Time Series Analysis:

1. Sampling and DFT

P. F. Góra

https://zfs.fais.uj.edu.pl/pawel_gora

2022

Prediction is very difficult, especially about the future.

attributed to Niels Bohr

The objective

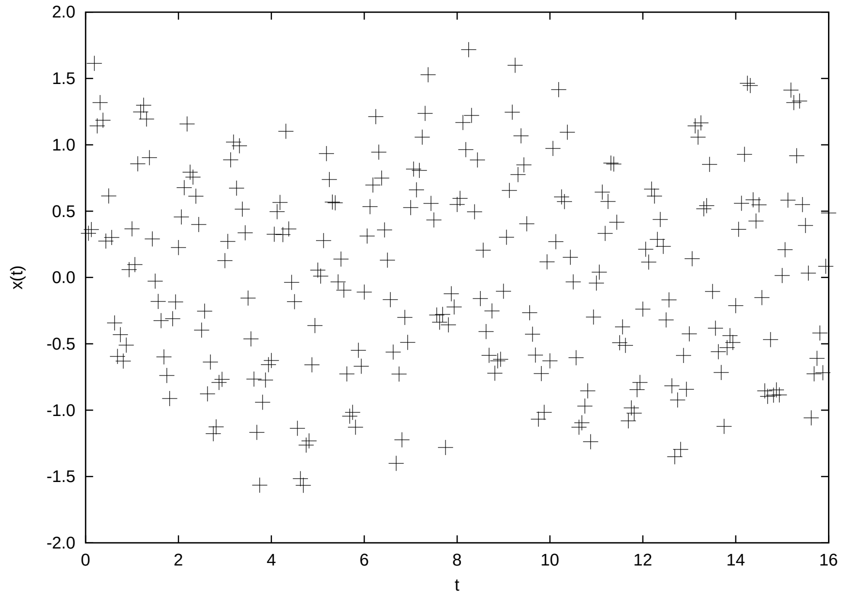
Time Series Analysis belongs to the broad range of Data Science. Unlike in other branches of Data Science, the order in which the data have been collected plays an important role.

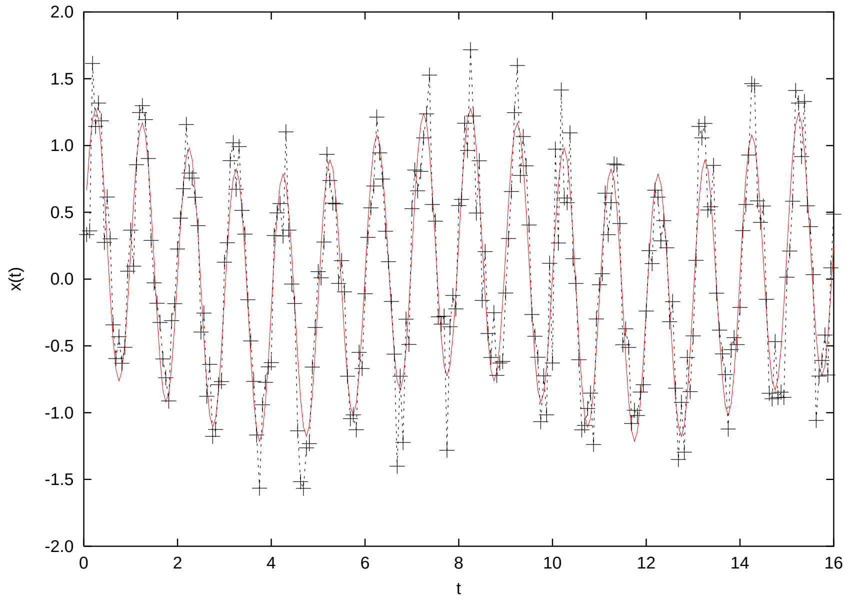
Given a time series

$$\{x_i\}_{i=1}^{N} = \{x_1, x_2, \ldots, x_N\}$$

(usually boring), gain some insight on the *mechanism* that has been responsible for generating this series (possibly interesting) and, perhaps, predict future values of the series (possibly profitable).

Example 1





By their fruit you will recognize them

The plan*

1. Sampling and Discrete Fourier Transform; a Fast Fourier Transform Algorithm
2. Convolution, stationarity, Power Spectrum and window functions; Wiener filter
3. Linear filters, smoothing and denoising
4. Stationary linear stochastic models (AR, MA, ARMA, GARCH)
5. Fitting parameters to stationary linear stochastic models; Youle-Walker equations; Akaike criterion; Linear prediction
6. Seasonal changes and trends (ARIMA)
7. Hurst phenomenon, fractional processes and Detrended Fluctuations Analysis
8. Fractional models (ARFIMA)
9. Multivariate time series
10. Wavelets and their applications
11. Kalman filters (?)
12. Takens theorem and phase space reconstruction, applications for nonlinear prediction

*May change dynamically!

The rules

Written assignments (numerical and theoretical)

- 5 assignments done — ☺
- 2-4 assignments done — an exam
- Less than 2 assignments done — ☹

All assignments will be published on my webpage

Discrete sampling

We are not dealing with continuous signals, but with signals that have been *sampled* with a constant timestep:

$$g_n = g(n\Delta)\,, \quad n = \ldots, -3, -2, -1, 0, 1, 2, 3, \ldots \tag{1}$$

The process of sampling introduces a characteristic frequency (Nyquist frequency):

$$\boxed{f_{\text{Nyq}} = \frac{1}{2\Delta}\,.} \tag{2}$$

Note: If you want to resolve a harmonic wave, you need to sample it *twice* in a period. If you sample with a timestep Δ, you can resolve frequencies $f \in [-f_{\text{Nyq}}, f_{\text{Nyq}}]$.

