Transformée de Fourier discrète

& FFT

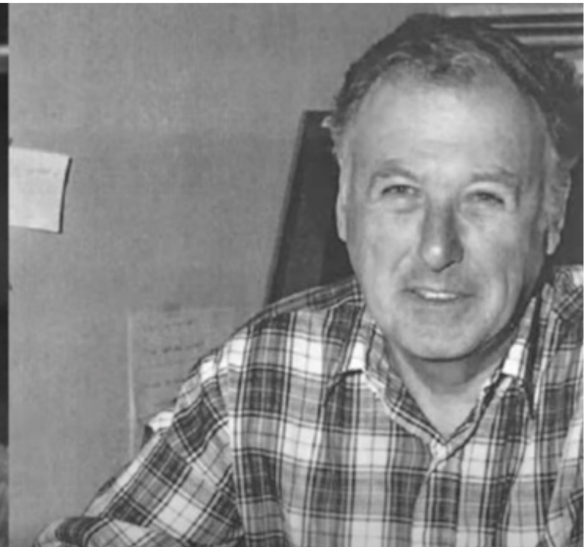
Hugues GARNIER

hugues.garnier@univ-lorraine.fr

Les hommes du jour

Carl Friedrich Gauss

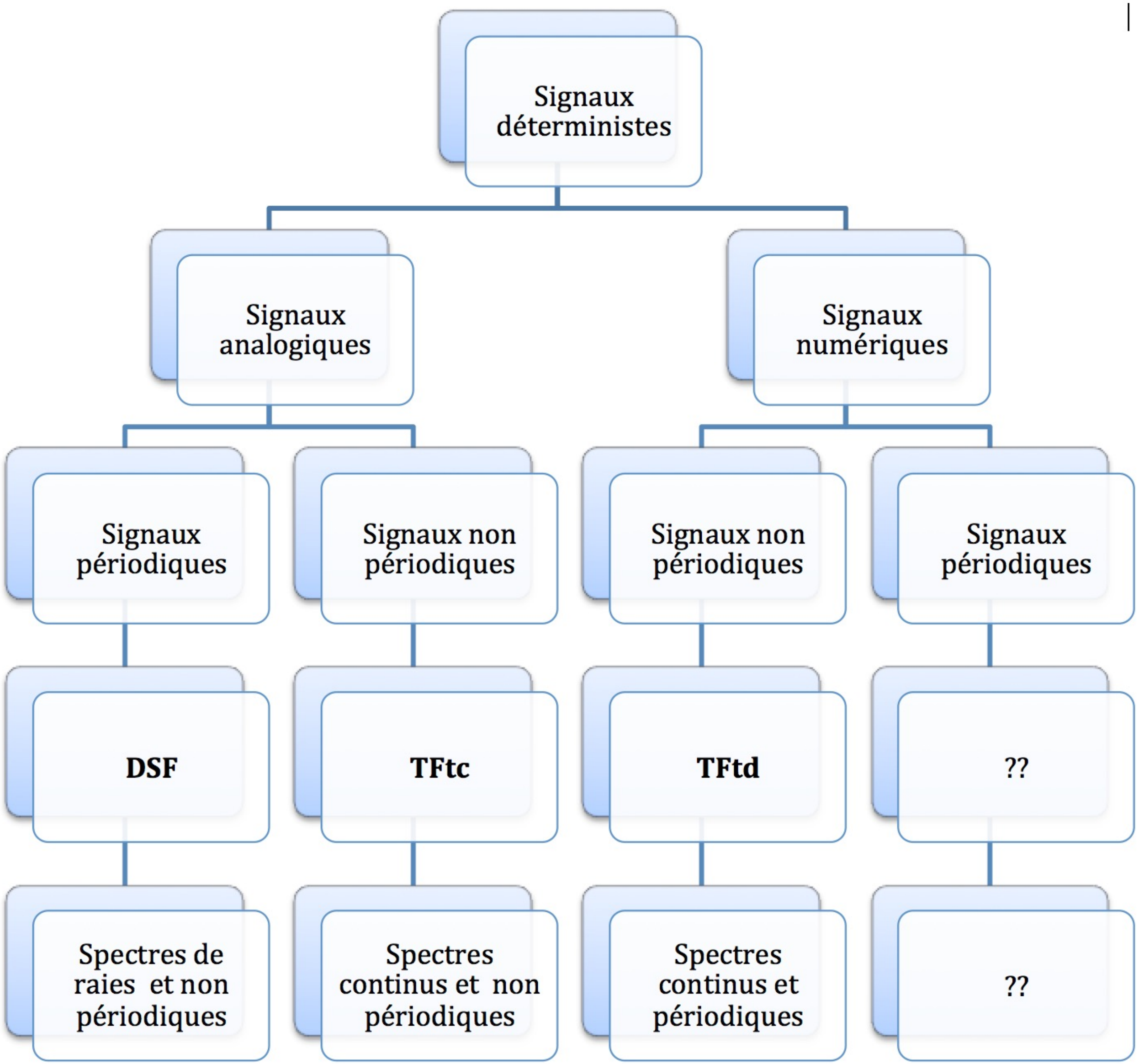
John Tukey



James Cooley

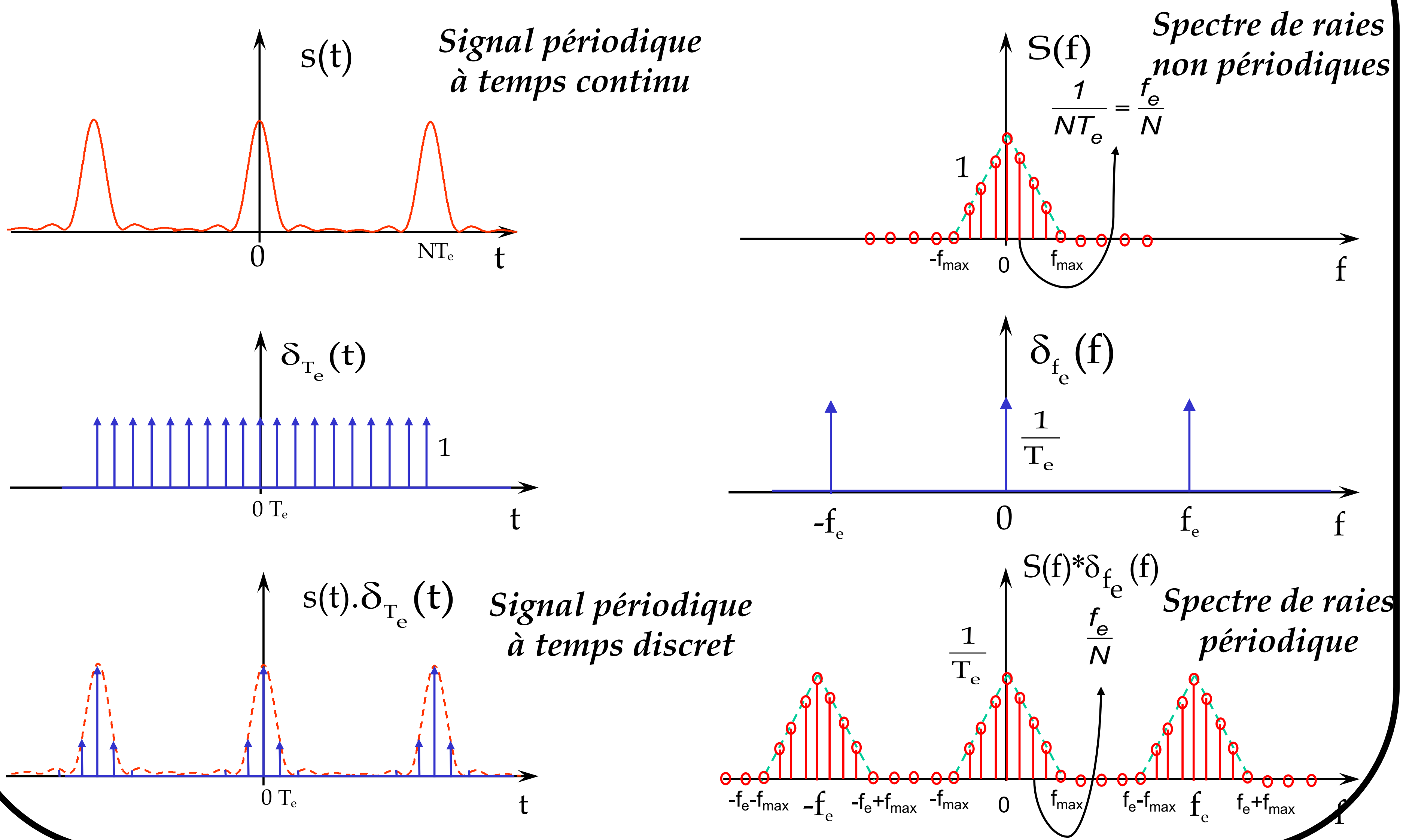
FFT : The algorithm that transformed the world

