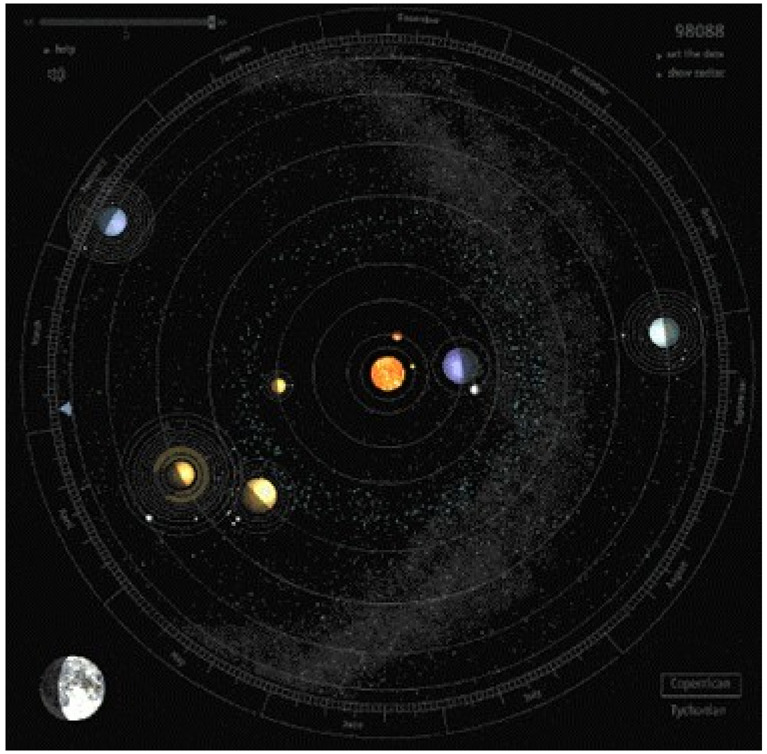
Periodic motions

- Periodic motions are known since the beginning of mankind:
  - Motion of planets around the Sun;
  - Pendulum;
  - And many more...

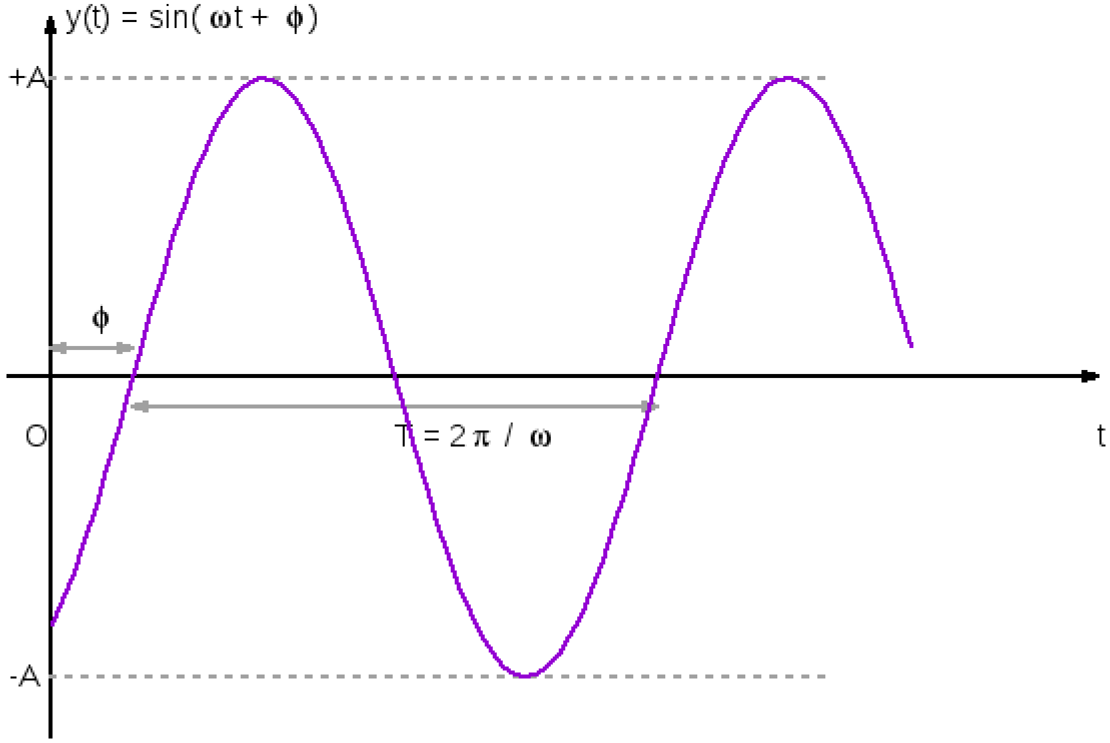



Periodic motions

- There are several quantities which describe a periodic (or oscillatory) motion (**in time**!):

- Amplitude A
- Period T
- Frequency ω

  $(=2\pi/T)$
- Phase ϕ

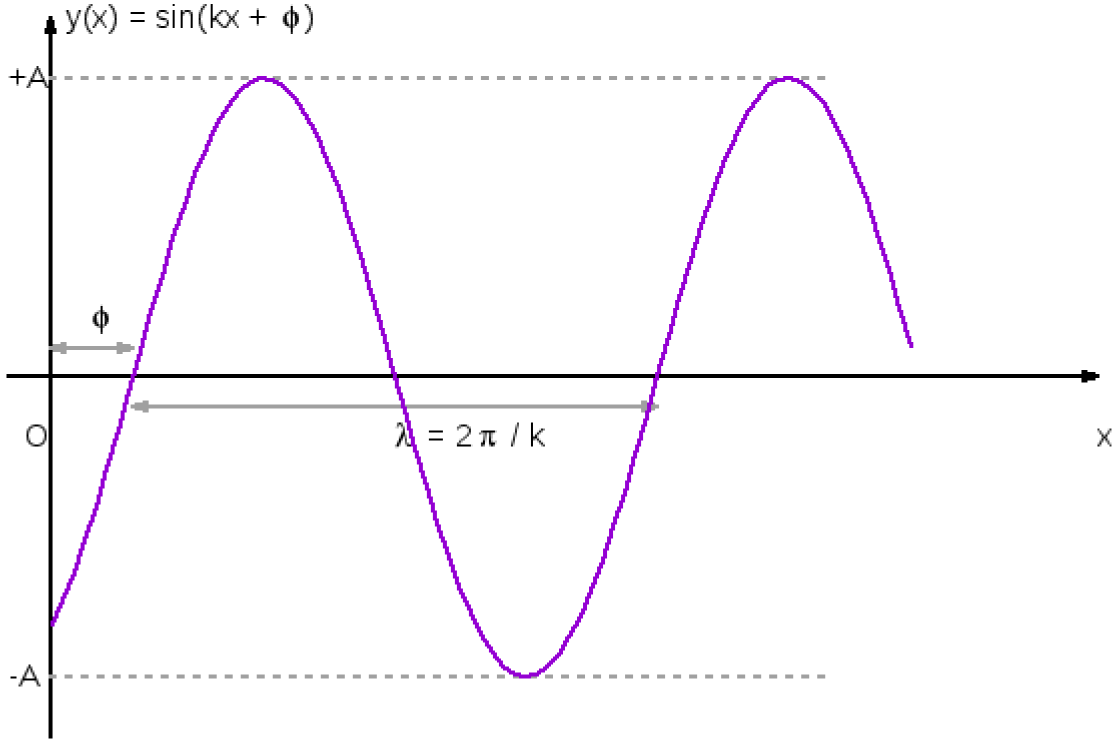


Periodic motions

- One may define analogous parameters for a **spatial oscillation**:

  - Amplitude A
  - Wavelength λ
  - Wave vector k

    $(=2\pi/\lambda)$
  - Phase ϕ

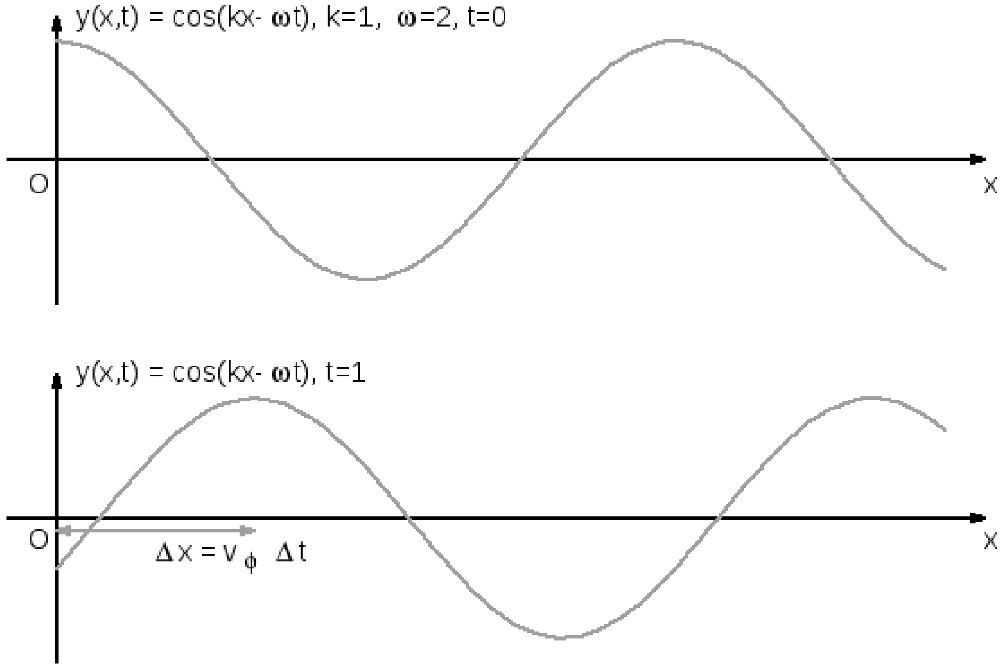


Periodic motions

- Finally, there are oscillations both in space and time, namely **waves:**

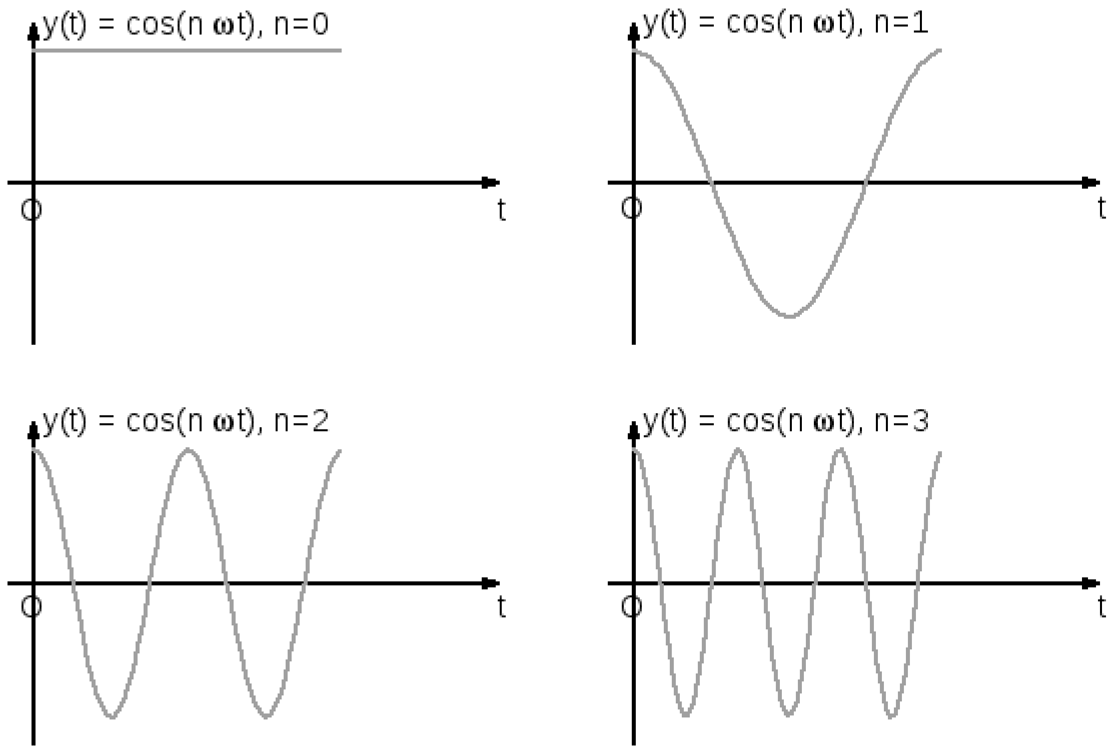
- Wavelength λ
- Period T
- Wave vector $k = 2\pi/\lambda$
- Frequency $\omega = 2\pi/T$
- Phase speed:

$v_\phi = \lambda/T = \omega/\mathrm{k}$



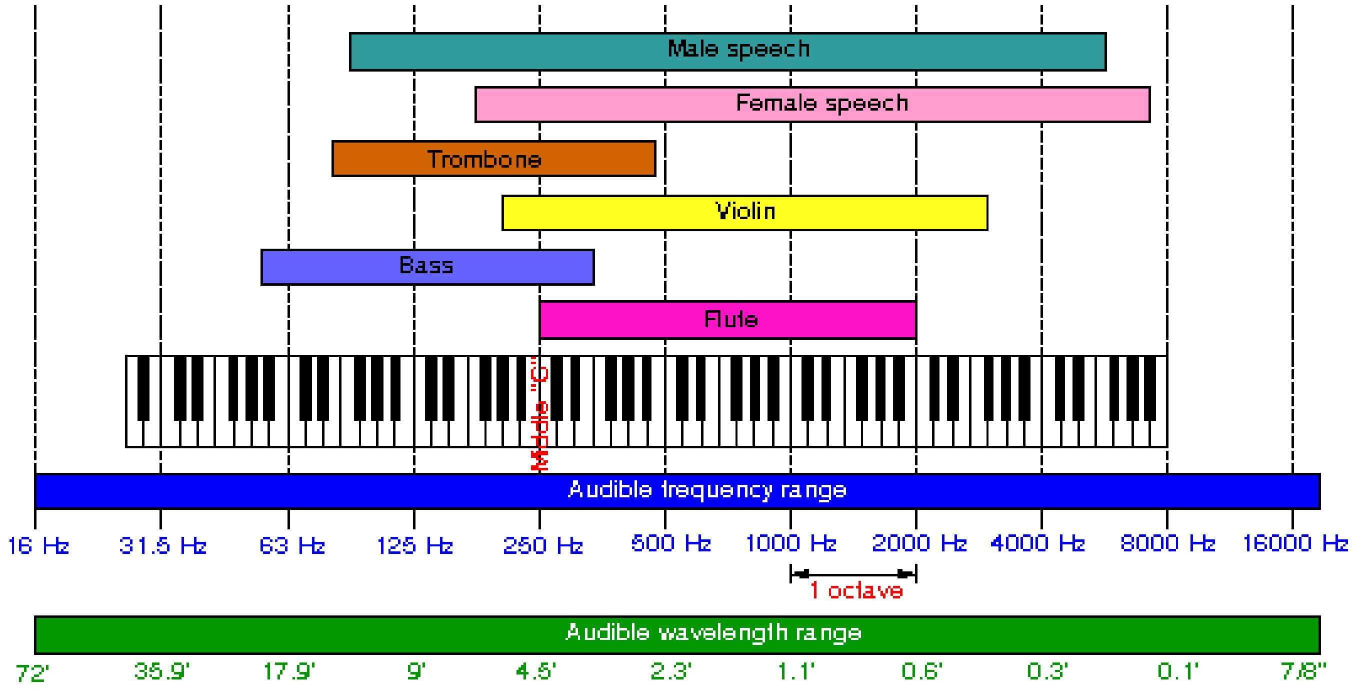
Periodic motions

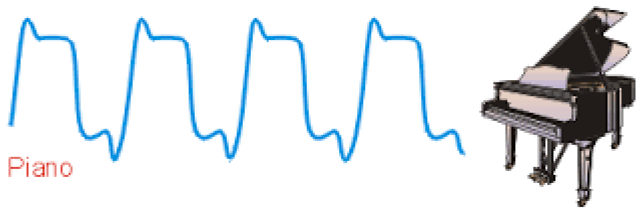
- The **frequency** ω determines the **number of oscillations** in the given period T.

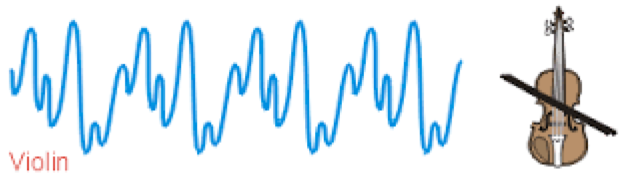


Examples of periodic motions

Sound:

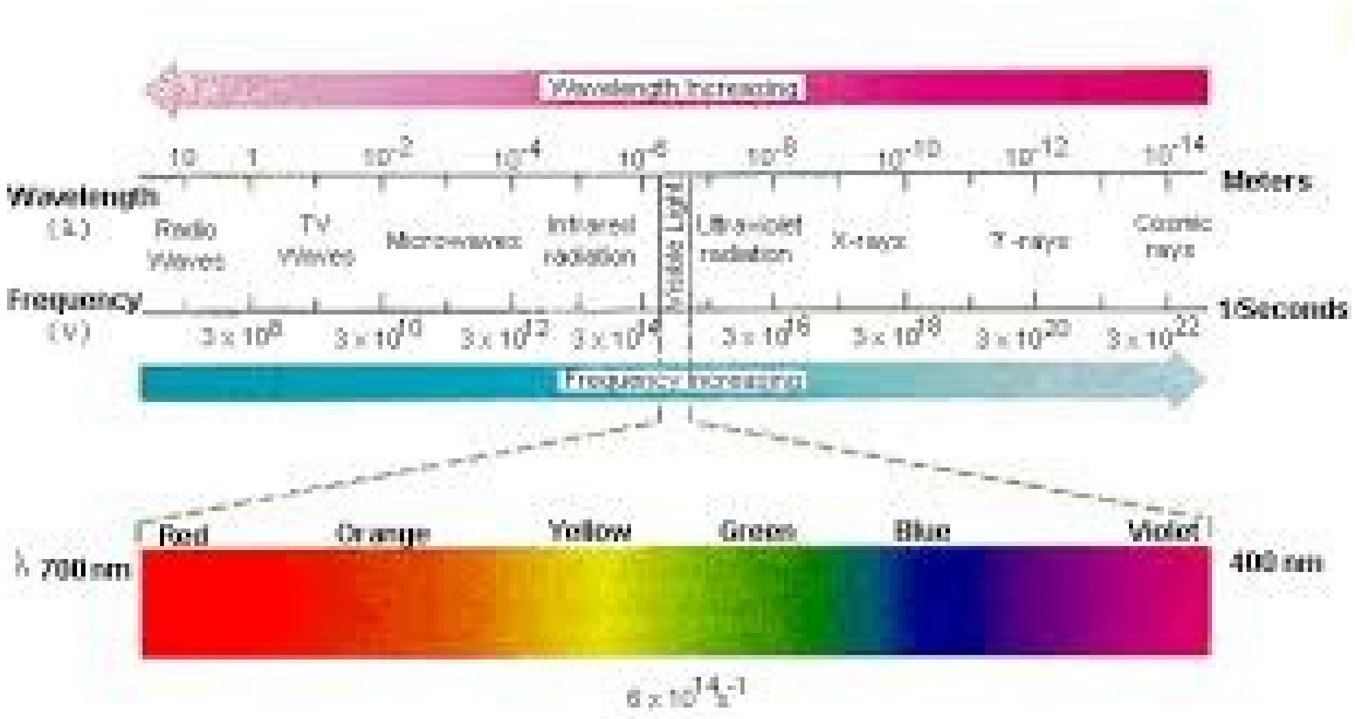


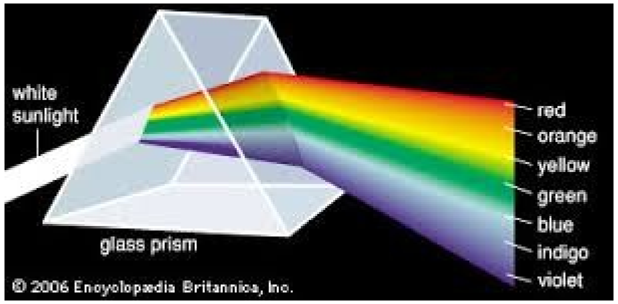




Examples of periodic motions

Light:





Superposition of periodic motions

- The important point in all those examples is that it is **impossible** to find in nature a **pure frequency or wavelenght**.
- Natural signals will always be a **superposition of different frequencies** (or wave vectors).
- Therefore it is natural to try to understand whether it is possible to **separate the different frequencies** (or wave vectors) **in a given signal**, just like a prism does with the different frequencies of the white light!

Superposition of periodic motions

- This has **very important applications** in both **science and technology**:

- Non-linear physics, astrophysics, data analysis, optical fibers, material spectroscopy, etc.;
- Tv and radio broadcasting, transmission of signals, music industry, compression of signals, and many more…

This is where the Fourier analysis comes into play!