A side note: Positive and negative frequencies

If you allow for both positive and negative frequencies, $e^{+2\pi ift}$, $e^{-2\pi ift} = e^{+2\pi i(-f)t}$, you can combine them to get both $\sin 2\pi ft$ and $\cos 2\pi ft$, which have the same *frequency*, but differ in *phase*.

Shannon-Kotelnikov Sampling Theorem

Question: When sampling is “good”? Or, under what assumptions, discrete sampling yields the same information as a continuous function?

Theorem: If a function $g(t)$ is *bandwidth limited*, or if it contains only frequencies from the interval $[-f_{\text{Nyq}}, f_{\text{Nyq}}]$, and if we have an *infinite* series sampled with a timestep Δ, then

$$\forall t: \quad g(t) = \Delta \sum_{n=-\infty}^{\infty} g_n \frac{\sin\left(2\pi f_{\text{Nyq}}(t - n\Delta)\right)}{\pi(t - n\Delta)}\,. \tag{3}$$

A signal is bandwidth limited if its Fourier Transform identically vanishes outside a certain interval. Noise (stochastic processes) and discontinuous functions are *not* bandwidth limited.

Aliasing

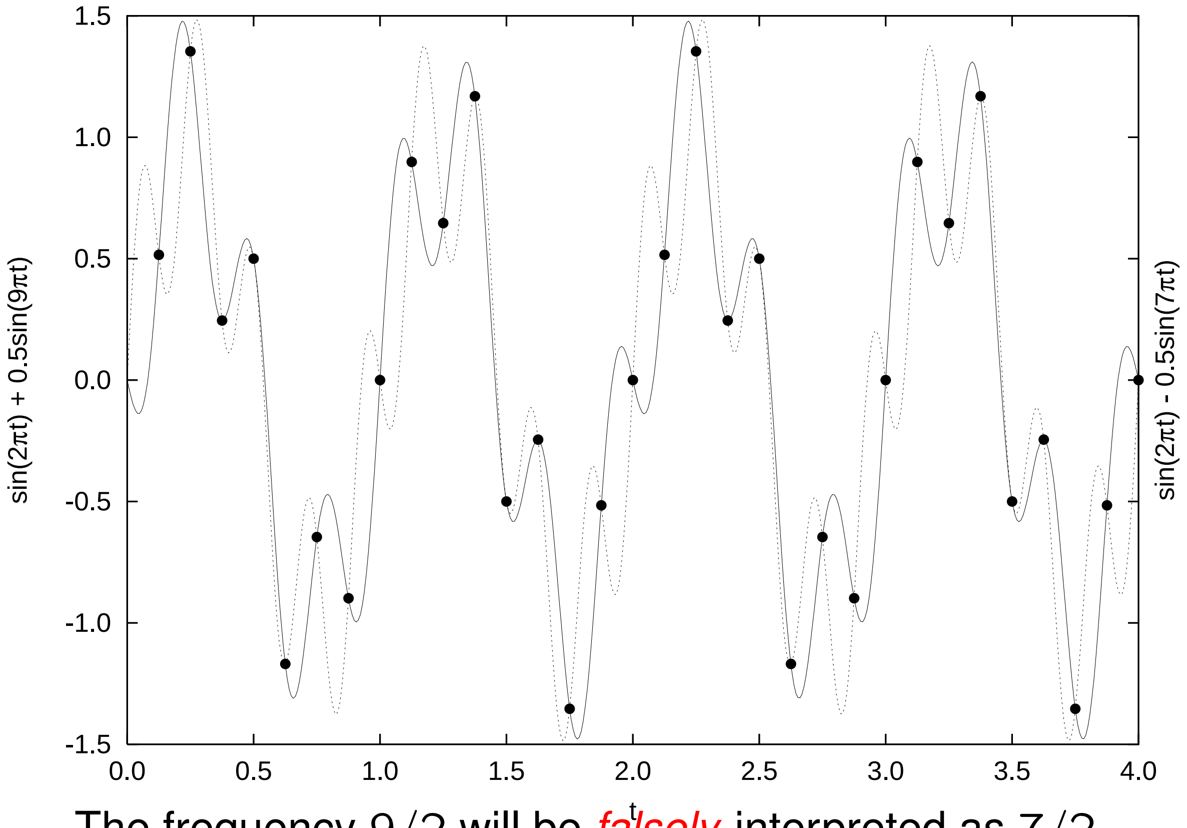
If a signal contains frequencies $f \notin [-f_{\text{Nyq}}, f_{\text{Nyq}}]$, not only are they lost, but they also spoil the information from within $[-f_{\text{Nyq}}, f_{\text{Nyq}}]$.

Consider two harmonic waves $\exp(2\pi i f_1 t)$, $\exp(2\pi i f_2 t)$, with $f_1 - f_2 = k/\Delta$. Then

$$\begin{aligned} \exp(2\pi i f_1 n\Delta) &= \exp(2\pi i (f_2 + k/\Delta) n\Delta) \\ &= \exp(2\pi i f_2 n\Delta + 2\pi i k n) = \exp(2\pi i f_2 n\Delta)\,. \end{aligned}$$

Such waves give *identical* samples when sampled with the timestep Δ.

