Transformadas para Análisis de Señal

- Introducción
- Transformada de Fourier
- Transformada Discreta de Fourier
- Transformada Rápida de Fourier
- Wavelets
    - Transformada Wavelet Continua
    - Transformada Wavelet Discreta

Introducción

Dada una función compleja de variable real (habitualmente, dependiente del tiempo):

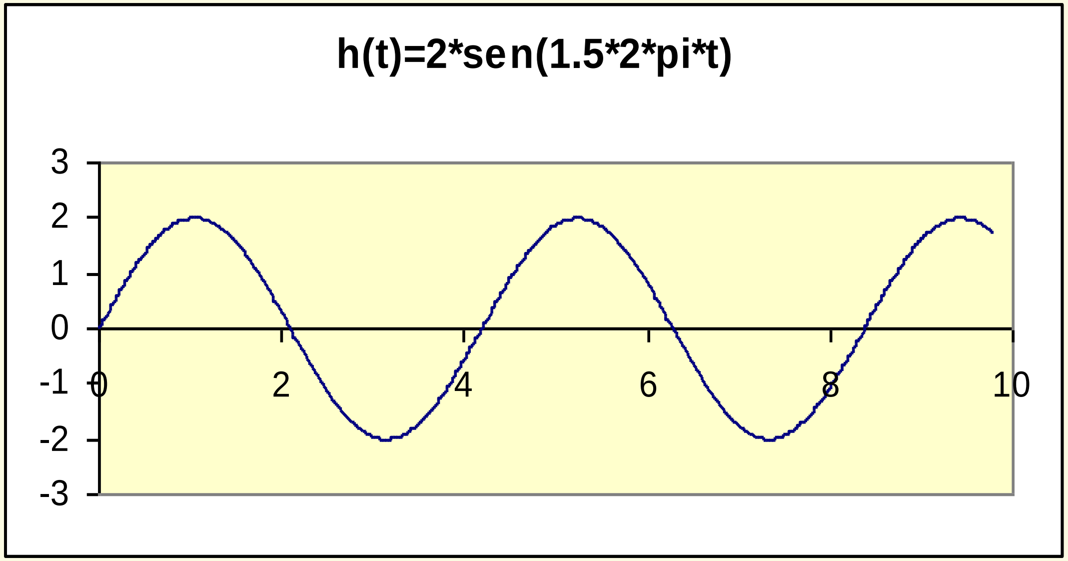
$$h: \Re \longrightarrow C$$
$$t \longrightarrow h(t)$$

h es una función periódica si existe $T \in \Re$ tal que $h(t+T)=h(t)$.

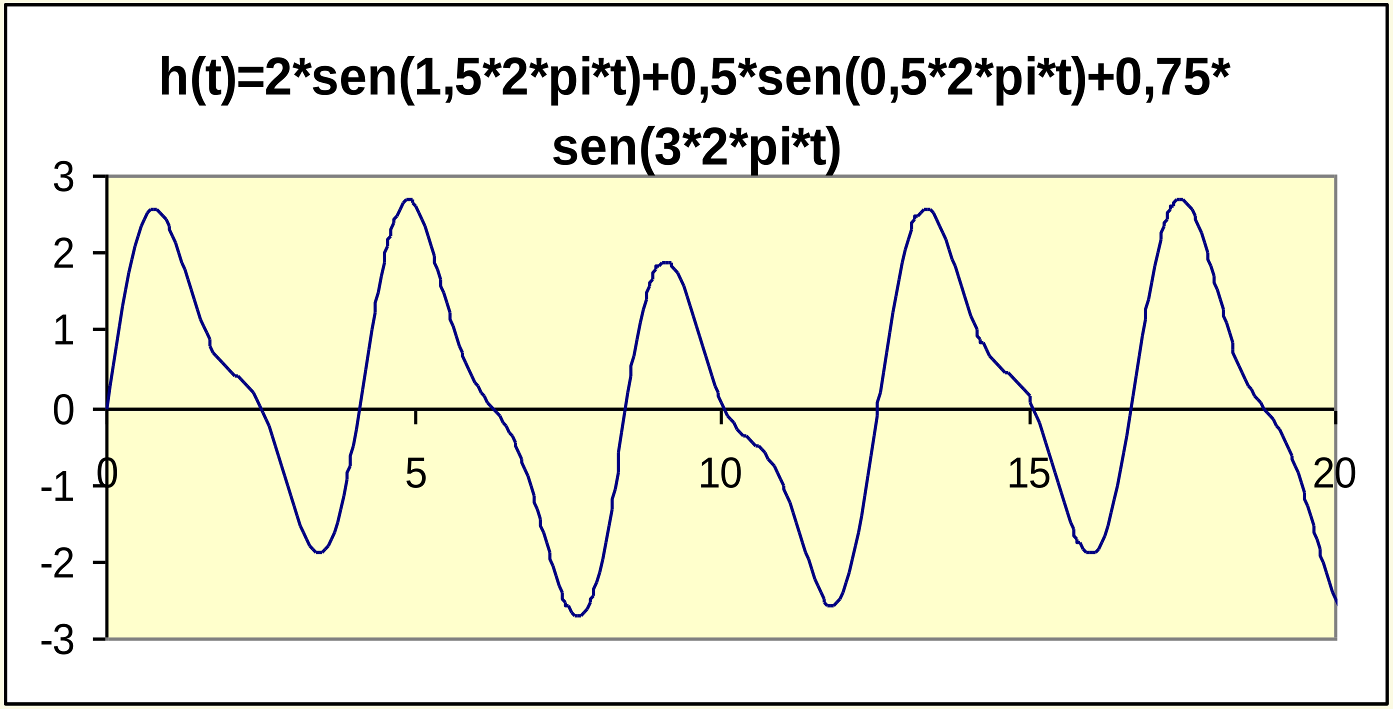
(Si t es el tiempo, T se mide en segundos y inversa 1/T, la **frecuencia**, se mide en ciclos /segundo o Hertzios).