Analyse de Fourier de signaux déterministes



Introduction à la transformée de Fourier discrète

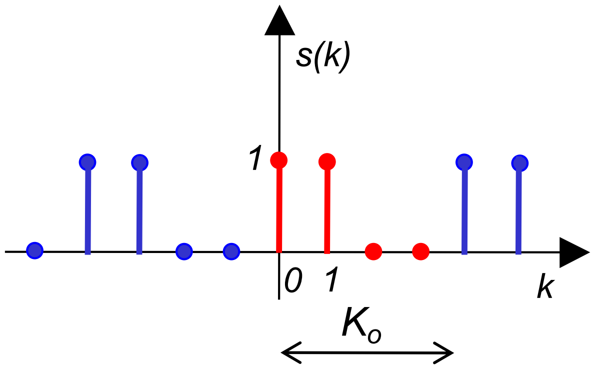
Domaine temporel | **Domaine fréquentiel**



Transformée de Fourier discrète (TFD) - Définition

- Soit $s(k)$ ***périodique*** de période K_o :

$$s\left[\left(k+lK_o\right)\right]=s\left(k\right) \qquad l\in Z$$



- TFD de $s(k)$ *(calculé à partir de K_o échantillons de $s(k)$)*

$$S(m)=S\left(m\Delta f\right)=\sum_{k=0}^{K_o-1}s(k)\,e^{-j2\pi\frac{km}{K_o}} \qquad m=0,1,\ldots\ K_o\text{-}1$$

$S(m)$ est une fonction discrète de la variable f (en Hz)

k : indice temporel, $k=0,\ 1,\ \ldots,\ K_o\text{-}1$

m : indice fréquentiel, $m=0,\ 1,\ \ldots,\ K_o\text{-}1$

K_o : nombre d'échantillons d'une période du signal et de composantes d'une période du spectre

$s(k)=s(kT_e)$: $k^{\text{ème}}$ échantillon temporel du signal

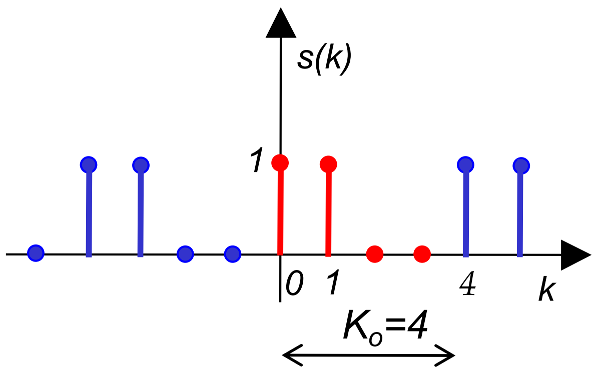
$S(m)=S(m\Delta f)$: $m^{\text{ème}}$ composante du spectre

T_e : période d'échantillonnage (intervalle entre 2 échantillons de $s(k)$)

$\Delta f=\dfrac{1}{K_oT_e}=\dfrac{f_e}{K_o}$: pas fréquentiel (intervalle entre 2 raies du spectre de $S(m)$)

TFD - Exemples

$$\begin{cases} s(k)=\delta(k)+\delta(k-1) \\ s(k+nK_o)=s(k), \quad n\in Z, \quad K_o=4 \end{cases}$$



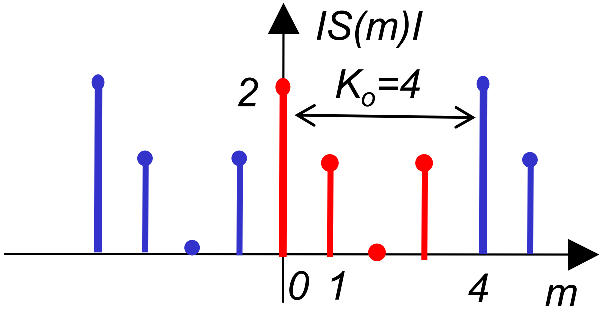
$$S(m)=S\left(m\frac{f_e}{4}\right)=\sum_{k=0}^{3}s(k)\,e^{-j\pi\frac{km}{2}} \qquad m=0,1,2,3$$