Fourier analysis

- Let us suppose to have a **periodic function of time *f(t)*** (but the same holds a function of the position x!), representing a signal that is a superposition of several frequencies and that has the property to be **square integrable**.
- **Fourier's series**:

Given a square integrable function $f(t)$, periodic over a period T, it can be always written as:

$$f(t) = a_0 + \sum_{n=1}^{+\infty} [a_n \cos(n\omega t) + b_n \sin(n\omega t)]$$

Fourier analysis

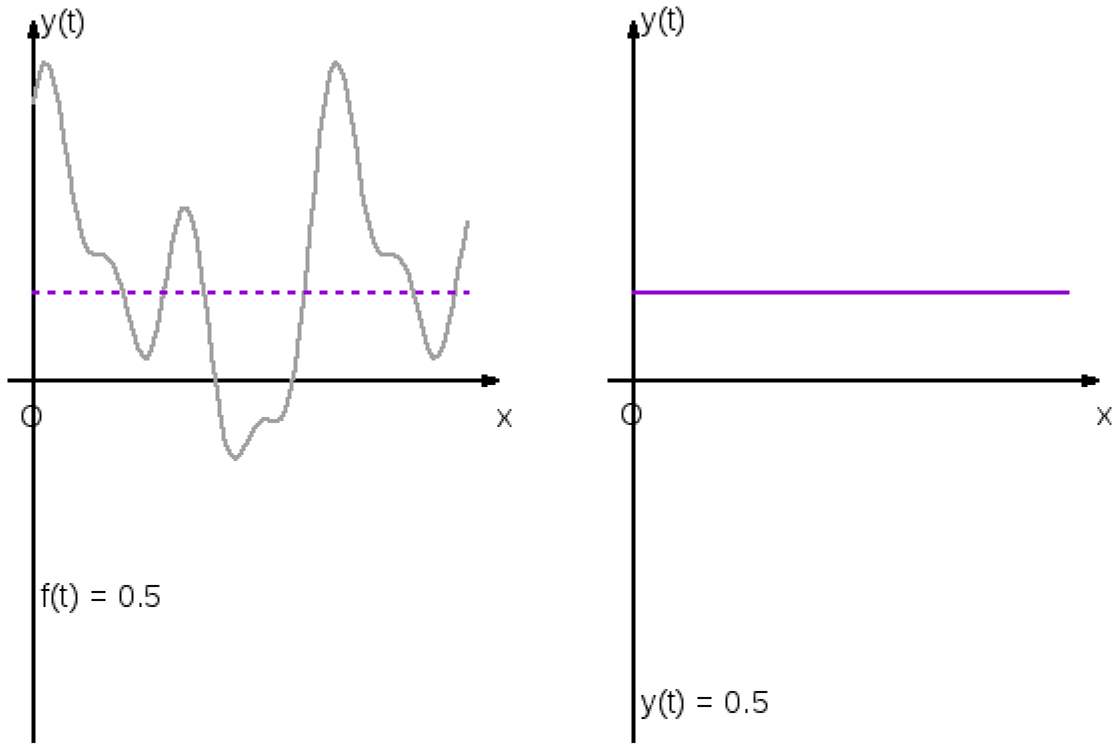
- That is, a **periodic function** $f(t)$ is always made of a **constant part** a_0, plus a superposition of **frequencies multiple** (for an **integer factor** n) of the **fundamental frequency**:

$$\omega = \frac{2\pi}{T}$$

- The components corresponding to the n-th multiple of the fundamental frequency is called the n-th **harmonic** of the series.

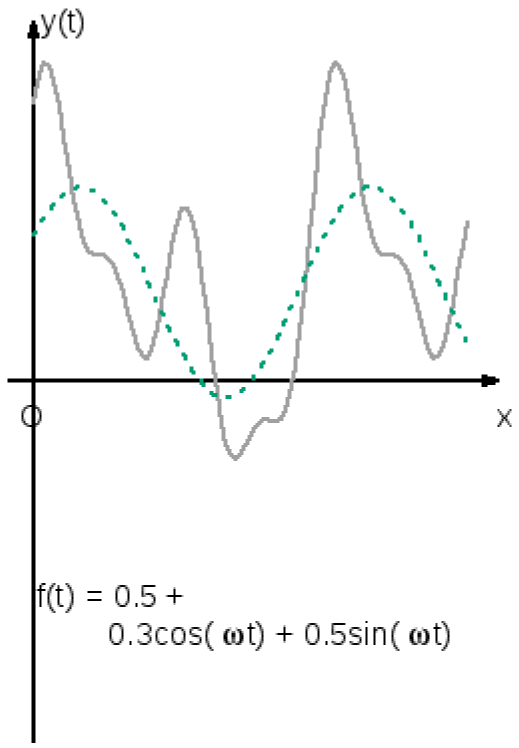
Fourier analysis

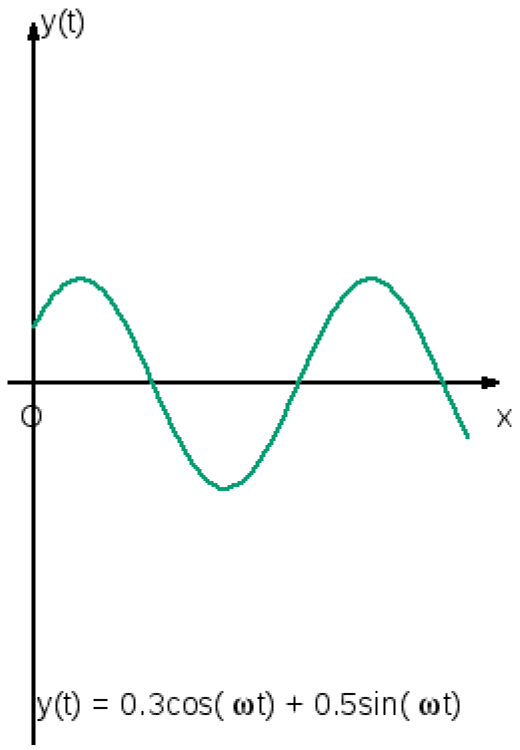
- Example: $f(t)$ and zero-th harmonic:



Fourier analysis

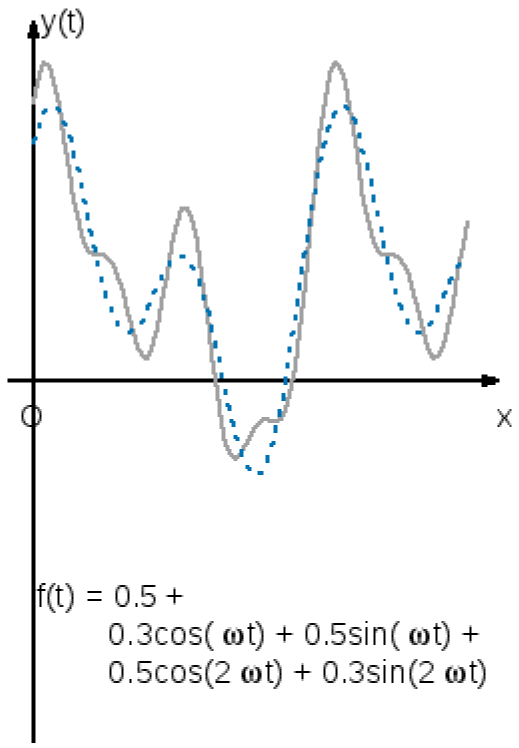
- Example: $f(t)$ and 0+1 harmonics:

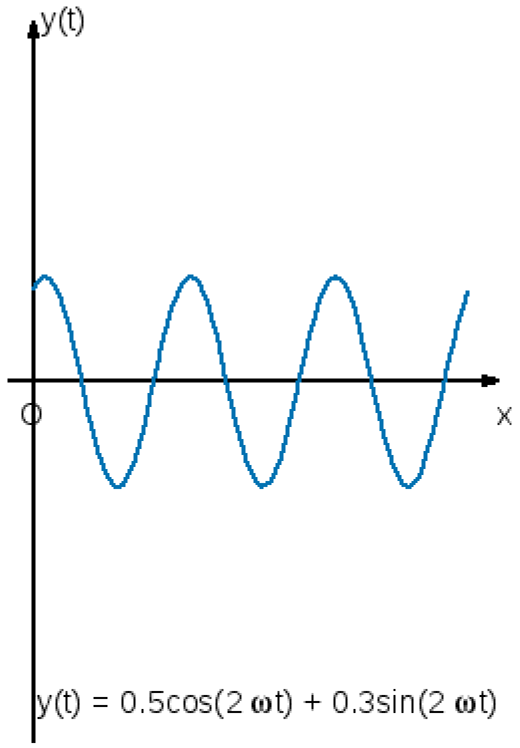




Fourier analysis

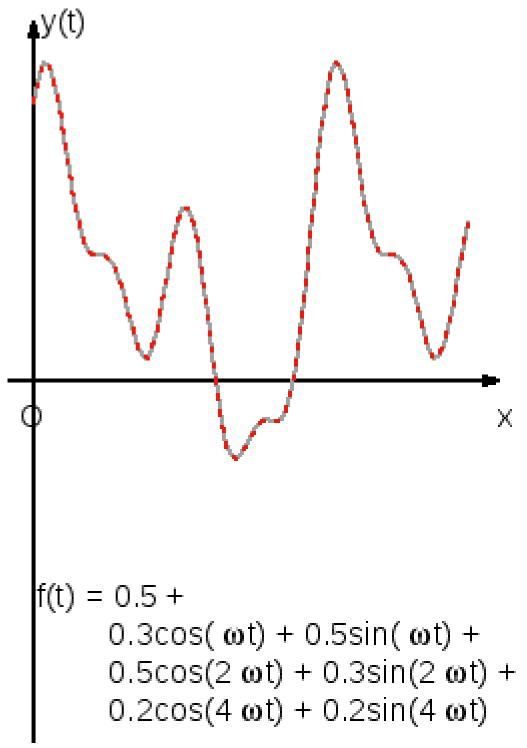
- Example: *f(t)* and 0+1+2 harmonics:

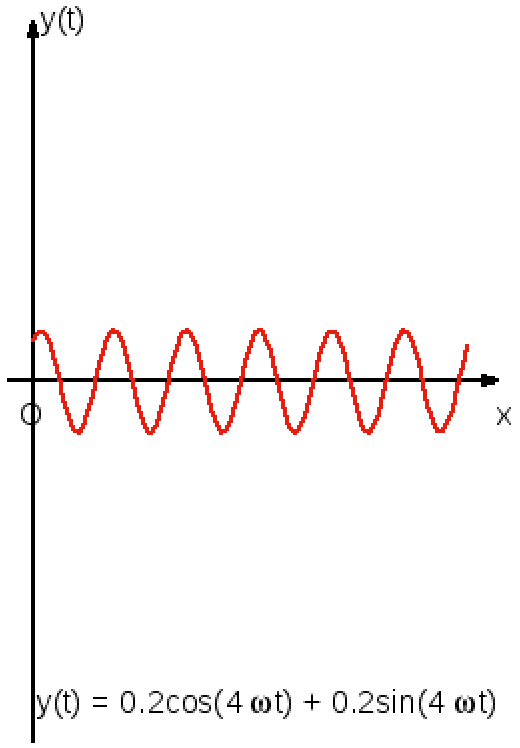




Fourier analysis

- Example: $f(t)$ and 0+1+2+4 harmonics:





Fourier analysis

- Problem: given *f(t)*, how does one **determine the coefficients** a_n, b_n that appear in the series?
- Instead of having to deal with **real numbers**, it is often more convenient to use the so-called **complex representation** for the Fourier's series.
- Euler's relations:

$$\cos x = \frac{e^{ix} + e^{-ix}}{2}; \qquad \sin x = \frac{e^{ix} - e^{-ix}}{2i}$$

Fourier analysis

- They allow us to write:

$$f(t) = a_0 + \sum_{n=1}^{+\infty} a_n \left[\frac{e^{i\omega nt} + e^{-i\omega nt}}{2}\right] + b_n \left[\frac{e^{i\omega nt} - e^{-i\omega nt}}{2i}\right]$$

from which, we obtain:

$$f(t) = a_0 + \sum_{n=1}^{+\infty} \left(\frac{a_n - ib_n}{2}\right) e^{i\omega nt} + \sum_{n=1}^{+\infty} \left(\frac{a_n + ib_n}{2}\right) e^{-i\omega nt}$$

- If we define the quantities:

$$c_0 = a_0; \quad c_n = \frac{a_n - ib_n}{2}; \quad c_{-n} = \frac{a_n + ib_n}{2}, \quad n = 1, \ldots, +\infty$$

Fourier analysis

- By noticing that: $e^{i\omega nt} = 1 \;\; \text{for } n = 0$

  we can write the Fourier's series in a more compact form:

$$f(t) = \sum_{n=-\infty}^{+\infty} c_n e^{i\omega nt}$$

  that is we have re-written the function *f(t)* as a superposition of complex oscillating functions, with complex coefficients c_n, where the integer multiple n of the fundamental frequency takes values both positive and negative (that is, we admit the possibility of having both **positive and negative frequencies**).