Introducción

Dada una función periódica, podemos describirla por su valor en función del tiempo o por su **amplitud** en función de la frecuencia:



Introducción



En muchos casos resulta necesario obtener el “contenido en frecuencia” de una señal.

Introducción

Dado un proceso físico dependiente del tiempo, podemos describirlo:

1) En el dominio temporal: función *h(t)*

2) En el dominio de frecuencias: El proceso se especifica dando la **amplitud** *H* como una función de la **frecuencia** *f*. (*H* puede ser un número complejo, indicando la componente compleja la **fase**)

Se puede considerar que h(t) y H(f) son dos representaciones de la misma función.

Transformada de Fourier

Dada una función compleja dependiente del tiempo h(t), es posible obtener su representación en función de las frecuencias utilizando la **Transformada de Fourier:**

$$H(f) = \int_{-\infty}^{+\infty} h(t) e^{2\pi i f t} dt =$$

$$\int_{-\infty}^{+\infty} h(t)\big(\cos(2\pi f t) + i\,\mathrm{sen}(2\pi f t)\big)dt$$

Transformada de Fourier

Dada una función compleja expresada en función de la frecuencia H(f), es posible obtener su representación en función del tiempo utilizando la **Transformada inversa de Fourier:**

$$h(t) = \int_{-\infty}^{+\infty} H(f) e^{-2\pi i f t} df$$

Algunas propiedades...

Si h(t) es la transformada de H(f),

$$h(at) \Leftrightarrow \frac{1}{|a|} H\left(\frac{f}{a}\right)$$

Escalado en tiempo

$$\frac{1}{|b|} h\left(\frac{t}{b}\right) \Leftrightarrow H(bf)$$

Escalado en frecuencia

$$h(t - t_0) \Leftrightarrow H(f) e^{2\pi i f t_0}$$

desplazamiento en tiempo

$$h(t) e^{-2\pi i f_0 t} \Leftrightarrow H(f - f_0)$$

desplazamiento en frecuencia

Aplicación: Convolución

Dadas dos funciones h(t) y g(t), y sus correspondientes transformadas H(f) y G(f), la convolución de h y g se calcula como:

$$g * h = \int_{-\infty}^{+\infty} g(\tau) h(t-\tau) d\tau$$

Teorema de la convolución:

$$g * h \Leftrightarrow G(f) H(f)$$

Transformada Discreta de Fourier

Caso discreto

Habitualmente, no se conoce h(t), sino que se obtienen (miden, registran) valores de h en puntos equidistantes. Dado un intervalo de muestreo Δ, los puntos de muestreo serán:

$t_0=0 \longrightarrow y_0=h(t_0)$

$t_1=\Delta \longrightarrow y_1=h(t_1)$

$t_2=2\Delta \longrightarrow y_2=h(t_2)$

... ...

En general, $t_n=n\Delta$. (Tomaremos n par). El recíproco del intervalo de muestreo **$1/\Delta$** es la tasa (o frecuencia) de muestreo.