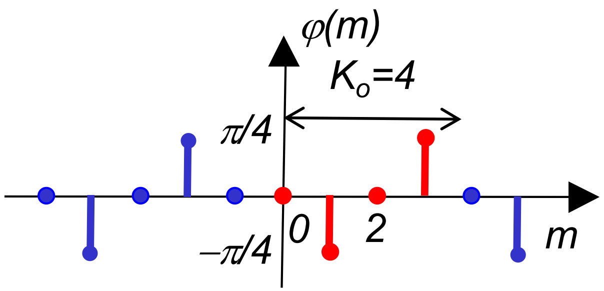
$$S(0)=2 \qquad |S(0)|=2 \qquad \varphi(0)=0$$

$$S(1)=S(f_e/4)=1-j \qquad |S(1)|=\sqrt{2} \qquad \varphi(1)=-\pi/4$$

$$S(2)=S(f_e/2)=0 \qquad |S(2)|=0 \qquad \varphi(2)=0$$

$$S(3)=S(3f_e/4)=1+j \qquad |S(3)|=\sqrt{2} \qquad \varphi(3)=\pi/4$$





s(k) : signal à temps discret

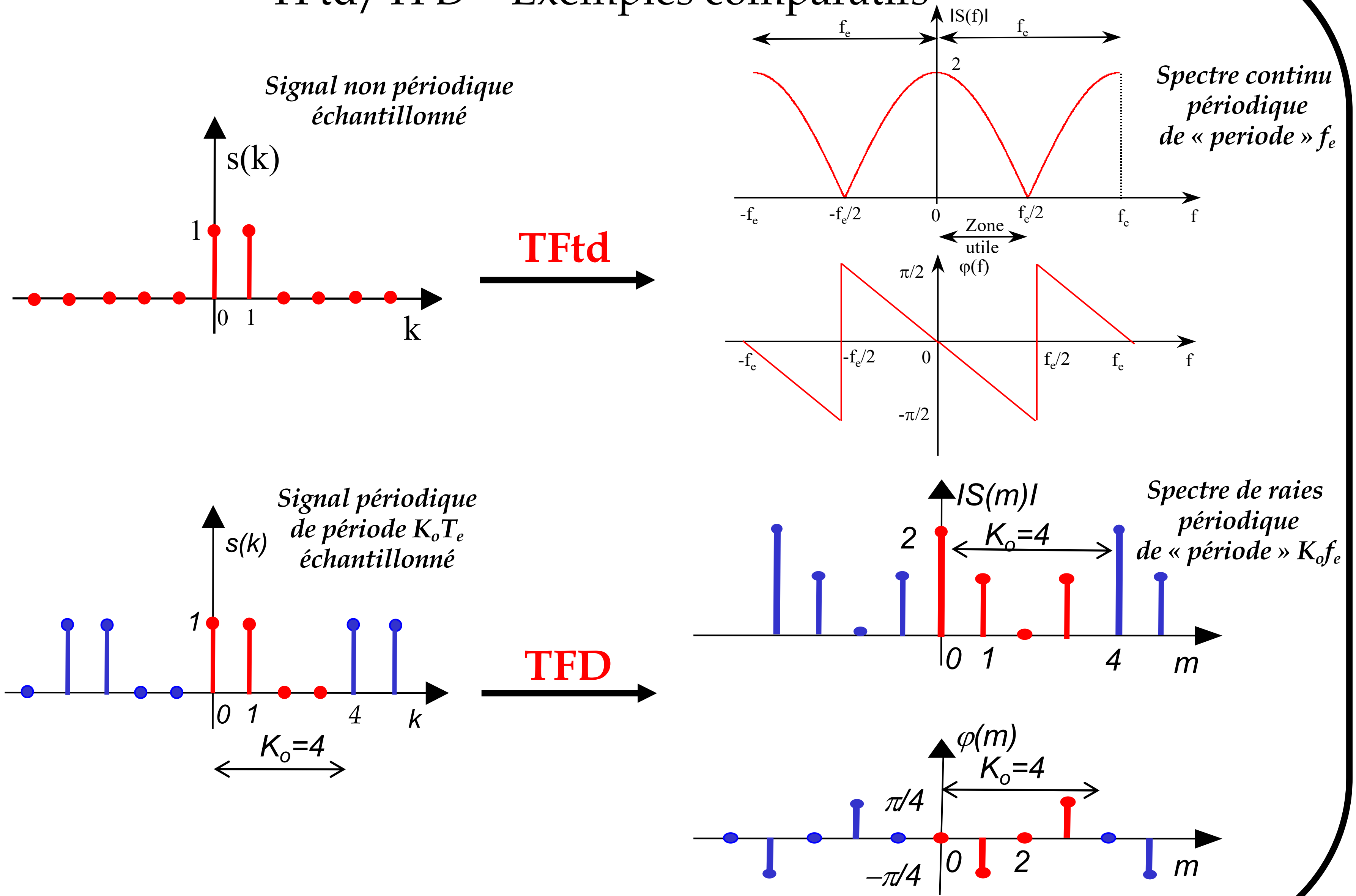
Ses spectres d'amplitude et de phase sont donc périodiques de « période » f_e