Example 2



The frequency $9/2$ will be *falsely* interpreted as $7/2$.

Fourier Transform

$$G(f) = \int\limits_{-\infty}^{\infty} g(t) e^{2\pi i f t}\, dt \tag{4}$$

The function must vanish sufficiently fast for $t \to \pm\infty$ for the Fourier Transform to exist.

The inverse transform:

$$g(t) = \int\limits_{-\infty}^{\infty} G(f) e^{-2\pi i f t}\, df \tag{5}$$

You need to remember where to put 2π, there are several conventions.

Properties of Fourier Transform

Convolution:

$$(g \star h)(t) = \int\limits_{-\infty}^{\infty} g(\tau)h(t-\tau)\,d\tau \tag{6a}$$

$$g \star h \leftrightarrow G(f)H(f) \tag{6b}$$

Correlation function:

$$\text{Corr}(g,h) = \int\limits_{-\infty}^{\infty} g(\tau+t)h(t)\,d\tau \tag{7a}$$

$$\text{Corr}(g,h) \leftrightarrow G(f)H^*(f) \tag{7b}$$

Wiener-Khinchin Theorem

Autocorrelation function:

$$\text{Corr}(g,g) \leftrightarrow |G(f)|^2 \tag{8}$$

Parseval identity

$$\int\limits_{-\infty}^{\infty} |g(t)|^2 \, dt = \int\limits_{-\infty}^{\infty} |G(f)|^2 \, df = \text{total power} \tag{9}$$

A finite time series

The sampling theorem demands an infinite series, but in reality we only have a finite series, of the length N. There are two conventions:

- We assume that the series goes to zero at both its ends (we multiply by an appropriate window function if it doesn't) and equals identically zero before it starts and after it ends.
- We assume that the finite series is, in fact, a period of an infinite periodic series. (The sine and cosine do have Fourier Transforms in this convention.)

We adopt the second convention and pretend that our series has the form

$$\dots, \underbrace{g_0, g_1, g_2, \dots, g_{N-1}}_{\text{copy } -1}, \underbrace{g_0, g_1, g_2, \dots, g_{N-1}}_{\text{true data}}, \underbrace{g_0, g_1, g_2, \dots, g_{N-1}}_{\text{copy } +1}, \dots \tag{10}$$

Because we have N (assume N is even) input samples, the Discrete Fourier Transform (DFT) can be evaluated at N points only. We choose to evaluate DFT for the following frequencies only:

$$f_n = \frac{n}{N\Delta}, \quad n = -\frac{N}{2}, \dots, \frac{N}{2}. \tag{11}$$

Note: under the assumption of periodicity, a finite number of samples from within the period is sufficient for the Sampling Theorem to hold.

Discrete Fourier Transform

$$G(f_n) = \int\limits_{-\infty}^{\infty} g(t)\, e^{2\pi i f_n t}\, dt$$

$$= \lim_{M\to\infty} \frac{1}{2M+1} \sum_{s=-M}^{M} \int\limits_{(s-1)(N-1)\Delta}^{s(N-1)\Delta} g(t)\, e^{2\pi i f_n t}\, dt = \int\limits_{0}^{(N-1)\Delta} g(t) e^{2\pi i f_n t} dt$$

$$\simeq \sum_{k=0}^{N-1} \Delta\, g_k\, e^{2\pi i f_n t_k} = \Delta \sum_{k=0}^{N-1} g_k\, e^{2\pi i k n/N}\,. \tag{12}$$

Numbers

$$G_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} g_k \, e^{2\pi i k n/N} \tag{13}$$

are called *discrete Fourier components* of g. The inverse transform reads

$$g_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} G_n \, e^{-2\pi i k n/N} \,. \tag{14}$$

Discrete Parseval identity:

$$\sum_{k=0}^{N-1} |g_k|^2 = \sum_{n=0}^{N-1} |G_n|^2 \,. \tag{15}$$

DFT as a linear transform

Eq. (13) can be written as

$$
\begin{aligned}
G_n &= \sum_{k=0}^{N-1} W_{nk}\, g_k\,, && \text{(16a)}\\
W_{nk} &= \frac{1}{\sqrt{N}} e^{2\pi i k n/N}\,. && \text{(16b)}
\end{aligned}
$$

Numbers W_{nk} can be interpreted as elements of a certain matrix. Thus, a compact form of (16) reads

$$
\mathbf{G} = \mathbf{W}\mathbf{g}\,, \tag{17}
$$

where $\mathbf{G}, \mathbf{g} \in \mathbb{C}^N$, $\mathbf{W} \in \mathbb{C}^{N\times N}$. What is the numerical cost of evaluating DFT? It appears to be $O(N^2)$, or the cost of multiplying a vector by a matrix.