Caso discreto

Se asume que los datos son periódicos:

$$\ldots\quad y_{N-2}\ y_{N-1}\ \underbrace{y_0\ \ y_1\ \ y_2\ \ \ldots\ldots\ \ y_{N-1}}_{\text{Periodo}}\ y_0\ \ y_1\ \ldots$$

Caso discreto

Dados N puntos, con N par, vamos a ajustar esos N puntos a N funciones trigonométricas: N/2 senos y N/2 cosenos, de período T o de forma que T sea múltiplo de su periodo:

0 ciclos $\quad cos\frac{2\pi 0 t_r}{T} \quad sen\frac{2\pi 0 t_r}{T}$

1 ciclo $\quad cos\frac{2\pi 1 t_r}{T} \quad sen\frac{2\pi 1 t_r}{T}$

2 ciclos $\quad cos\frac{2\pi 2 t_r}{T} \quad sen\frac{2\pi 2 t_r}{T}$

$\vdots$

m=N/2 ciclos $\quad cos\frac{2\pi m t_r}{T} \quad sen\frac{2\pi m t_r}{T}$

= 0

Efecto del muestreo

Caso discreto

Dados N puntos, con N par: (t_r, y_r) r=0,1,...,N-1, queremos que para todos los puntos se satisfaga la ecuación:

$$y_r = \frac{1}{N}\sum_{k=0}^{m}\left(A_k \cos\left(\frac{2\pi k t_r}{T}\right) + B_k \operatorname{sen}\left(\frac{2\pi k t_r}{T}\right)\right)$$

$\Downarrow$

$$y_r = \frac{1}{N}\left(A_0 + \sum_{k=1}^{m-1}\left\{A_k \cos\left(\frac{2\pi k t_r}{T}\right) + B_k \operatorname{sen}\left(\frac{2\pi k t_r}{T}\right)\right\} + A_m \cos\left(\frac{2\pi m t_r}{T}\right)\right)$$

(1)

Periodos de cada seno: T/k

$\longrightarrow$ Frecuencias 1/T, 2/T, ..., m/T

Periodos de cada coseno: T/k

Caso discreto

Los N puntos están igualmente espaciados en el rango T

==> $t_r = rT/N$, r= 0,1,...,N-1

Sustituyendo en la ecuación **(1)**:

$$y_r = \frac{1}{N}\left(A_0 + \sum_{k=1}^{m-1}\left\{ A_k \cos\left(\frac{2\pi k t_r}{T}\right) + B_k \operatorname{sen}\left(\frac{2\pi k t_r}{T}\right)\right\} + A_m \cos\left(\frac{2\pi m t_r}{T}\right)\right)$$

Obtenemos:

$$y_r = \frac{1}{N}\left(A_0 + \sum_{k=1}^{m-1}\left\{ A_k \cos\left(\frac{2\pi k r}{N}\right) + B_k \operatorname{sen}\left(\frac{2\pi k r}{N}\right)\right\} + A_m \cos(\pi r)\right)$$

(2)

Caso discreto

Usando las expresiones:

$$\exp\left(\frac{i2\pi kr}{N}\right) = \cos\left(\frac{2\pi kr}{N}\right) + i\,\mathrm{sen}\left(\frac{2\pi kr}{N}\right) \quad k = 1,2,\cdots m-1;$$

$$\exp\left(\frac{i2\pi(N-k)r}{N}\right) = \exp\left(-\frac{i2\pi kr}{N}\right) \quad k = 1,2,\cdots m-1;$$

Se transforma **(2)** en:

$$y_r = \frac{1}{N}\sum_{k=0}^{N-1} Y_k \exp\left(\frac{i2\pi kr}{N}\right); \quad r = 0,1,2,\cdots, N-1 \qquad \textbf{(3)}$$

Donde $Y_0=A_0,$ $Y_m=A_m$ (m=N/2)

$Y_k= (A_k-iB_k)/2$, $Y_{N-k}= (A_k+iB_k)/2$, k=1,2,...,m-1

Caso discreto

Necesitamos hallar los valores Y_k: Usamos la siguiente propiedad ortogonal:

$$\sum_{r=0}^{N-1} \exp\left(\frac{i2\pi rj}{N}\right)\exp\left(\frac{-i2\pi rk}{N}\right) = \begin{cases} 0 & |j-k| \neq 0,\, N,\, 2N, \cdots \\ N & |j-k| = 0,\, N,\, 2N, \cdots \end{cases}$$