Fourier analysis

- Notice however that, although the c_n are **complex**, they must satisfy the condition:

$$c_{-n} = c_n^*$$

  such that the function $f(t)$ is **real**!

- This means that the negative coefficients of the complex Fourier's development are the complex conjugates of the corresponding positive ones. This is called **reality condition**.

- Moreover, we can just easily notice that c_0 must be **real**!

Fourier analysis

- We have seen so far that the **real coefficients** a_n and b_n, or their **complex combinations** c_n, represent the **contribution of the n-th harmonic $n\omega$ to the signal!**
- How do we compute such a contribution?
- Let us start again from the development:

$$f(t) = \sum_{n=-\infty}^{+\infty} c_n e^{i\omega n t}$$

multiply both sides for $e^{-i\omega m t}$ and integrate between 0 and T.

Fourier analysis

- We then get:

$$\int_0^T f(t)e^{-i\omega m t}dt = \int_0^T \sum_{n=-\infty}^{+\infty} c_n e^{i\omega n t} e^{-i\omega m t} dt =$$

$$= \sum_{n=-\infty}^{+\infty} c_n \int_0^T e^{i(n-m)\omega t} dt$$

- Now, we can notice that:

$$\int_0^T e^{i(n-m)\omega t} dt = \frac{1}{i(n-m)\omega} \int_0^{i(n-m)\omega T} e^z dz =$$

$$\frac{1}{i(n-m)\omega}\left[e^{i(n-m)\omega T} - 1\right] = 0 \quad \text{if } m \neq n$$

Fourier analysis

where we used the substitution:

$$z = i(n-m)\omega t; \qquad dz = i(n-m)\omega dt$$

and the fact that:

$$e^{ik\omega T} = \exp\left(ik\frac{2\pi}{T}T\right) = e^{2\pi ik} = 1$$

if k=n-m is an integer!

- Instead, for n=m, we get:

$$\int_0^T e^{i(n-m)\omega t}dt = \int_0^T dt = T$$

Fourier analysis

- From that, we infer:

$$\int_0^T e^{i(n-m)\omega t} dt = T\delta_{nm}$$

- By substituting in the original formula:

$$\int_0^T f(t) e^{-i\omega m t} dt = \sum_{n=-\infty}^{+\infty} c_n \int_0^T e^{i(n-m)\omega t} dt =$$

$$= \sum_{n=-\infty}^{+\infty} c_n T \delta_{nm} = T c_m$$

Fourier analysis

- That is, the Fourier's coefficient c_n can be calculated by multiplying $f(t)$ for the factor $e^{-i\omega mt}$, integrating on the periodicity interval [0,T], then dividing by T :

$$c_n = \frac{1}{T}\int_0^T f(t)e^{-i\omega nt}dt$$

- In this way, we can compute the c_n by calculating an ensemble of infinite integrals. We can notice that:

$$c_0 = \frac{1}{T}\int_0^T f(t)dt \text{ that is, } \; c_0 \textbf{ is the average of } f(t)$$

Parceval's theorem

- Sometimes, instead of computing the Fourier's coefficients c_n, it is more meaningful to compute the contribution that the n-th harmonic gives to the total **energy of the signal**! We mean with that the quantity:

$$E = \int_0^T |f(t)|^2 dt$$

- Parceval's theorem states that:

$$E = \int_0^T |f(t)|^2 dt = T\left(c_0^2 + 2\sum_{n=1}^{+\infty} |c_n|^2\right)$$

Parceval's theorem

- Proof:

by substituting to *f(t)* its Fourier's development:

$$E = \int_0^T |f(t)|^2 dt = \int_0^T f^*(t) f(t) dt =$$

$$= \int_0^T \left(\sum_{m=-\infty}^{+\infty} c_m^* e^{-i\omega m t} \right) \left(\sum_{n=-\infty}^{+\infty} c_n e^{+i\omega n t} \right) dt =$$

$$= \int_0^T \left(\sum_{n=-\infty}^{+\infty} \sum_{m=-\infty}^{+\infty} c_m^* c_n e^{i(n-m)\omega t} \right) dt =$$

$$= \sum_{n=-\infty}^{+\infty} \sum_{m=-\infty}^{+\infty} c_m^* c_n \int_0^T e^{i(n-m)\omega t} dt$$

Parceval's theorem

- By remembering that:

$$\int_0^T e^{i(n-m)\omega t}dt = T\delta_{nm}$$

we get the result:

$$E = \sum_{n=-\infty}^{+\infty}\sum_{m=-\infty}^{+\infty} c_m^* c_n T\delta_{nm} = T\sum_{n=-\infty}^{+\infty} c_n c_n^* = T\sum_{n=-\infty}^{+\infty} |c_n|^2$$

- We can now split the sum in three parts, one relative to negative n, another for $n=0$, the last for positive n.

Parceval’s theorem

$$E = T\left(\sum_{n=-\infty}^{-1} |c_n|^2 + |c_0|^2 + \sum_{n=1}^{+\infty} |c_n|^2\right)$$

- And, if we remember that $c_{-n}=c^*_n$, that is:

$$|c_n|^2 = c_n^* c_n = (c^*_{-n})^* c^*_{-n} = c_{-n} c^*_{-n} = |c_{-n}|^2$$

we finally get:

$$E = \int_0^T |f(t)|^2 dt = T\left(c_0^2 + 2\sum_{n=1}^{+\infty} |c_n|^2\right)$$

as it was to demonstrate.

Parceval's theorem

- This means that the n-th harmonics $n\omega$ contributes to the total energy of the signal with a term equal to: $2T|c_n|^2$ for any n different from 0, and with a term equal to: Tc_0^2, for $n=0$!
- We define the function:

$$E(n) = \begin{cases} Tc_0^2 & \text{per } n = 0; \\ 2T|c_n|^2 & \text{per } n = 1, \ldots, +\infty \end{cases}$$

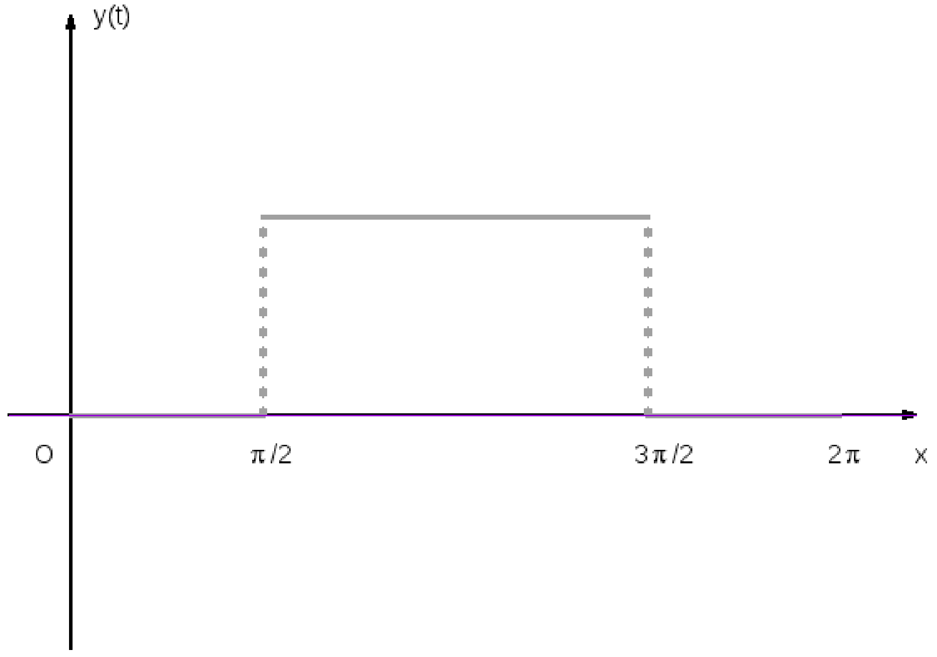
called **energy spectrum** of the signal, so that:

$$E = \sum_{n=0}^{\infty} E(n)$$

Pulse signal

- As an example, let us consider a rectangular pulse:

$$f(t) = \begin{cases} 0 & \text{for } 0 \le t < \pi/2 \\ 1 & \text{for } \pi/2 \le t \le 3\pi/2 \\ 0 & \text{for } 3\pi/2 < t \le 2\pi \end{cases}$$



Pulse signal

- We have, for the Fourier's coefficients c_n:

$$c_n = \frac{1}{T}\int_0^T f(t)e^{-in\omega t}dt = \frac{1}{2\pi}\int_0^{2\pi} f(t)e^{-int}dt =$$

$$= \frac{1}{2\pi}\left[\lim_{\epsilon\to 0}\int_0^{\pi/2-\epsilon} f(t)e^{-int}dt + \int_{\pi/2}^{3\pi/2} f(t)e^{-int}dt + \right.$$

$$\left. + \lim_{\epsilon\to 0}\int_{3\pi/2-\epsilon}^{2\pi} f(t)e^{-int}dt\right] = \frac{1}{2\pi}\int_{\pi/2}^{3\pi/2} e^{-int}dt$$

because:

$$T = 2\pi \qquad \text{and} \qquad \omega = \frac{2\pi}{T} = 1$$

Pulse signal

- By operating the substitution:

$$int = z; \qquad dt = \frac{1}{in} dz$$

we find:

$$c_n = \frac{1}{2in\pi} \int_{in\pi/2}^{3in\pi/2} e^{-z} dz = -\frac{1}{2in\pi} \left[e^{-z} \right] = -\frac{1}{2in\pi} \left(e^{-3in\pi/2} - e^{-in\pi/2} \right)$$