Properties of $\mathbf{W}$

$$\left(\mathbf{W}\mathbf{W}^\dagger\right)_{ls} = \sum_{k=0}^{N-1} \mathbf{W}_{lk} \left(\mathbf{W}^\dagger\right)_{ks} = \sum_{k=0}^{N-1} \mathbf{W}_{lk} \left(\mathbf{W}_{sk}\right)^* = \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi i(l-s)k/N} \tag{18}$$

If $l = s$, all elements are equal to 1 and the result is 1. If $l - s = m \neq 0$,

$$\left(\mathbf{W}\mathbf{W}^\dagger\right)_{ls} = \frac{1}{N} \sum_{k=0}^{N-1} \left(e^{2\pi i m/N}\right)^k = \frac{1 - \left(e^{2\pi i m/N}\right)^N}{N(1 - e^{2\pi i m/N})} = \frac{1 - e^{2m\pi i}}{N(1 - e^{2\pi i m/N})} = 0\,. \tag{19}$$

$$\left(\mathbf{W}\mathbf{W}^\dagger\right)_{ls} = \delta_{ls}.$$

The matrix $\mathbf{W}$ is *Unitary*.

DFT symmetries

Discrete Fourier components:

$$G_n = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} g_k e^{2\pi i n k/N} \tag{20}$$

1. Discrete Fourier components are periodic:

$$\begin{aligned} G_{n+N} &= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} g_k e^{2\pi i (n+N) k/N} \\ &= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} g_k e^{2\pi i n k/N} \underbrace{e^{2\pi i N k/N}}_{e^{2\pi i k}=1} = G_n \end{aligned} \tag{21}$$

2. DFT of a real signal: $g_k \in \mathbb{R} \Rightarrow g_k^* = g_k$.

$$G_n^* = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} g_k^* \, e^{-2\pi i n k/N} = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} g_k \, e^{-2\pi i n k/N} e^{2\pi i N k/N} =$$
$$= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} g_k e^{2\pi i (N-n) k/N} = G_{N-n} \tag{22}$$

3. DFT of a purely imaginary signal: $g_k^* = -g_k$. Calculations similar to those above show that

$$G_n^* = -G_{N-n}\,. \tag{23}$$

Consequences of DFT symmetries

Let $g_k = x_k + iy_k$, $x_k, y_k \in \mathbb{R}$. Because DFT is linear, we have

$$G_n = X_n + iY_n \tag{24a}$$

and because x_k, y_k are real,

$$G_n^* = X_{N-n} - iY_{N-n} \tag{24b}$$

We thus have

$$G_n + G_{N-n}^* = X_n + iY_n + X_n - iY_n = 2X_n \tag{25a}$$
$$G_n - G_{N-n}^* = X_n + iY_n - X_n + iY_n = 2iY_n \tag{25b}$$

We can use (25) to find DFT's of real and imaginary parts of a complex signal.

DFT of two real signals

For reasons that will be explained later, we frequently simultaneously need transforms of two *real* signals, x_k, y_k. We form a complex “signal” $g_k = x_k + iy_k$, calculate its DFT and then use (25) to identify transforms of x_k and y_k. The numerical cost is $O(N \log N)$. The cost of calculating transforms of x_k and y_k separately would be twice as large.

DFT of a single real signal

If we have N samples of a real signal x_k, we form a complex “signal” of half the original length by $g_k = x_k + ix_{N/2+k}$, $k = 0, 1, \dots, N/2$. We calculate DFT of this complex “signal” and use (25) to find the transform of the original signal. Again, the numerical cost of calculating DFT of x_k in its original form would be twice as large.