Calcul de $K_o=4$ composantes spectrales sur [0 ; f_e[

$S(m)=S(m\Delta f)$ pour $m=0, 1, 2, 3$ et $\Delta f=f_e/K_o$

s(k) et *S(m)* sont périodiques de même période : $K_o=4$

TFtd/TFD – Exemples comparatifs

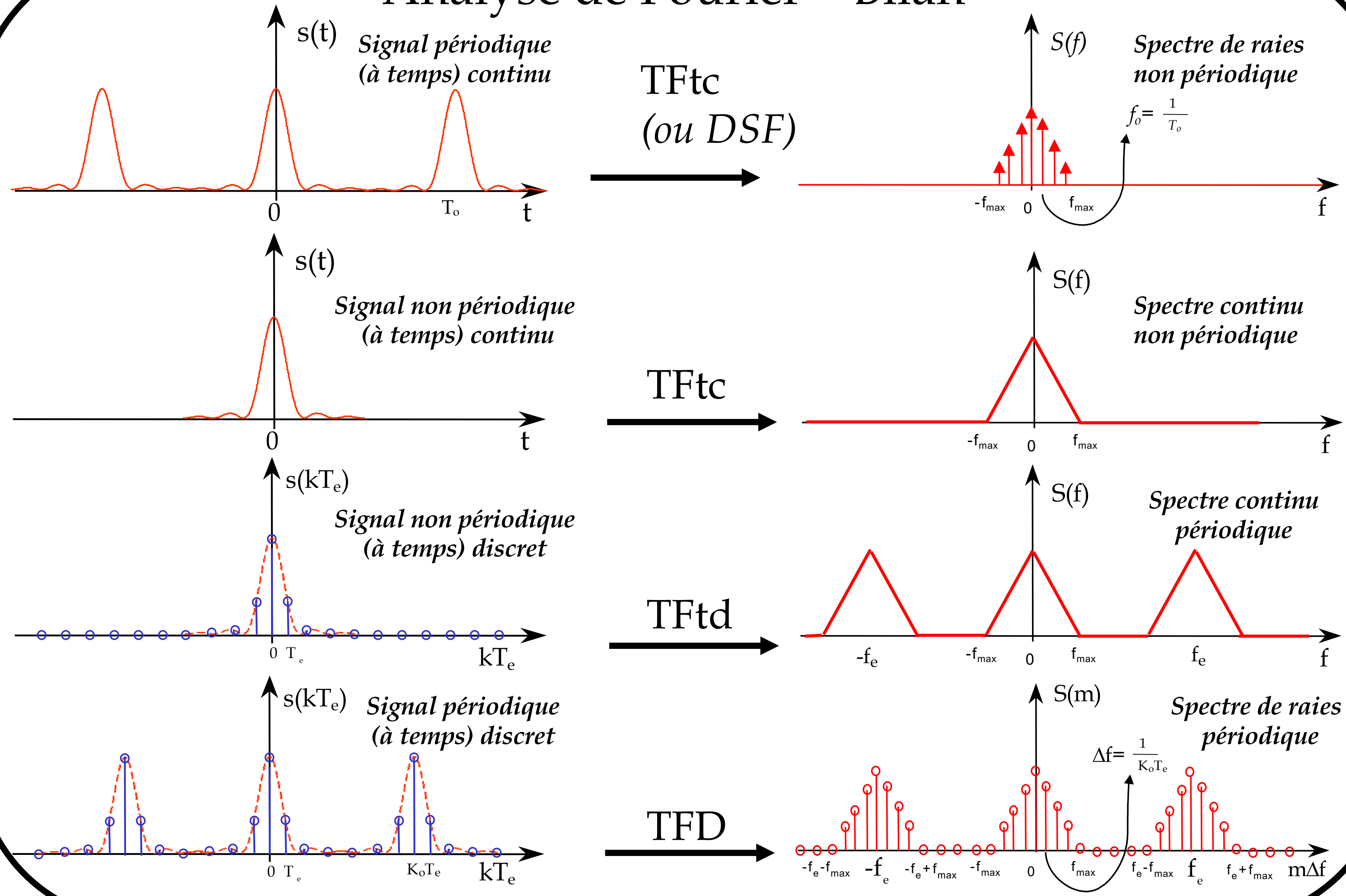


Analyse de Fourier – Bilan

Signal	**Temps continu**	**Temps discret**
Périodique	Décomposition en série de Fourier $$s(t)=\sum_{n=-\infty}^{+\infty} c_n e^{j2\pi n f_o t}$$ $$c_n=\frac{1}{T_o}\int_{t_o}^{t_o+T_o} s(t)e^{-j2\pi n f_o t}dt$$ ***spectre* de raies *non périodique***	Transformée de Fourier discrète TFD $$S(m)=\sum_{k=0}^{K_o-1} s(k)e^{-j\frac{2\pi}{K_o}km}$$ ***spectre* de raies *et périodique***
Non périodique	Transformée de Fourier à temps continu - **TFtc** $$S(f)=\int_{-\infty}^{+\infty} s(t)e^{-j2\pi ft}dt$$ ***spectre continu non périodique***	Transformée de Fourier à temps discret – **TFtd** $$S(f)=\sum_{k=-\infty}^{+\infty} s(k)e^{-j2\pi fkT_e}$$ ***spectre continu et périodique***

Dualité	***échantillonné***	⟷	***périodique***
temps-fréquence	***continu***	⟷	***non périodique***

Analyse de Fourier – Bilan



Implantation pratique de la TFD : vers la FFT

- La TFD calculée pour N composantes fréquentielles est appelée ***TFD à N points*** ou ***TFD d'ordre N***
  - $s(k)$ et $S(m)$ sont périodiques de même période N
  - Elle présente l'inconvénient majeur de nécessiter N^2 opérations pour calculer $S(m)$ à partir de N échantillons de $s(k)$
- Il existe un calcul de $S(m)$ qui utilise de façon très ingénieuse les propriétés de l'exponentielle si N est une puissance de 2