- By remembering Euler's relations:

$$e^{-3in\pi/2} = \cos\left(\frac{3n\pi}{2}\right) - i \sin\left(\frac{3n\pi}{2}\right)$$

Pulse signal

- We find:

$$e^{-3in\pi/2} = \begin{cases} -1 - i \cdot 0 = -1 & n = 2, 6, \ldots, 2(2k+1) \\ +1 - i \cdot 0 = +1 & n = 4, 8, \ldots, 2(2k) \\ 0 - +i = +i & n = 1, 5, \ldots, 2(2k) + 1 \\ 0 - i = -i & n = 3, 7, \ldots, 2(2k+1) + 1 \end{cases}$$

for k = 0, 1, 2, 3,… and:

$$\mathrm{e}^{-in\pi/2} = \begin{cases} -1 + -i \cdot 0 = -1 & n = 2, 6, \ldots, 2(2k+1) \\ +1 - i \cdot 0 = +1 & n = 4, 8, \ldots, 2(2k) \\ 0 - i = -i & n = 1, 5, \ldots, 2(2k) + 1 \\ 0 + i = +i & n = 3, 7, \ldots, 2(2k+1) + 1 \end{cases}$$

Pulse signal

- Then, by subtracting the two formulas term by term:

$$c_n = \frac{1}{2in\pi}\begin{cases} 0 & \text{for even } n \\ +2i & \text{for } n = 1, 5, \dots \\ -2i & \text{for } n = 3, 7, \dots \end{cases} = \begin{cases} 0 & \text{for even } \; n \\ \frac{1}{n\pi}(-1)^{\frac{n-1}{2}} & \text{for odd } \; n \end{cases}$$

for $n \neq 0$, while, for $n = 0$, we have:

$$c_0 = \frac{1}{2\pi}\int_0^{2\pi} f(t)dt = \frac{1}{2\pi}\int_{\pi/2}^{3\pi/2} dt = \frac{1}{2\pi}\left[t\right]_{\pi/2}^{3\pi/2} = \frac{1}{2}$$

Pulse signal

- That is, the energy spectrum is given by:

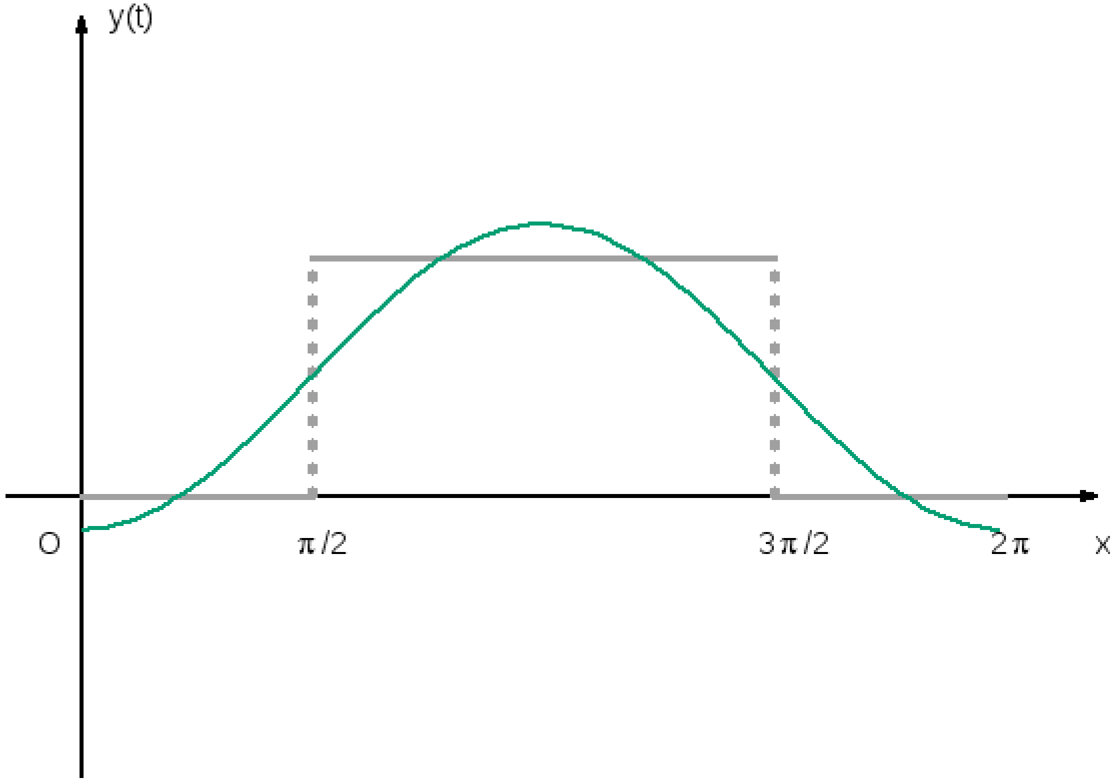
$$E(n) = \begin{cases} 2\pi c_0^2 = \pi/2 & \text{for } n = 0; \\ 0 & \text{for even } n; \\ 2 \cdot 2\pi |c_n|^2 = \frac{4}{\pi} n^{-2} & \text{for odd } n. \end{cases}$$

  which corresponds to a **power-law spectrum** in n^{-2}!

- We may now reconstruct the function by visualizing the contribution of the first N harmonics...