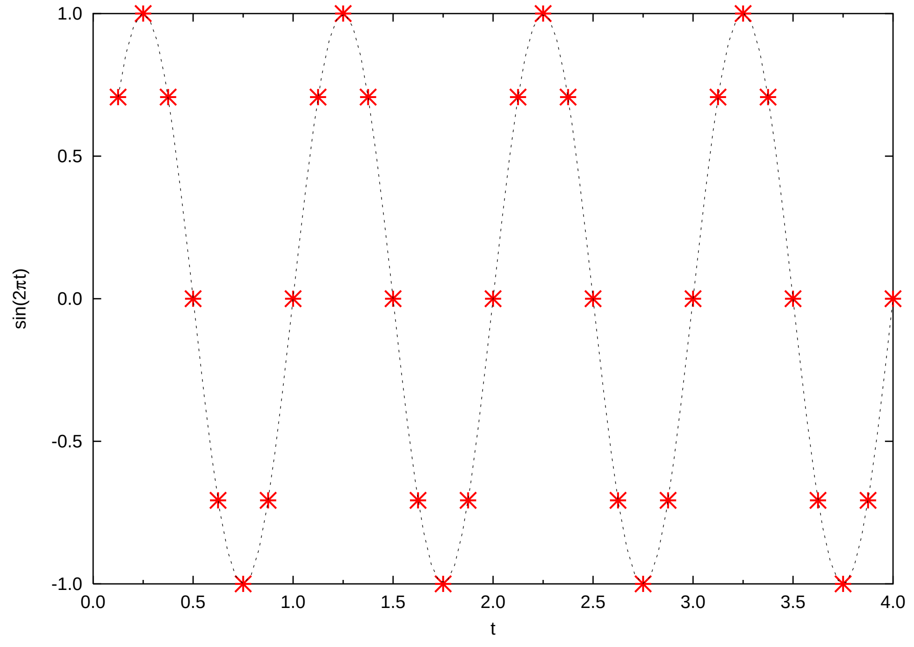
Summary

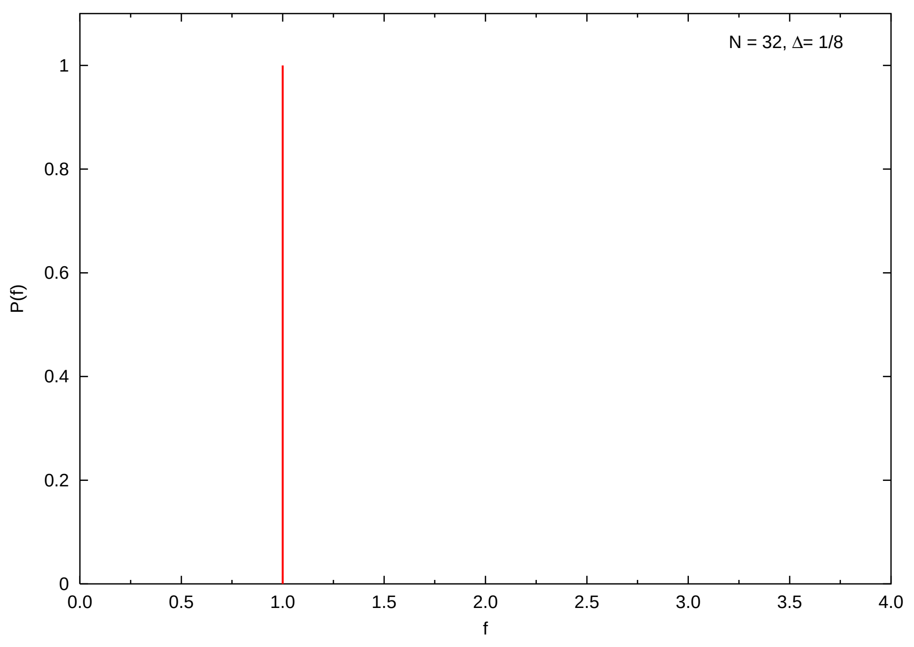
DFT is, up to the normalization, a unitary transform of the *vector of samples* into the *vector of Fourier components.*

DFT represents the vector of samples in another basis, namely in a basis spanned by discretized sines and cosines of frequencies $0, 1/(N\Delta), 2/(N\Delta), \dots, 1/(2\Delta)$.

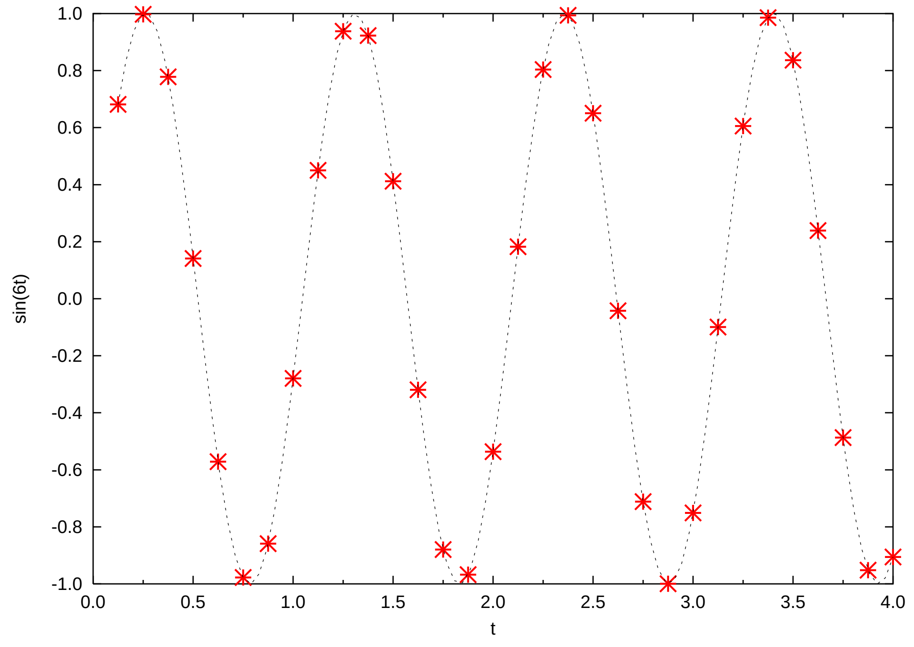
Thanks to symmetries of $\mathbf{W}$, the numerical cost of evaluating DFT can be reduced *significantly* (FFT).