Multiplicando **(3)** por exp(-i2πrj/N) y sumando sobre los n valores de r:

$$Y_k = \sum_{r=0}^{N-1} y_r \exp\left(\frac{-i2\pi kr}{N}\right); \quad k = 0,1,2,\cdots, N-1 \quad \textbf{(4)}$$

TRANSFORMADA DISCRETA DE FOURIER

Caso discreto: Simplificación

Sea W_N la ***constante*** compleja exp(- i2π/N);

$$r = 1 \quad \Rightarrow \quad \exp\left(\frac{-i2\pi 1k}{N}\right) = W_N^{k}$$

$$r = 2 \quad \Rightarrow \quad \exp\left(\frac{-i2\pi 2k}{N}\right) = W_N^{2k}$$

En general,

$\exp\left(\dfrac{-i2\pi rk}{N}\right) = W_N^{rk}$,y **(4)** se puede escribir como:

$$Y_k = \sum_{r=0}^{N-1} y_r W_N^{kr}; \quad k = 0,1,2,\cdots,N-1$$

Caso discreto:

Sea W_N la ***constante*** compleja exp(- i2π/N); si construimos la matriz compleja:

$$F_N = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^{N-1} \\ 1 & W_N^2 & W_N^{2\cdot 2} & \cdots & W_N^{2\cdot(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2\cdot(N-1)} & \cdots & W_N^{(N-1)\cdot(N-1)} \end{pmatrix}$$

La DFT se puede expresar matricialmente como:

$$\begin{pmatrix} Y_0 \\ Y_1 \\ \vdots \\ Y_{N-1} \end{pmatrix} = F_N \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix}$$

Matriz de la DFT

Ejemplos:

$$F_1 = [1] \qquad F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

Propiedades de la matriz F_N (DFT)

1) La matriz F_N es simétrica (pero, si N>2, no es hermítica).

2) La matriz F_N cumple $F_N^H F_N = nI_n$

$\Rightarrow$ La matriz $\dfrac{F_N}{\sqrt{n}}$ es una transformación unitaria

Caso discreto: Transformada Inversa de Fourier

$$Y_k = \sum_{r=0}^{N-1} y_r \exp\left(\frac{-i2\pi kr}{N}\right); \quad k = 0,1,2,\cdots,N-1$$

$\Updownarrow$

TRANSFORMADA DISCRETA DE FOURIER

$$y_r = \frac{1}{N}\sum_{k=0}^{N-1} Y_k \exp\left(\frac{i2\pi kr}{N}\right); \quad r = 0,1,2,\cdots,N-1$$

$\Updownarrow$

TRANSFORMADA DISCRETA INVERSA DE FOURIER

Aplicación de la DFT: Convolución discreta

Si $\{h_k\}$ y $\{g_k\}$ son secuencias bi-infinitas con periodo n ($h_k = h_{k+\alpha n}$, para todo par de enteros α y k, la convolución de $\{h_k\}$ y $\{g_k\}$ es otra secuencia $\{f_k\}$ definida como:

$$f_k = \sum_{j=0}^{n-1} g_j h_{k-j}$$

Si H y G son las DFTs de las secuencias h y g respectivamente, la DFT de la convolución f es F= H·G (producto elemento a elemento).

Mas aplicaciones: Correlaciones, análisis de señal, resolución de ciertos S.E.L., resolución de ciertas P.D.Es, etc.

Transformada Rápida de Fourier

Transformada Rápida de Fourier

- Para un número de datos que sea potencia de 2, se desarrolló un algoritmo que permite calcular la DFT en $O(2n \cdot \log_2 n)$ operaciones. (Cooley-Tukey, 1965).

- Se han desarrollado otras versiones para cantidades de datos que no son potencia de 2.

Transformada Rápida de Fourier; Lema de Dannielson-Lanczos

Tomemos los N puntos (N potencia de 2), y los dividimos en pares e impares:

$$\left.\begin{aligned} u_r &= y_{2r} \\ v_r &= y_{2r+1} \end{aligned}\right\} \quad r = 0,1,\cdots,(\frac{N}{2}-1)$$

Calculamos las DFTs de los u_r y v_r, por separado:

$$U_k = \sum_{r=0}^{\frac{N}{2}-1} u_r \exp\left(\frac{-i2\pi kr}{(N/2)}\right); \quad k = 0,1,2,\cdots,(N/2-1)$$

$$V_k = \sum_{r=0}^{\frac{N}{2}-1} v_r \exp\left(\frac{-i2\pi kr}{(N/2)}\right); \quad k = 0,1,2,\cdots,(N/2-1)$$