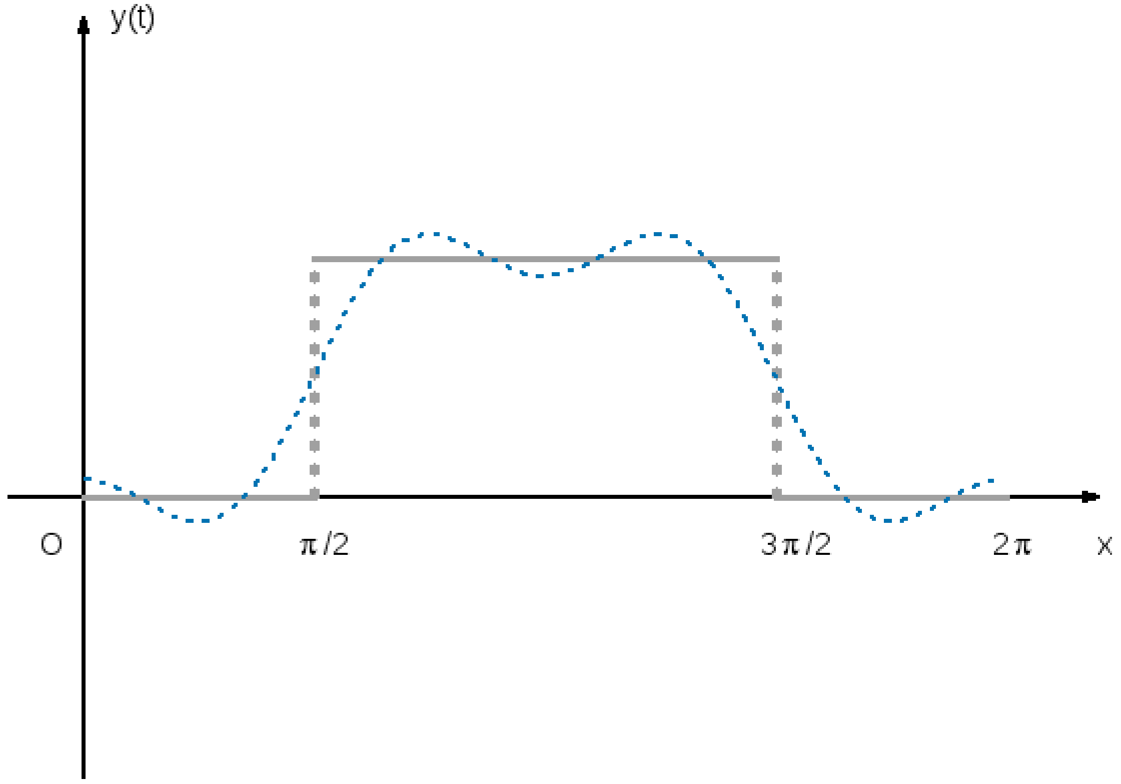
Pulse signal

$$f_N(t) = \sum_{n=-N}^{N} c_n e^{int} \qquad \text{for } N = 1$$



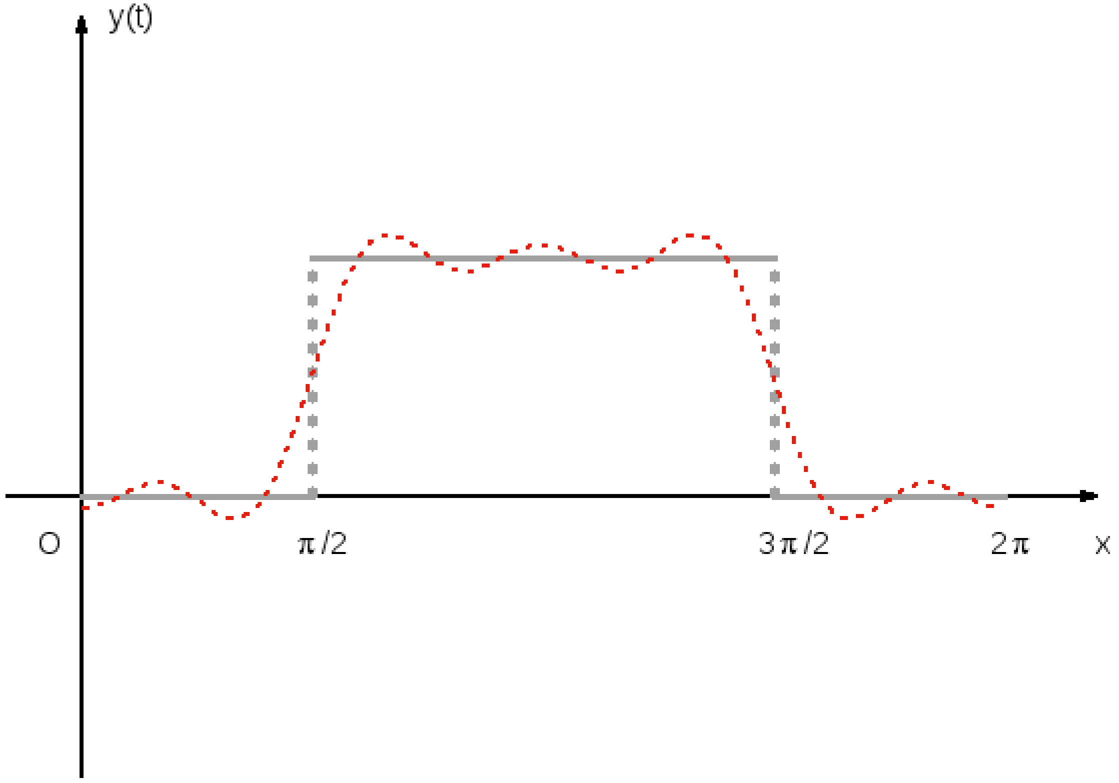
Pulse signal

$$f_N(t) = \sum_{n=-N}^{N} c_n e^{int} \qquad \text{for } N = 3$$



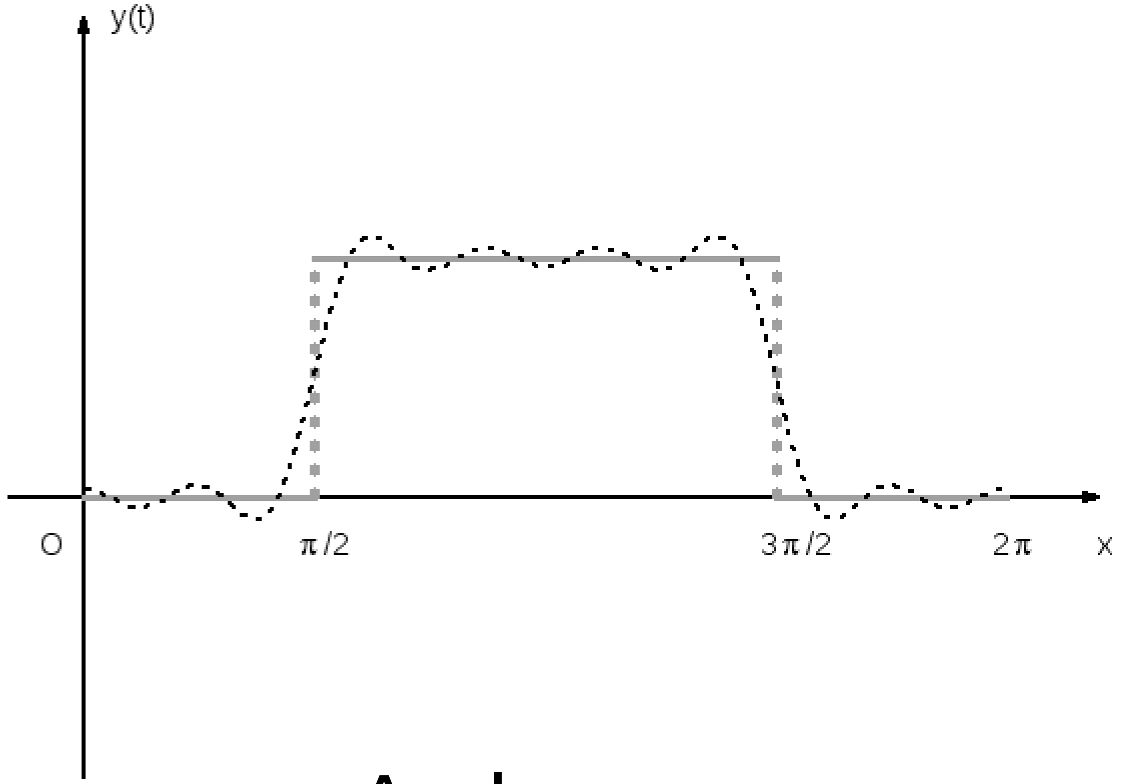
Pulse signal

$$f_N(t) = \sum_{n=-N}^{N} c_n e^{int} \qquad \text{for } N = 5$$



Pulse signal

$$f_N(t) = \sum_{n=-N}^{N} c_n e^{int} \qquad \text{for } N = 7$$



And so on...

Discrete Fourier transforms (DFT)

- All we said till now is valid for a function known in a given **periodicity subinterval of the real set**.
- However, when we want to analyze **“real” data** (coming from a measuring instrument), we have to deal with some **“sampled”** data, that is, the signal is sampled at some **“discrete”** times:

$$t_j = j\Delta t, \text{ with } j = 0, \ldots, N \text{ and } \Delta t = T/N.$$

Discrete Fourier transforms (DFT)

- **Sampling theorem (or Nyquist-Shannon's theorem)**:

Given the Fourier development of a given periodic function $f(t_j)$, sampled on N discrete points, as:

$$f(t_j) = \sum_{n=-\infty}^{+\infty} c_n e^{in\omega t_j} \qquad j = 0, \ldots, N-1$$

not all of the values n are independent, but only the ones in the interval: $-N_{max} \leq n \leq N_{max}$

*where N_{max} = $N/2$. N_{max} is said "**Nyquist number**" and the corresponding frequency:*

$\omega_{max} = N_{max}\omega$ "***Nyquist frequency***".

Discrete Fourier transforms (DFT)

- What is the meaning of the theorem?

  If I have a **discrete signal** that contains a frequency equal to, for instance:

  $$n\omega = (N_{max} + m)\omega$$

  where m is a given integer number, when I **sample the signal** on N equally spaced points the frequency $n\omega$ is **indistinguishable** from a frequency: $n'\omega = (-N_{max} + m)\omega$

- Then, any **positive frequency** greater than N_{max} is mapped into a **negative frequency**.

Discrete Fourier transforms (DFT)

- In fact:

$$e^{in\omega t_j} = e^{i(N_{max}+m)\omega t_j} = e^{iN_{max}\omega t_j} e^{im\omega t_j}$$

and, by recalling that:

$$N_{max} = \frac{N}{2}, \ \omega = \frac{2\pi}{T} \quad \text{and} \quad t_j = j\Delta t = \frac{jT}{N}$$

we have:

$$e^{in\omega t_j} = e^{i\frac{N}{2}\omega t_j} e^{im\omega t_j} = e^{i(N-\frac{N}{2})\frac{2\pi}{T}\frac{jT}{N}} e^{im\omega t_j} =$$

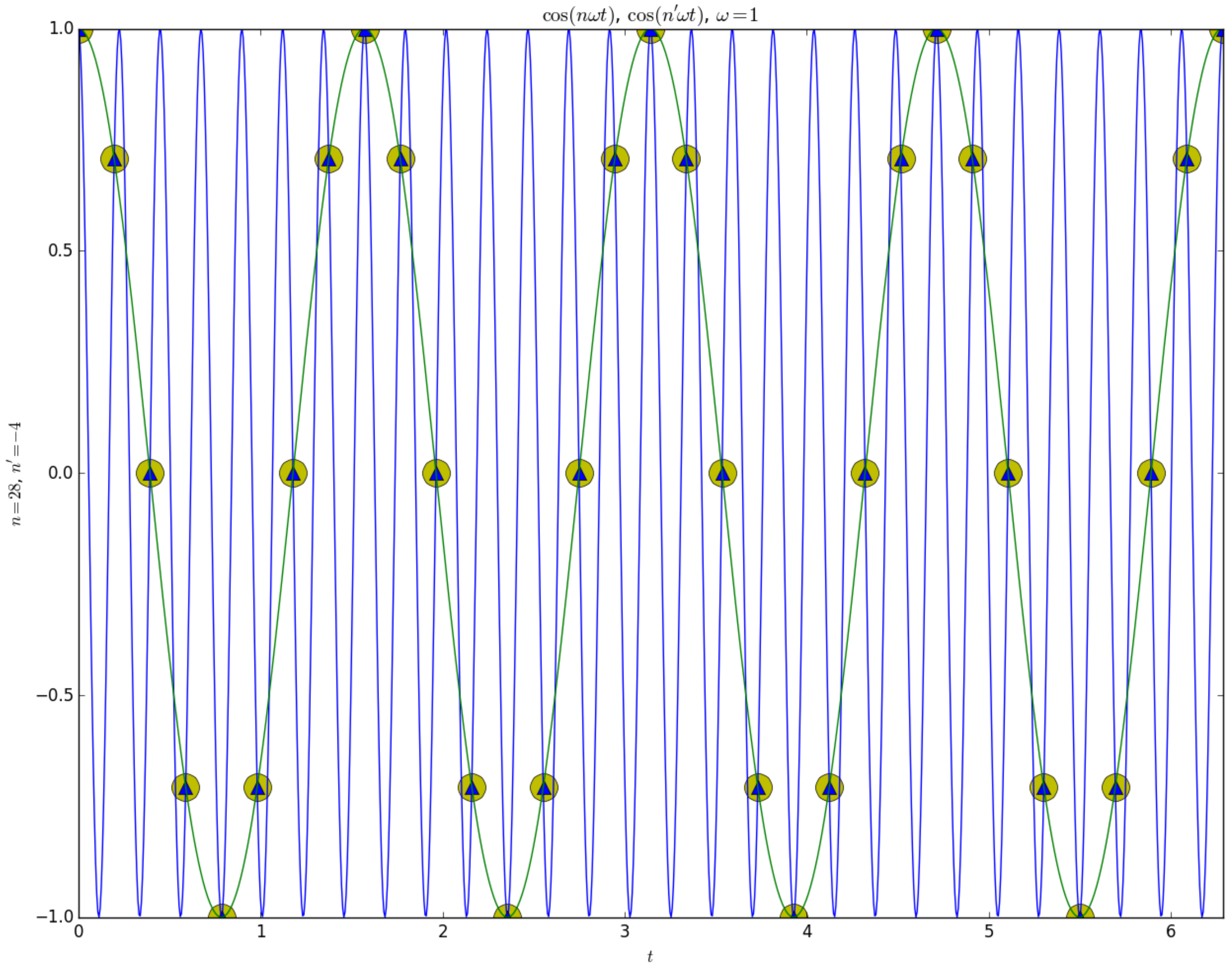
$$= \underbrace{e^{i2\pi j}}_{=1} e^{-i\frac{N}{2}\omega t_j} e^{im\omega t_j} = e^{i(m-N_{max})\omega t_j}$$

Discrete Fourier transforms (DFT)

- This means, in fact, that, when sampling on a discrete number of points, oscillations corresponding to a multiple: $n=N_{max}+m$ of the fundamental frequency ω, pass through **the same points** as oscillations corresponding to a multiple: $n'=-N_{max}+m$ of the fundamental frequency.

- In the following example, $N=32$, $m=12$, which gives: $n=N_{max}+m=28$, which passes through the same points (marked with blue triangles in the plot) as $n'=-N_{max}+m=-4$ (marked with yellow big dots in the plot).

Discrete Fourier transforms (DFT)



Discrete Fourier transforms (DFT)

- The same problem exists when you try to sample a (negative) frequency $n\omega=-(N_{max}+m)\omega$ which is mapped into a (positive) frequency: $n'\omega=(N_{max}-m)\omega$.
- This problem is known under the name **"aliasing"** and it is a consequence of the fact that the signal is **sampled**.
- For instance, to **master a CD** the audio signal is sampled with a sampling rate of **44.1 kHz**, because the human hearing range is (roughly) between 20-20 kHz (however, aliasing problems can persist for a trained ear, like the one of a musician!).

Discrete Fourier transforms (DFT)

- Therefore, in the discrete Fourier's series of a function $f(t_j)$, sampled on N points, it makes sense to consider **only** the frequencies between the negative and positive **Nyquist frequencies**, that is the values of n in the interval $[-N/2,+N/2]$:

$$f(t_j) = \sum_{n=-N_{max}}^{+N_{max}} c_n e^{in\omega t_j}$$

Discrete Fourier transforms (DFT)

- Analogously, for the Parceval's theorem, we have:

$$\sum_{n=0}^{N} |f(t_j)|^2 = T\left(c_0^2 + 2\sum_{n=1}^{N_{max}} |c_n|^2\right)$$

- Therefore, now the problem is to finally find a method to **compute the coefficients** c_n for a function $f(t)$, periodic on an interval T, sampled on N points $t_j = jT/N$.