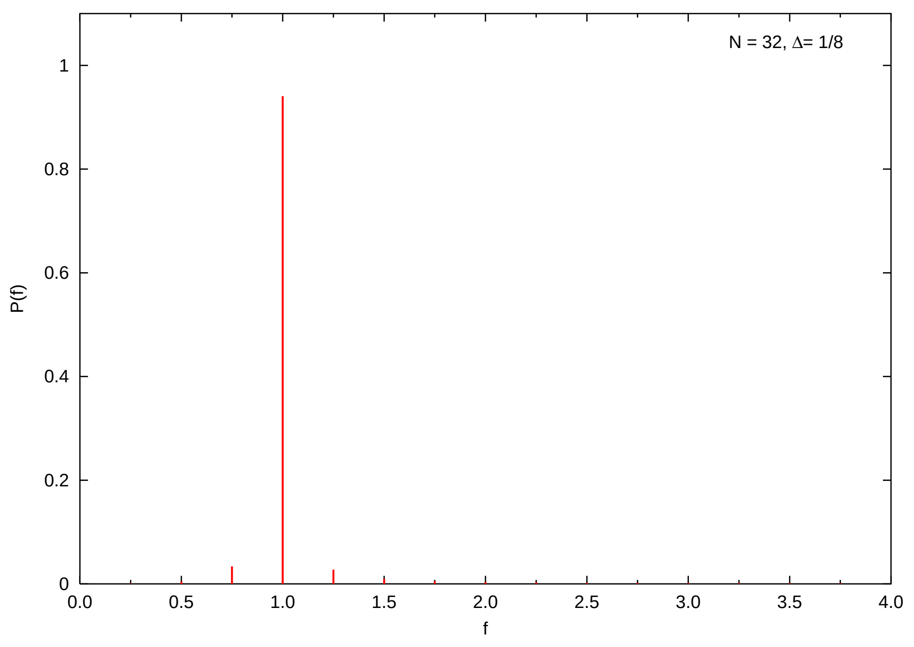
Transform of a signal that coincides with an element of the basis



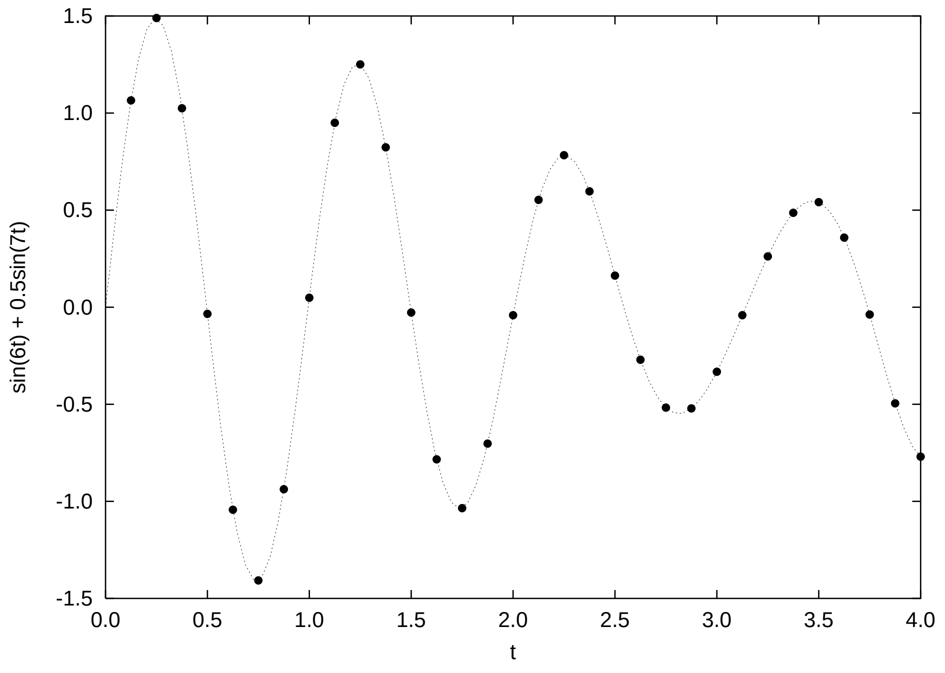


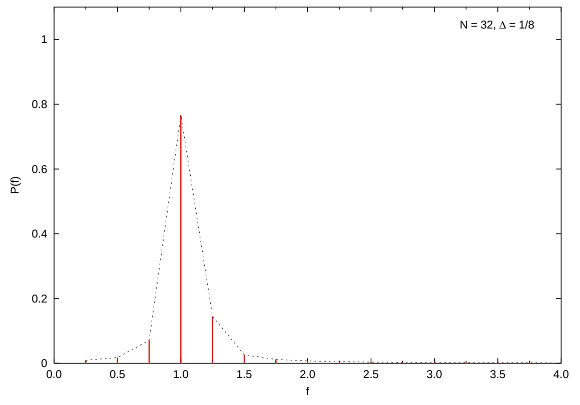
Transform of a signal that doesn’t coincide with an element of the basis

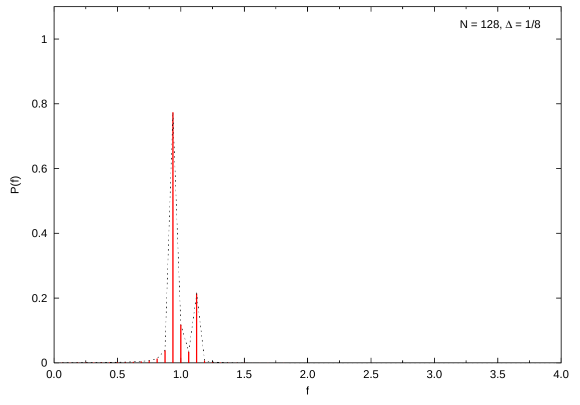




Increasing the number of samples without changing the timestep, improves the resolution







A Vandermonde matrix

The DFT matrix is a special case of a Vandermonde matrix $\mathbf{W} = \frac{1}{\sqrt{N}}\mathbf{V}^{(\mathbf{N})} \in \mathbb{C}^{N\times N}$, where

$$\mathbf{V}^{(N)} = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ z_0 & z_1 & \cdots & z_{N-2} & z_{N-1} \\ z_0^2 & z_1^2 & \cdots & z_{N-2}^2 & z_{N-1}^2 \\ \vdots & \vdots & & \vdots & \vdots \\ z_0^{N-1} & z_1^{N-1} & \cdots & z_{N-2}^{N-1} & z_{N-1}^{N-1} \end{bmatrix} \tag{26}$$

In case of DFT, $z_k = \exp(2\pi i k/N)$.