- Cette TFD s'appelle **FFT** (*Fast Fourier Transform*), inventée en 1965
  - La FFT présente l'avantage de nécessiter $N \log2(N)$ opérations pour calculer $S(m)$ à partir de N échantillons de $s(k)$
  - Fonction *fft* sous Matlab

Article référence de la FFT - 1965

An Algorithm for the Machine Calculation of Complex Fourier Series

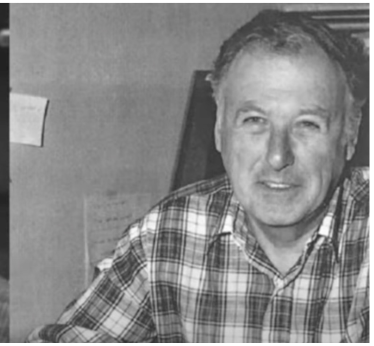
By James W. Cooley and John W. Tukey

An efficient method for the calculation of the interactions of a 2^m factorial experiment was introduced by Yates and is widely known by his name. The generalization to 3^m was given by Box et al. [1]. Good [2] generalized these methods and gave elegant algorithms for which one class of applications is the calculation of Fourier series. In their full generality, Good's methods are applicable to certain problems in which one must multiply an N-vector by an $N \times N$ matrix which can be factored into m sparse matrices, where m is proportional to $\log N$. This results in a procedure requiring a number of operations proportional to $N \log N$ rather than N^2. These methods are applied here to the calculation of complex Fourier series. They are useful in situations where the number of data points is, or can be chosen to be, a highly composite number. The algorithm is here derived and presented in a rather different form. Attention is given to the choice of N. It is also shown how special advantage can be obtained in the use of a binary computer with $N = 2^m$ and how the entire calculation can be performed within the array of N data storage locations used for the given Fourier coefficients.