Transformada Rápida de Fourier; Lema de Dannielson-Lanczos

La DFT de la secuencia original es:

$$Y_k = \sum_{r=0}^{N-1} y_r \exp\left(\frac{-i2\pi kr}{N}\right) = \sum_{r=0}^{N/2-1} y_{2r} \exp\left(\frac{-i2\pi k(2r)}{N}\right) +$$

$$\sum_{r=0}^{N/2-1} y_{(2r+1)} \exp\left(\frac{-i2\pi k(2r+1)}{N}\right)$$

Sustituyendo y_{2r} por u_r y y_{2r+1} por v_r:

$$Y_k = \sum_{r=0}^{N/2-1} u_r \exp\left(\frac{-i2\pi kr}{(N/2)}\right) + \exp\left(\frac{-i2\pi k}{N}\right) \sum_{r=0}^{N/2-1} v_r \exp\left(\frac{-i2\pi kr}{(N/2)}\right)$$

Transformada Rápida de Fourier; Lema de Dannielson-Lanczos

Por tanto, tenemos que :

$$Y_k = U_k + \exp\left(\frac{-i2\pi k}{N}\right)V_k; \quad k = 0,1,\cdots,\frac{N}{2}-1$$

O, lo que es lo mismo:

$$\boxed{Y_k = U_k + W_n^k V_k;} \quad k = 0,\cdots,\frac{N}{2}-1$$

Usando que U_k y V_k son periódicos en k, tenemos también que:

$$\boxed{Y_{k+n/2} = U_k - W_n^k V_k;} \quad k = 0,\cdots,\frac{N}{2}-1$$

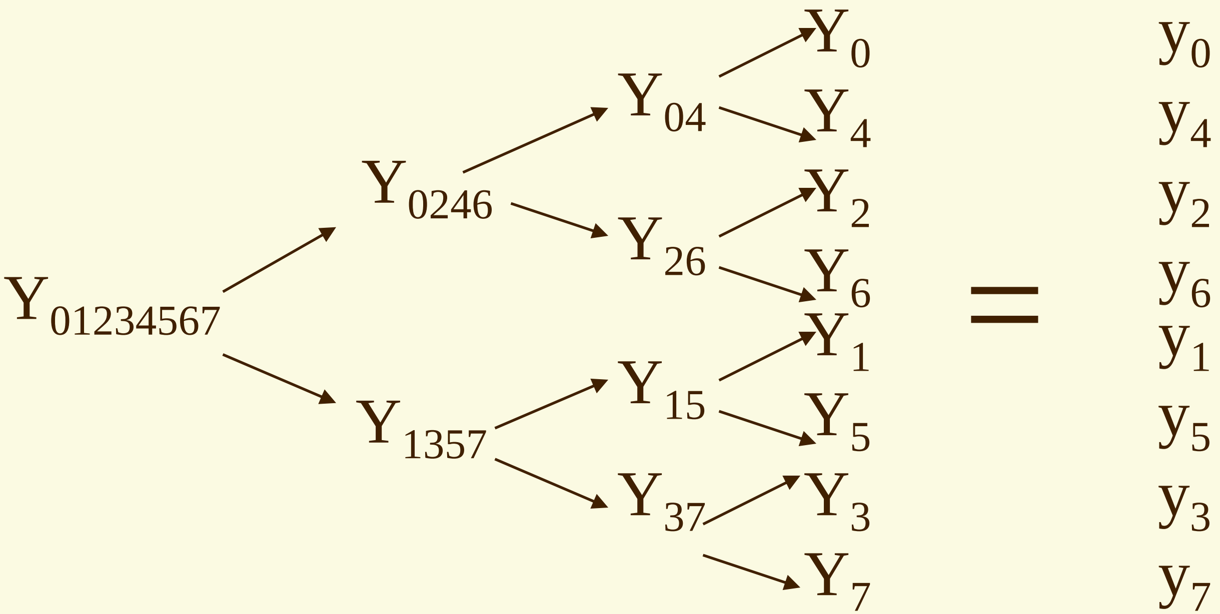
Transformada Rápida de Fourier; Lema de Dannielson-Lanczos

Ejemplo:

Tenemos 8 puntos, $y_0, y_1, ..., y_{7.}$

¿Como calculamos la DFT de $y_0, y_1, ..., y_{7.}$ =>

$$(Y_0, Y_1, ..., Y_7)=Y_{012...7}\ ?$$



Danielson-Lanczos se aplica recursivamente

Transformada Rápida de Fourier; Bit-Reversal Algorithm

Ordenación para poder aplicar Danielson-Lanczos de forma recursiva:

y_0		y_0
y_1	?	y_4
y_2		y_2
y_3		y_6
y_4		y_1
y_5		y_5
y_6		y_3
y_7		y_7

Transformada Rápida de Fourier; Bit-Reversal Algorithm

Ordenación para poder aplicar Danielson-Lanczos de forma recursiva:

1) Obtener los bits del índice:

y_0		y_{000}
y_1		y_{001}
y_2		y_{010}
y_3		y_{011}
y_4	$\Rightarrow$	y_{100}
y_5		y_{101}
y_6		y_{110}
y_7		y_{111}

Transformada Rápida de Fourier; Bit-Reversal Algorithm

2) Invertir los bits del índice:

y_0		y_{000}		y_{000}
y_1		y_{001}		y_{100}
y_2		y_{010}		y_{010}
y_3		y_{011}		y_{110}
y_4	$\Rightarrow$	y_{100}	$\Rightarrow$	y_{001}
y_5		y_{101}		y_{101}
y_6		y_{110}		y_{011}
y_7		y_{111}		y_{111}

Transformada Rápida de Fourier; Bit-Reversal Algorithm

3) El entero correspondiente a los nuevos bits da la ordenación:

y_0	y_{000}	y_{000}	y_0
y_1	y_{001}	y_{100}	y_4
y_2	y_{010}	y_{010}	y_2
y_3	y_{011}	y_{110}	y_6
y_4	y_{100}	y_{001}	y_1
y_5	y_{101}	y_{101}	y_5
y_6	y_{110}	y_{011}	y_3
y_7	y_{111}	y_{111}	y_7

Transformada Rápida de Fourier
Algoritmo:

Dado un input de N datos (N potencia entera de 2, p. Ej. 2^K)

1) Reordenar usando el algoritmo “bit reversal”

2) Aplicar de forma recursiva el algoritmo de Danielson-Lanczos, con K etapas

FFT de Cooley-Tukey

Coste: $O(K \cdot N) = O(\log_2(N) \cdot N)$

Transformada Rápida de Fourier
FFT recursiva:

```
function y=fft(x)
n=length(x);    % n debe ser potencia de 2
if n==1
    y=x
else
   m=n/2
   w=exp(-2πi/n); omega=diag(1,w,...,w^(m-1))
   zt=fft(x(0:2:n-1)); zb=omega*fft(x(1:2:n-1))
```

$$y = \begin{bmatrix} I_m & I_m \\ I_m & -I_m \end{bmatrix} \begin{bmatrix} zt \\ zb \end{bmatrix}$$

```
end
```

Transformada Rápida de Fourier
Versión Matricial

Producto de Kronecker: Si $A \in C^{p,q}$ y $B \in C^{m,n}$

$$A \otimes B = \begin{bmatrix} a_{0,0}B & \cdots & a_{0,q-1}B \\ \vdots & & \vdots \\ a_{p-1,0}B & \cdots & a_{p-1,q-1}B \end{bmatrix}$$

Ojo a los subíndices!

$$A \otimes B \in C^{pm,qn}$$

Caso particular:

$$I_n \otimes B = \begin{pmatrix} B & 0 & \cdots & 0 \\ 0 & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B \end{pmatrix}$$

Propiedad:

$$\left(A \otimes B\right)^T = A^T \otimes B^T$$

Transformada Rápida de Fourier Versión Matricial

Matrices permutación: Intercambiando columnas de la matriz Identidad

-Dada una reordenación del vector v=0:n-1, la matriz P_v se define como:

$$P_v = I_n(:, v)$$

-Propiedad:

$$P_v^T P_v = I_n$$

Transformada Rápida de Fourier Versión Matricial

Recordemos:

$$F_N = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^{N-1} \\ 1 & W_N^2 & W_N^{2\cdot 2} & \cdots & W_N^{2\cdot(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2\cdot(N-1)} & \cdots & W_N^{(N-1)\cdot(N-1)} \end{pmatrix}$$

Buscamos una descomposición de la matriz F_N a partir de $F_{N/2}$, de $F_{N/2}$ a partir de $F_{N/4}$, y así sucesivamente hasta llegar al nivel 1.

Transformada Rápida de Fourier
Versión Matricial: ejemplo

Sean F_4: $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$ y F_2: $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Reordenamos F_4 según la permutación $\Pi_{4:}$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = I_n(:,[0,2,1,3])$$

Mueve las columnas pares (0,2) delante y las impares detrás (1,3):

Transformada Rápida de Fourier Versión Matricial: ejemplo

$$F_4\Pi_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & -1 & -i & i \\ \hline 1 & 1 & -1 & -1 \\ 1 & -1 & i & -i \end{array}\right]$$

Vamos a mirar esta matriz como una matriz por bloques 2 por 2.