A Fast Fourier Transform (FFT) Algorithm

An example: $N = 8$

$$\mathbf{G} = \frac{\mathbf{V}^{(8)}}{\sqrt{8}}\mathbf{g} = \frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \frac{1+i}{\sqrt{2}} & i & \frac{-1+i}{\sqrt{2}} & -1 & \frac{-1-i}{\sqrt{2}} & -i & \frac{1-i}{\sqrt{2}} \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & \frac{-1+i}{\sqrt{2}} & -i & \frac{1+i}{\sqrt{2}} & -1 & \frac{1-i}{\sqrt{2}} & i & \frac{-1-i}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & \frac{-1-i}{\sqrt{2}} & i & \frac{1-i}{\sqrt{2}} & -1 & \frac{1+i}{\sqrt{2}} & -i & \frac{-1+i}{\sqrt{2}} \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & \frac{1-i}{\sqrt{2}} & -i & \frac{-1-i}{\sqrt{2}} & -1 & \frac{-1+i}{\sqrt{2}} & i & \frac{1+i}{\sqrt{2}} \end{bmatrix}\begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \end{bmatrix} \tag{27}$$

There are some *patterns*, but it is difficult to see any *symmetries*.

Permute the columns of the matrix and rows of the input vector — this does not change the result:

$$\mathbf{G} = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i & \frac{1+i}{\sqrt{2}} & \frac{-1+i}{\sqrt{2}} & \frac{-1-i}{\sqrt{2}} & \frac{1-i}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & i & -i & i & -i \\ 1 & -i & -1 & i & \frac{-1+i}{\sqrt{2}} & \frac{1+i}{\sqrt{2}} & \frac{1-i}{\sqrt{2}} & \frac{-1-i}{\sqrt{2}} \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & i & -1 & -i & \frac{-1-i}{\sqrt{2}} & \frac{1-i}{\sqrt{2}} & \frac{1+i}{\sqrt{2}} & \frac{-1+i}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & -i & i & -i & i \\ 1 & -i & -1 & i & \frac{1-i}{\sqrt{2}} & \frac{-1-i}{\sqrt{2}} & \frac{-1+i}{\sqrt{2}} & \frac{1+i}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} g_0 \\ g_2 \\ g_4 \\ g_6 \\ g_1 \\ g_3 \\ g_5 \\ g_7 \end{bmatrix} \tag{28}$$

Now we start to see something!

Rewrite Eq. (28) as (block notation)

$$\mathbf{G} = \frac{1}{\sqrt{8}} \begin{bmatrix} \mathbf{V}^{(4)} & \mathbf{\Omega}^{(4)}\mathbf{V}^{(4)} \\ \\ \mathbf{V}^{(4)} & -\mathbf{\Omega}^{(4)}\mathbf{V}^{(4)} \\ \\ \end{bmatrix} \begin{bmatrix} g_0 \\ g_2 \\ g_4 \\ g_6 \\ g_1 \\ g_3 \\ g_5 \\ g_7 \end{bmatrix} \tag{29}$$

…where

$$\mathbf{V}^{(4)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}, \tag{30}$$

$$\mathbf{\Omega}^{(4)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1+i}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & \frac{-1+i}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \left(\frac{1+i}{\sqrt{2}}\right)^0 & & & \\ & \left(\frac{1+i}{\sqrt{2}}\right)^1 & & \\ & & \left(\frac{1+i}{\sqrt{2}}\right)^2 & \\ & & & \left(\frac{1+i}{\sqrt{2}}\right)^3 \end{bmatrix} \tag{31}$$

Thus

$$\mathbf{G} = \frac{1}{\sqrt{8}} \begin{bmatrix} \mathbf{V}^{(4)} \begin{bmatrix} g_0 \\ g_2 \\ g_4 \\ g_6 \end{bmatrix} + \mathbf{\Omega}^{(4)} \mathbf{V}^{(4)} \begin{bmatrix} g_1 \\ g_3 \\ g_5 \\ g_7 \end{bmatrix} \\ \\ \mathbf{V}^{(4)} \begin{bmatrix} g_0 \\ g_2 \\ g_4 \\ g_6 \end{bmatrix} - \mathbf{\Omega}^{(4)} \mathbf{V}^{(4)} \begin{bmatrix} g_1 \\ g_3 \\ g_5 \\ g_7 \end{bmatrix} \end{bmatrix} \tag{32}$$

Pieces of different colours are evaluated only once. The multiplication by $\mathbf{\Omega}^{(4)}$ proceeds in a linear time. We have reduced the total number of operations *by half*.