Consider the problem of calculating the complex Fourier series

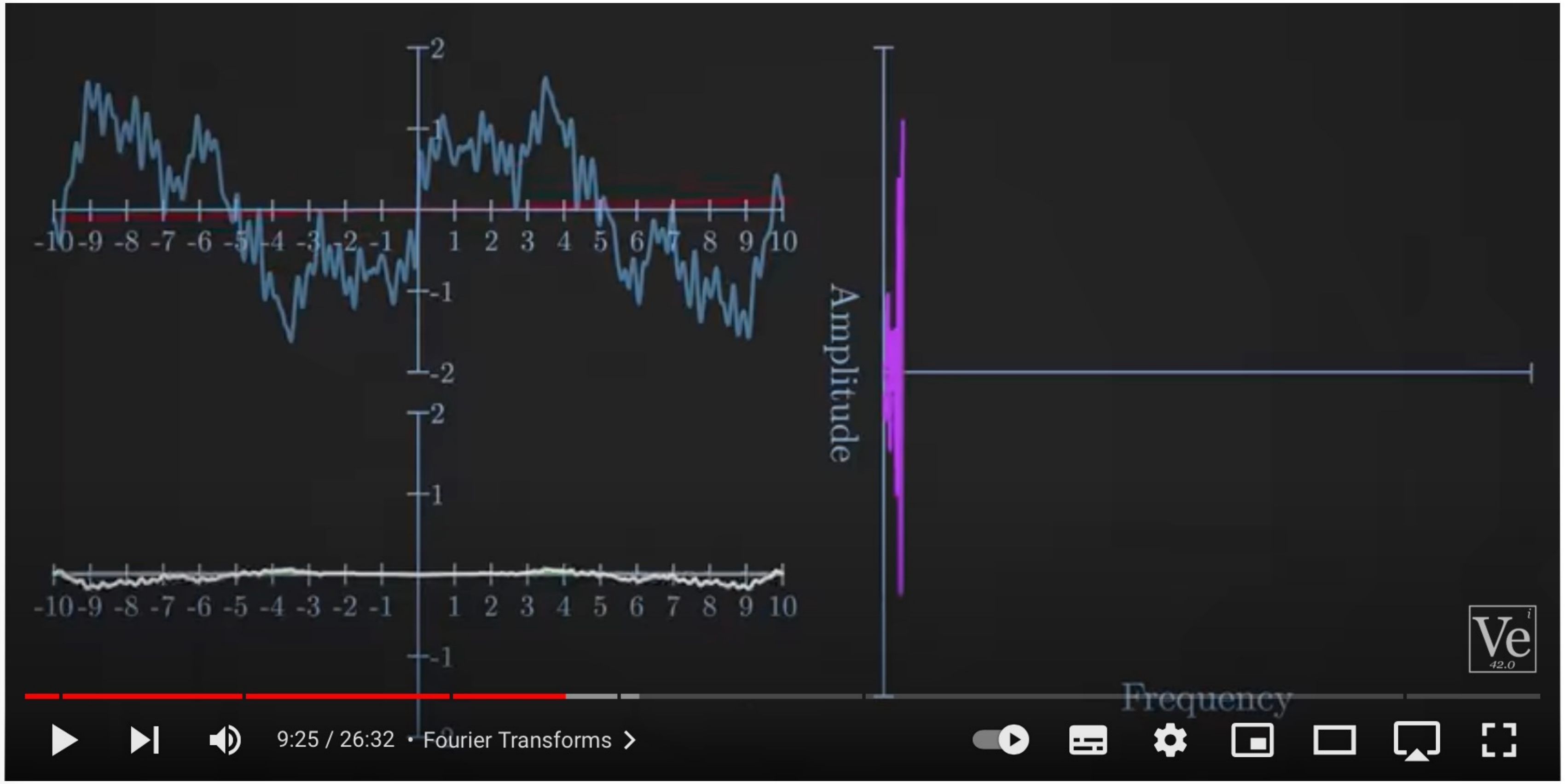
$$(1) \qquad X(j) = \sum_{k=0}^{N-1} A(k) \cdot W^{jk}, \qquad j = 0, 1, \cdots, N-1,$$