Discrete Fourier transforms (DFT)

- According to what we found, computing the c_n is equivalent to compute the integrals:

$$c_n = \frac{1}{T}\int_0^T f(t)e^{-in\omega t}dt$$

where $f(t)$ is sampled on the discrete points t_j.

- This can be easily made by using the methods we have studied for solving integrals of functions known on equally spaced intervals h, for instance the **trapezoidal rule**!

Discrete Fourier transforms (DFT)

- For instance, we can write:

$$c_n = \frac{1}{T}\int_0^T f(t)e^{-in\omega t}dt = \frac{1}{T}\left[\int_0^T F_n(t)dt - i\int_0^T G_n(t)dt\right]$$

where:

$$F_n(t) = f(t)\cos(n\omega t);$$
$$G_n(t) = f(t)\sin(n\omega t).$$

are **real**, **periodic**, functions of t, known on the sampling points t_j.

Discrete Fourier transforms (DFT)

- According to the trapezoidal rule, their integrals can be approximated as:

$$\int_0^T F_n(t)dt \simeq \Delta t \left[\frac{F_n(t_0) + F_n(t_N)}{2} + \sum_{j=1}^{N-1} F_n(t_j) \right];$$

$$\int_0^T G_n(t)dt \simeq \Delta t \left[\frac{G_n(t_0) + G_n(t_N)}{2} + \sum_{j=1}^{N-1} G_n(t_j) \right].$$

- Since both F_n and G_n are periodic functions, this means that: $F_n(t_0)=F_n(t_N)$ and $G_n(t_0)=G_n(t_N)$.

Discrete Fourier transforms (DFT)

- Hence, we find the approximation of the coefficients c_n as:

$$c_n \simeq \frac{\Delta t}{T}\left[\sum_{j=0}^{N-1} F_n(t_j) - i\sum_{j=0}^{N-1} G_n(t_j)\right]$$

that is, by remembering the definition of F_n and G_n and the fact that $T=N\Delta t$, we have:

$$c_n \simeq \frac{1}{N}\left[\sum_{j=0}^{N-1} f(t_j)\cos(n\omega t_j) - i\sum_{j=0}^{N-1} f(t_j)\sin(n\omega t_j)\right]$$

Discrete Fourier transforms (DFT)

- We have know two basic questions to answer:

1) What is the error introduced by the trapezoidal rule?

2) How many operations are required in order to compute the DFT of the function $f(t)$, sampled on N grid-points?

- Both questions are easy to answer!
- Error of the trapezoidal rule:

$$\mathcal{E} = -\frac{1}{12}h^2(b-a)\bar{f''} - \frac{1}{24}h^3(b-a)\bar{f'''} - \frac{1}{80}h^4(b-a)f^{\bar{IV}} + O(h^4)$$

Discrete Fourier transforms (DFT)

- It is easy to show that, since $f(t)$ is a periodic function, **the averages of all its derivatives**, starting from the first, **are exactly zero**!
- This is because $f(t)$ can be expanded as a Fourier series, namely the sum of infinite terms of the kind: $\cos(n\omega t)$ and $\sin(n\omega t)$, whose derivatives are always functions of the same kind.
- But the averages of $\cos(n\omega t)$ and $\sin(n\omega t)$ are identically zero, when computed on a period $[0,T]$!
- Therefore all error terms are zero and the trapezoidal rule give the **exact coefficients** c_n!

Discrete Fourier transforms (DFT)

- Concerning the number of operations required to compute the c_n-s, they are easily estimated:

1) For the reality condition: $c_{-n}=c_n^*$, and the Nyquist-Shannon theorem, we need to compute only the c_n for $n=0,\ldots,N/2$.

2) The total number of coefficients (real and imaginary part) is made of $2\text{x}(N/2+1)$ numbers.

3) We need to remember that actually c_0 is real, so its imaginary part need not to be computed!

Discrete Fourier transforms (DFT)

4) It can be easily shown that the coefficient $c_{N/2}$ is purely real, in turn:

$$c_{N/2} = \frac{1}{N}\left[\sum_{j=0}^{N-1} f(t_j)\cos\left(\frac{N}{2}\omega t_j\right) - i\sum_{j=0}^{N-1} f(t_j)\sin\left(\frac{N}{2}\omega t_j\right)\right] =$$

$$= \frac{1}{N}\left[\sum_{j=0}^{N-1} f(t_j)\cos\left(\frac{N}{2}\frac{2\pi}{T}\frac{jT}{N}\right) - i\sum_{j=0}^{N-1} f(t_j)\sin\left(\frac{N}{2}\frac{2\pi}{T}\frac{jT}{N}\right)\right] =$$

$$= \frac{1}{N}\left[\sum_{j=0}^{N-1} f(t_j)\cos(j\pi) - i\sum_{j=0}^{N-1} f(t_j)\sin(j\pi)\right] =$$

$$= \frac{1}{N}\left[\sum_{j=0}^{N-1} (-1)^j f(t_j)\right]$$

Discrete Fourier transforms (DFT)

5) The total number of coefficients to compute is therefore: $2\text{x}(N/2+1)-2 = N$.

6) For each coefficient we have to compute the product $f(t_j)\cos(n\omega t_j)$ (for the real parts) or $f(t_j)\sin(n\omega t_j)$ (for the imaginary parts), for

$j=0,\ldots,N-1$, that is N products.

Hence, the total number of operations is N^2!

Fast Fourier Transform (FFT)

- Is it possible to **decrease the number of operations**?
- The anwer is **positive**, due to an algorithm called **FFT (Fast Fourier Transform)** proposed by J.W. **Cooley** and J.W. **Tukey** [Math. Comp., 19, 1965].
- Although the algorithm was officially proposed in 1965, it is based on a lemma due to **Danielson and Lanczos (1942)** and FFT-like methods date back even to **Gauss (1805)**!!!

Fast Fourier Transform (FFT)

- Let’s restart from the DFT:

$$c_n = \frac{1}{N}\left[\sum_{j=0}^{N-1} f(t_j)\cos(n\omega t_j) - i\sum_{j=0}^{N-1} f(t_j)\sin(n\omega t_j)\right] =$$

$$= \frac{1}{N}\left[\sum_{j=0}^{N-1} f(t_j)e^{-in\omega t_j}\right] = \frac{1}{N}\left[\sum_{j=0}^{N-1} f(t_j)e^{-in\frac{2\pi}{T}\frac{jT}{N}}\right] =$$

$$= \frac{1}{N}\left[\sum_{j=0}^{N-1} f(t_j)e^{-i\frac{2\pi nj}{N}}\right] = \frac{1}{N}\left[\sum_{j=0}^{N-1} f(t_j)W_N^{nj}\right]$$

Fast Fourier Transform (FFT)

Where we pose: $W_N = e^{-\frac{2\pi i}{N}}$

- We may now observe that $n = 0, \ldots, N/2$ and, for each value of n we have to compute N complex multiplications of the real values $f(t_j)$ by the complex values W_N^{nj}, for a total of $(N/2 + 1)\text{x}(2N) = N^2 + 2N$ multiplications.
- The **Danielson and Lanczos lemma** states that the **single transform on N points**, namely the computation of the c_n can be re-written as **two separated transforms on N/2 points!**

Fast Fourier Transform (FFT)

- In fact:

$$c_n = \frac{1}{N}\sum_{j=0}^{N-1} f(t_j) W_N^{nj} =$$

$$= \frac{1}{N}\left[\sum_{k=0}^{\frac{N}{2}-1} f(t_{2k}) W_N^{n(2k)} + \sum_{k=0}^{\frac{N}{2}-1} f(t_{2k+1}) W_N^{n(2k+1)}\right] =$$

$$= \frac{1}{N}\left[\sum_{k=0}^{\frac{N}{2}-1} f(t_{2k}) (W_N^2)^{nk} + W_N^n \sum_{k=0}^{\frac{N}{2}-1} f(t_{2k+1}) (W_N^2)^{nk}\right]$$

Fast Fourier Transform (FFT)

- We now notice that:

$$W_N^2 = e^{-\frac{2\pi i}{N}\cdot 2} = e^{-\frac{2\pi i}{N/2}} = W_{N/2}$$

and, therefore:

$$c_n = \frac{1}{N}\left[\sum_{k=0}^{\frac{N}{2}-1} f(t_{2k}) W_{N/2}^{nk} + W_N^n \sum_{k=0}^{\frac{N}{2}-1} f(t_{2k+1}) W_{N/2}^{nk}\right]$$

that is, we have re-written the **transform on *N* points** as a combination, with coefficients W_N^n of 2 transforms on $N/2$ points!

Fast Fourier Transform (FFT)

- The main point in all this consists in the fact that 1 single transform on N points requires (approximately!) N^2 multiplications. For the same reason, 1 single transform on $N/2$ points will require $(N/2)^2 = N^2/4$ multiplications!
- Hence, through the Danielson-Lanczos lemma, we may compute the $N/2$ values of c_n in: $2\mathrm{x}(N^2/4)+N=$**$N^2/2+N$** multiplications, instead of **N^2**. Therefore, after this first subdivision, **we gained a factor 2 in the number of operations!**

Fast Fourier Transform (FFT)

- Now we notice that, provided N **is a power of two**: $N = 2^m$, we may repeat the procedure by dividing the transform on the even points in the **even and odd values of j**, and the same for the odd points, by rewriting the **original transform on N points** into **4 transforms on $N/4$ points**!
- This means $4 \text{x} (N/4)^2 = N^2/4$ multiplications, instead of N^2, so **we gained another factor 2 in the number of operations**!