The key point is: $\mathbf{V}^{(4)}$ can be factorised in the same manner, with an appropriate permutation of the input vector:

$$\mathbf{V}^{(4)} \begin{bmatrix} g_0 \\ g_2 \\ g_4 \\ g_6 \end{bmatrix} = \begin{bmatrix} \mathbf{V}^{(2)} & \mathbf{\Omega}^{(2)}\mathbf{V}^{(2)} \\ \mathbf{V}^{(2)} & -\mathbf{\Omega}^{(2)}\mathbf{V}^{(2)} \end{bmatrix} \begin{bmatrix} g_0 \\ g_4 \\ g_2 \\ g_6 \end{bmatrix} \tag{33}$$

$$\mathbf{V}^{(2)} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathbf{\Omega}^{(2)} = \begin{bmatrix} 1 & \\ & i \end{bmatrix} = \begin{bmatrix} i^0 & \\ & i^1 \end{bmatrix}$$

(similarly for $[g_1, g_3, g_5, g_7]^T$)

Thus

$$\mathrm{G} = \frac{1}{\sqrt{8}} \begin{bmatrix} \begin{bmatrix} \mathbf{V}^{(2)} \begin{bmatrix} g_0 \\ g_4 \end{bmatrix} + \mathbf{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_2 \\ g_6 \end{bmatrix} \\ \mathbf{V}^{(2)} \begin{bmatrix} g_0 \\ g_4 \end{bmatrix} - \mathbf{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_2 \\ g_6 \end{bmatrix} \end{bmatrix} + \mathbf{\Omega}^{(4)} \begin{bmatrix} \mathbf{V}^{(2)} \begin{bmatrix} g_1 \\ g_5 \end{bmatrix} + \mathbf{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_3 \\ g_7 \end{bmatrix} \\ \mathbf{V}^{(2)} \begin{bmatrix} g_1 \\ g_5 \end{bmatrix} - \mathbf{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_3 \\ g_7 \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \mathbf{V}^{(2)} \begin{bmatrix} g_0 \\ g_4 \end{bmatrix} + \mathbf{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_2 \\ g_6 \end{bmatrix} \\ \mathbf{V}^{(2)} \begin{bmatrix} g_0 \\ g_4 \end{bmatrix} - \mathbf{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_2 \\ g_6 \end{bmatrix} \end{bmatrix} - \mathbf{\Omega}^{(4)} \begin{bmatrix} \mathbf{V}^{(2)} \begin{bmatrix} g_1 \\ g_5 \end{bmatrix} + \mathbf{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_3 \\ g_7 \end{bmatrix} \\ \mathbf{V}^{(2)} \begin{bmatrix} g_1 \\ g_5 \end{bmatrix} - \mathbf{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_3 \\ g_7 \end{bmatrix} \end{bmatrix} \end{bmatrix} \tag{34}$$

2-d vectors of different colours are evaluated only once. We have reduced the number of operations *four times*.

The final puzzle:

We have performed a permutation of the input vector
$[g_0, g_1, g_2, g_3, g_4, g_5, g_6, g_7]^T \to [g_0, g_4, g_2, g_6, g_1, g_5, g_3, g_7]^T$.
Is this permutation easy to implement?

$$\begin{array}{c} 0\\1\\2\\3\\4\\5\\6\\7 \end{array} = \begin{array}{c} 000_2\\001_2\\010_2\\011_2\\100_2\\101_2\\110_2\\111_2 \end{array} \xrightarrow[\text{reverse the order}]{} \begin{array}{c} 000_2\\100_2\\010_2\\110_2\\001_2\\101_2\\011_2\\111_2 \end{array} = \begin{array}{c} 0\\4\\2\\6\\1\\5\\3\\7 \end{array} \tag{35}$$

The input vector is sorted in the bit reversal order of indices.

This algorithm easily generalizes to any $N = 2^s$, $s \in \mathbb{N}$.

The need to evaluate terms like $\sin\left(\frac{n\pi}{N}\right)$, $\cos\left(\frac{n\pi}{N}\right)$ is also reduced — there is no need to call library functions $\sin(\cdot)$, $\cos(\cdot)$. On each stage of factorisations a root of i is evaluated *only once* with ready-to-use formulas. Then only the powers of that root are evaluated.

What is the final numerical cost?

Each factorisation reduces the number of operations by half.

There are $\log_2 N$ factorisations.

The numerical cost of the FFT algorithm equals

$$O(N \log_2 N)$$

For example, for $N = 65536 = 2^{16}$ FFT reduces the numerical cost of evaluating DFT more than *four thousand times*.

Remarks

- Similar algorithms can be constructed for $N = 3^s$, $N = 5^s$ and, in general, any $N = q^s$, where q is prime.
- Good libraries automatically factorise also matrices of the size $N = q_1^{s_1} q_2^{s_2} \cdots q_m^{s_m}$, where $q_1, q_2, \dots, q_m$ are small primes.
- If we analyse a series that we *calculate*, we can control its length and we *should* make it a power of a small prime. If we analyse an “experimental” series, it is wise either to truncate it or to pad it with zeros to a power of a small prime.
- There are *many* FFT packages over the Internet; some of them are free and good. I recommend The Fastest Fourier Transform in the West, `http://www.fftw.com`. Python users may use functions from the NumPy library.

- Transforms spanned by sines or cosines only are also in use (one-sided transforms). There are “fast” algorithms for one-sided transforms, too.
- A 2-d DFT can also be used: If $\mathbf{g} \in \mathbb{C}^{N\times N}$ with $N = 2^s$, then

$$\mathbf{G} = \mathbf{W}\mathbf{g}\mathbf{W}^\dagger \tag{36}$$

There are “fast” algorithms for that. Their numerical cost is $O(N^2 \log_2 N)$.