John Tukey



James Cooley

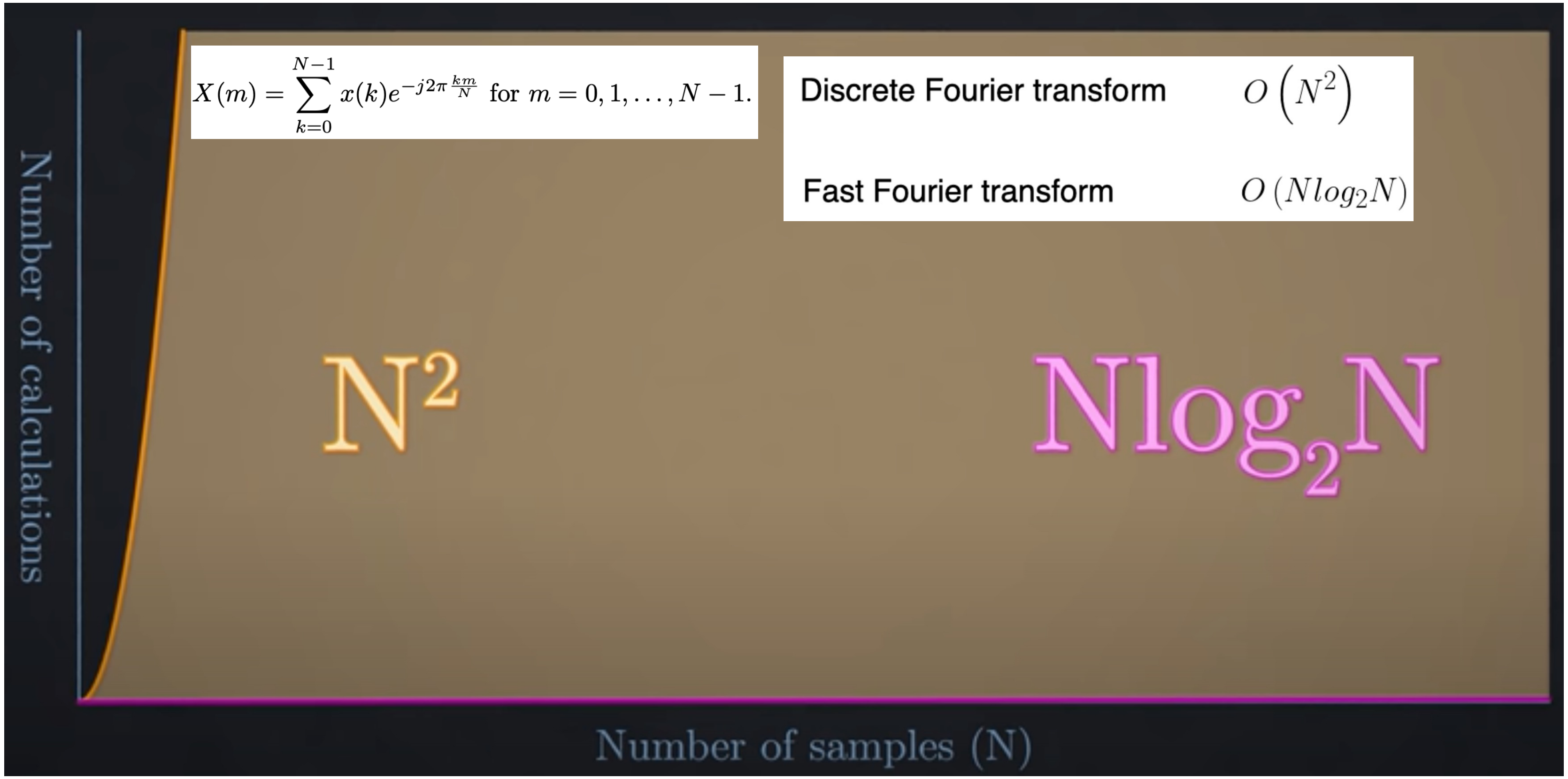
FFT: « *The most important numerical algorithm of our lifetime* ». Gilbert Strang



The Algorithm That Transformed The World

www.youtube.com/watch?app=desktop&v=nmgFG7PUHfo

Avantage de la FFT – Moins de calculs et donc plus rapide, d'où son nom



From **FFT : the Algorithm that transformed the world**
www.youtube.com/watch?app=desktop&v=nmgFG7PUHfo

On the surface, this might not seem like a big deal. However, when N is large enough it can make a world of difference. Have a look at the following table.

N	1000	10^6	10^9
N^2	10^6	10^{12}	10^{18}
$N log_2 N$	10^4	$20\,x\,10^6$	$30\,x\,10^9$

Say it took 1 nanosecond to perform one operation. It would take the Fast Fourier Transform algorithm approximately 30 seconds to compute the Discrete Fourier Transform for a problem of size `N = 10`^{9}. In contrast, the regular algorithm would need several decades.

$$10^{18} ns \rightarrow 31.2\, years$$

$$30\,x\,10^9 ns \rightarrow 30\, seconds$$