Fast Fourier Transform (FFT)

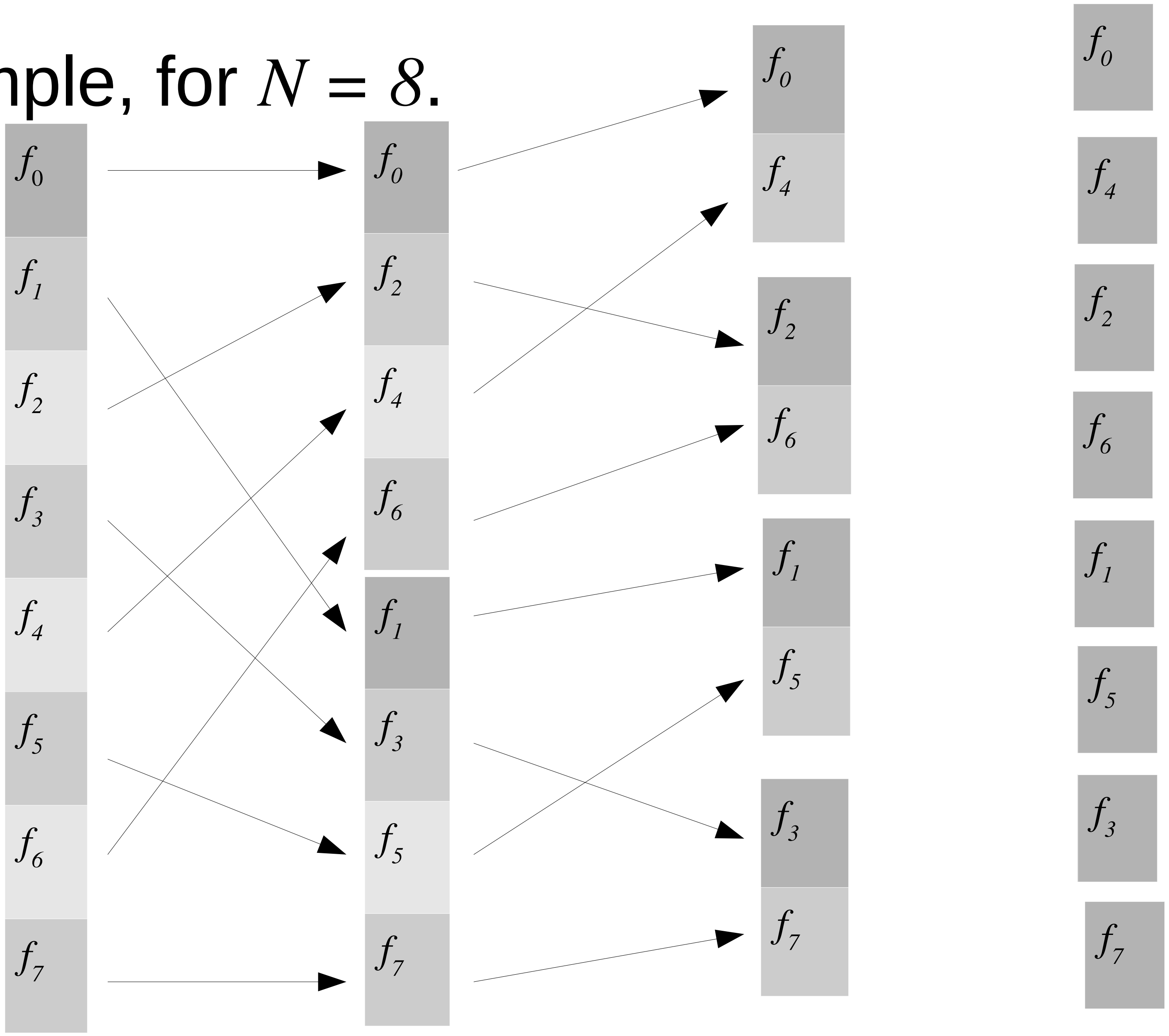
- We can continue this procedure of dividing m times, so that at the end we will have ***N*** **transforms on one single point!**
- The transform on a single point is just the functions itself, as it may be seen from the definition:

$$c_n = \frac{1}{N}\sum_{j=0}^{N-1} f(t_j) W_N^{nj} = \frac{1}{N} f(t_0) \text{ for } N = 1$$

therefore we will have a number of operations equal to: m x N = $log_2 N$ x N !

Fast Fourier Transform (FFT)

- Example, for $N = 8$.



Fast Fourier Transform (FFT)

- The correct order of the couplings is given by the order of the bit-reversed positions:

Original order	Original order in binary digits	Order in reversed binary digits	FFT combined order
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Fast Fourier Transform (FFT)

- It is possible to apply the same arguments to the computation of the original function $f(t)$, given its Fourier's coefficients c_n (**Inverse FFT**).
- By definition:

$$f(t_j) = \sum_{n=-N/2}^{+N/2} c_n e^{in\omega t_j} = \sum_{n=-N/2}^{+N/2} c_n W_N^{-nj}$$

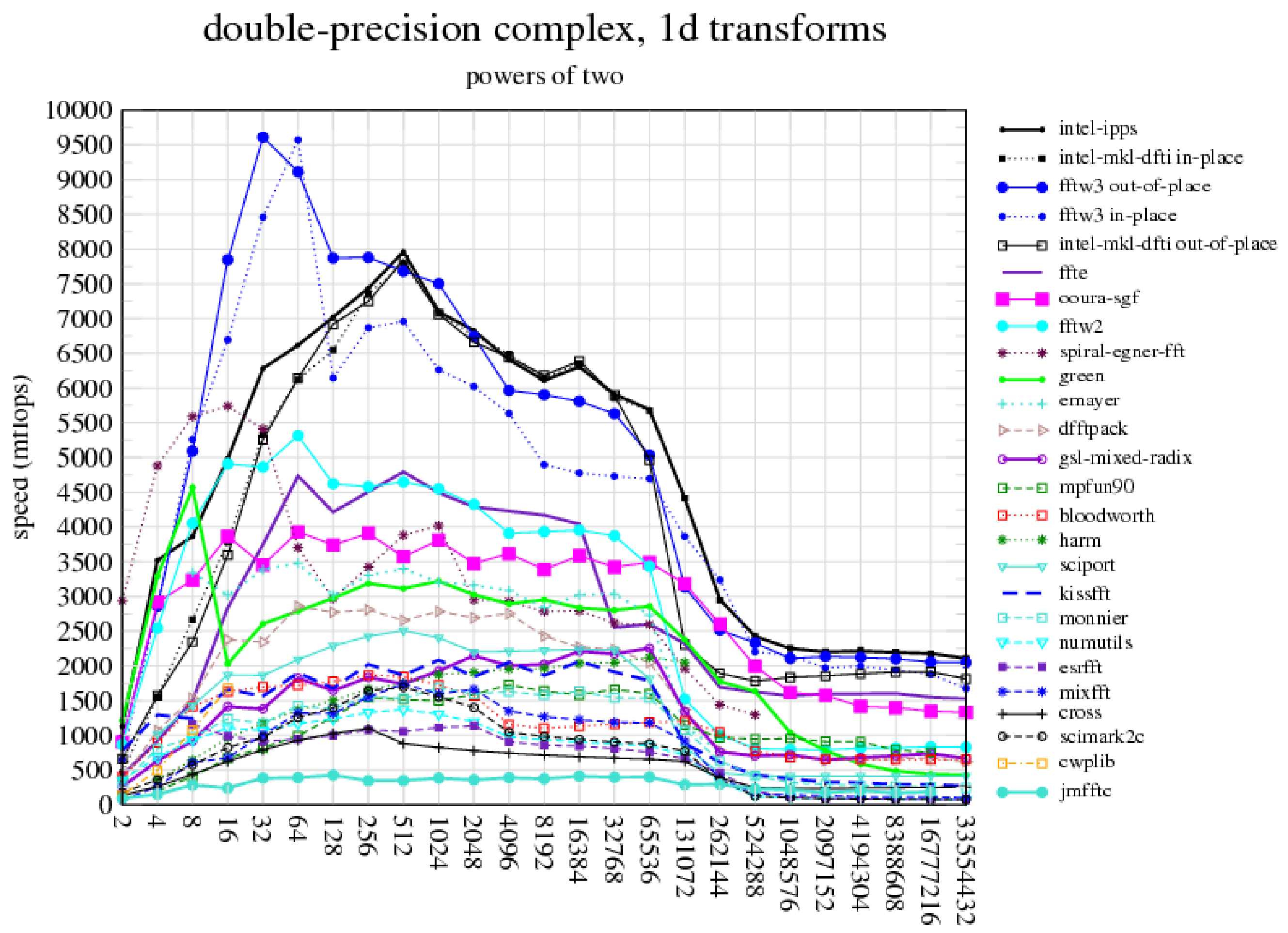
that is the original function can be reconstructed by dividing all the terms into even and odd values of n and by combining them with the coefficients W_N^{-nj}, as before!

Fastest Fourier Transform of the West (FFTW)

- A **fast** and **precise** way to compute ffts in C (or fortran) is the use of the **FFTW** library.
- The library is written in C, but there are **fortran77** and **fortran90 wrappers** to make the use of the library easy for fortran users too.
- The library is **open-source** and, therefore, **portable** on almost any machine, in contrast with proprietary library (e.g. Intel MKL or Cray's CRAFFT) which have optimized performances on particular types of processors, but are not portable!

Fastest Fourier Transform of the West (FFTW)

- Very good **performances** (close or even better than proprietary libraries) and **precision**.



Comparison of performances of FFTW with respect to other libraries on an Intel 3.0 GHz 64-bits processor

Fastest Fourier Transform of the West (FFTW)

- FFTW combines a collection of specialized algorithms which apply to different particular cases (N equal to powers of two, or some combinations of powers of prime numbers, etc.). Each of this algorithm is implemented in a "**codelet**" (there are currently about 150 codelets in the 3.3 version of the library!) and a special code called "**planner**" tries to find the best combination of codelets to make the computation of the FFT as faster as possible.

Fastest Fourier Transform of the West (FFTW)

- After a suitable “**plan**” has been chosen by the planner (one may use special parameters when creating the plans to indicate to the planner how deep the search for the optimal plan must be carried on: the choice of the optimal plan may take even several minutes on a fast machine!), two arrays are passed to the planner which indicate the input and output values of the FFT.
- Once the plan has been created, the evaluation of the FFT is carried out by calling another function which takes the input array and transforms it into the output array!

Fastest Fourier Transform of the West (FFTW)

Note that:

1) The library includes special memory allocation functions (**fftw_malloc**) to ensure the alignment of data in RAM, to optimize the use of the vectorial capabilities of the CPU (SIMD);

2) The library supports multi-threads paradigms (pthreads, OpenMP) for shared memory machines (e.g. multi-core machines);

3) It supports MPI versions of the functions for distributed memory machines (e.g., clusters).

Fastest Fourier Transform of the West (FFTW)

- Example of FFT of a vector of real data:

```
#include <fftw3.h>
...
double *in;
fftw_complex *out;
fftw_plan pf, pb;
...
// Variable allocation
int n = ...;
in = fftw_malloc( sizeof( double ) * n );
out = fftw_malloc( sizeof( fftw_complex ) * n );
...
// Create the forward and backward transform planes
unsigned int flags = FFTW_ESTIMATE;
pf = fftw_plan_dft_r2c_1d( n, in, out, flags );
pb = fftw_plan_dft_c2r_1d( n, out, in, flags );  // Notice the
swap of in and out!
...
// Execute the fft-s
fftw_execute( pf );        // Computes the forward transform
fftw_execute( pb );        // Computes the backward transform
```

Fastest Fourier Transform of the West (FFTW)

- Example of FFT of a vector of real data:

```
...
// Variable and plan de-allocation
fftw_free( in );
fftw_free( out );
fftw_destroy_plan( pf );
fftw_destroy_plan( pf );
}
```

- Notice the use of the “flags” variable, which can take values:

➢ `FFTW_ESTIMATE, FFTW_MEASURE, FFTW_PATIENT, FFTW_EXAUSTIVE;`

➢ `FFTW_DESTROY_INPUT, FFTW_PRESERVE_INPUT`

Fastest Fourier Transform of the West (FFTW)

- For more information and documentation, visit:

  `http://www.fftw.org`