En este caso, $W_n = W_4 = \exp(-2\pi i/4) = -i$.

Definimos Ω_2 = diag (1, W_4) = $\begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix}$ y recordamos que:

$$F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

(Está en la versión permutada de $F_4\Pi_4$)

Transformada Rápida de Fourier
Versión Matricial: ejemplo

Además, tenemos:

$$\Omega_2 F_2 = \begin{pmatrix} 1 & 1 \\ -i & -i \end{pmatrix} \quad \text{y que} \quad -\Omega_2 F_2 = \begin{pmatrix} -1 & -1 \\ i & -i \end{pmatrix}$$

Entonces:

$$F_4 \Pi_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -i & i \\ 1 & 1 & -1 & -1 \\ 1 & -1 & i & -i \end{bmatrix} = \begin{pmatrix} F_2 & \Omega_2 F_2 \\ F_2 & -\Omega_2 F_2 \end{pmatrix}$$

$$= \begin{pmatrix} I_2 & \Omega_2 \\ I_2 & -\Omega_2 \end{pmatrix} \begin{pmatrix} F_2 & 0 \\ 0 & F_2 \end{pmatrix} = \begin{pmatrix} I_2 & \Omega_2 \\ I_2 & -\Omega_2 \end{pmatrix} \left(I_2 \otimes F_2 \right)$$

Transformada Rápida de Fourier Versión Matricial

En general, si n=2m, tenemos la permutación par-impar de orden n:

$$\Pi_n = I_n(:, v) \quad v = \begin{bmatrix} 0:2:n-1 \\ 1:2:n-1 \end{bmatrix}$$

Y definiendo Ω_m como:

$$\Omega_m = diag\begin{pmatrix} 1 & \omega_n & \cdots & \omega_n^{m-1} \end{pmatrix},$$

entonces

$$F_n \Pi_n = \begin{pmatrix} F_m & \Omega_m F_m \\ F_m & -\Omega_m F_m \end{pmatrix} = \begin{pmatrix} I_m & \Omega_m \\ I_m & -\Omega_m \end{pmatrix} \left(I_2 \otimes F_m \right)$$

Transformada Rápida de Fourier
Ejemplo, n=16

Si r=2s, definimos la matriz “Mariposa” B_r como:

$$B_r = \begin{pmatrix} I_s & \Omega_s \\ I_s & -\Omega_s \end{pmatrix}, \quad donde$$

$$\Omega_s = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & w_s & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_r^{s-1} \end{pmatrix}$$

Transformada Rápida de Fourier
Ejemplo, n=16

La DFT del vector x, desde 0 hasta 15, se obtiene multiplicando la matriz F_n por el vector x: $F_{16}x$

F_{16} se obtiene a partir de las dos matrices DFT de orden 8:

$$F_{16}x = \begin{pmatrix} I_8 & \Omega_8 \\ I_8 & -\Omega_8 \end{pmatrix} \begin{bmatrix} F_8 x(0:2:15) \\ F_8 x(1:2:15) \end{bmatrix} = B_{16} \begin{bmatrix} F_8 x(0:2:15) \\ F_8 x(1:2:15) \end{bmatrix}$$

$$= (I_1 \otimes B_{16}) \begin{bmatrix} F_8 x(0:2:15) \\ F_8 x(1:2:15) \end{bmatrix}$$

Transformada Rápida de Fourier
Ejemplo, n=16

Las dos DFTs de orden 8, una aplicada a las componentes pares (x(0:2:15)) y otra a las impares (x(1:2:15)) se obtienen respectivamente de DFTs de orden 4:

$\boxed{0\ \ 2\ \ 4\ \ 6\ \ 8\ \ 10\ \ 12\ \ 14}$ (=0:2:15)

pares

$\boxed{0\ \ 4\ \ 8\ \ 12}$

||

(0:4:15)

impares

$\boxed{2\ \ 6\ \ 10\ \ 14}$

||

(2:4:15)

$$F_8 x(0:2:15) = \begin{pmatrix} I_4 & \Omega_4 \\ I_4 & -\Omega_4 \end{pmatrix} \begin{bmatrix} F_4 x(0:4:15) \\ F_4 x(2:4:15) \end{bmatrix}$$

Transformada Rápida de Fourier Ejemplo, n=16

La DFT de orden 8 impar:

$\boxed{1\ \ 3\ \ 5\ \ 7\ \ 9\ \ 11\ \ 13\ \ 15}$ (=1:2:15)

pares	impares
$\boxed{1\ \ 5\ \ 9\ \ 13}$	$\boxed{3\ \ 7\ \ 11\ \ 15}$
‖	‖
(1:4:15)	(3:4:15)

$$F_8 x(1:2:15) = \underbrace{\begin{pmatrix} I_4 & \Omega_4 \\ I_4 & -\Omega_4 \end{pmatrix}}_{\| \atop B_8} \begin{bmatrix} F_4 x(1:4:15) \\ F_4 x(3:4:15) \end{bmatrix}$$

Transformada Rápida de Fourier Ejemplo, n=16

Juntando las dos:

$$F_8x(0:2:15)=\begin{pmatrix} I_4 & \Omega_4 \\ I_4 & -\Omega_4 \end{pmatrix}\begin{bmatrix} F_4x(0:4:15) \\ F_4x(2:4:15) \end{bmatrix}$$

$$F_8x(1:2:15)=\begin{pmatrix} I_4 & \Omega_4 \\ I_4 & -\Omega_4 \end{pmatrix}\begin{bmatrix} F_4x(1:4:15) \\ F_4x(3:4:15) \end{bmatrix}$$

$$\begin{bmatrix} F_8x(0:2:15) \\ F_8x(1:2:15) \end{bmatrix}=\begin{bmatrix} I_4 & \Omega_4 & 0 & 0 \\ I_4 & -\Omega_4 & 0 & 0 \\ 0 & 0 & I_4 & \Omega_4 \\ 0 & 0 & I_4 & -\Omega_4 \end{bmatrix}\begin{bmatrix} F_4x(0:4:15) \\ F_4x(2:4:15) \\ F_4x(1:4:15) \\ F_4x(3:4:15) \end{bmatrix}$$

Transformada Rápida de Fourier
Ejemplo, n=16

Y obtenemos:

$$\begin{bmatrix} F_8 x(0:2:15) \\ F_8 x(1:2:15) \end{bmatrix} = \left(I_2 \otimes B_8\right) \begin{bmatrix} F_4 x(0:4:15) \\ F_4 x(2:4:15) \\ F_4 x(1:4:15) \\ F_4 x(3:4:15) \end{bmatrix}$$

Transformada Rápida de Fourier
Ejemplo, n=16

Análogamente, F_4 se obtiene a partir de F_2:

$$\begin{bmatrix} F_4 x(0:4:15) \\ F_4 x(2:4:15) \\ F_4 x(1:4:15) \\ F_4 x(3:4:15) \end{bmatrix} = \left(I_4 \otimes B_4\right) \begin{bmatrix} F_2 x(0:8:15) \\ F_2 x(4:8:15) \\ F_2 x(2:8:15) \\ F_2 x(6:8:15) \\ F_2 x(1:8:15) \\ F_2 x(5:8:15) \\ F_2 x(3:8:15) \\ F_2 x(7:8:15) \end{bmatrix}$$

Transformada Rápida de Fourier
Ejemplo, n=16

Y previamente se habrá hecho la etapa de orden 1:

$$\begin{bmatrix} F_2 x(0:8:15) \\ F_2 x(4:8:15) \\ F_2 x(2:8:15) \\ F_2 x(6:8:15) \\ F_2 x(1:8:15) \\ F_2 x(5:8:15) \\ F_2 x(3:8:15) \\ F_2 x(7:8:15) \end{bmatrix} = \left(I_8 \otimes B_2\right) \begin{bmatrix} x(0) \\ x(8) \\ x(4) \\ x(12) \\ x(2) \\ x(10) \\ x(6) \\ x(14) \\ x(1) \\ x(9) \\ x(5) \\ x(13) \\ x(3) \\ x(11) \\ x(7) \\ x(15) \end{bmatrix}$$

Vector x “bit-reversed”

||

$$P_{16}^T x$$

Transformada Rápida de Fourier
Ejemplo, n=16

Para obtener F_{16} x:

1) Permutar el vector x $\longrightarrow$ $P_{16}^{T}x$

2) Premultiplicar el resultado por $I_8 \otimes B_2$

3) Premultiplicar el resultado por $I_4 \otimes B_4$

4) Premultiplicar el resultado por $I_2 \otimes B_8$

5) Premultiplicar el resultado por $I_1 \otimes B_{16}$

$$F_{16}x = (I_1 \otimes B_{16})(I_2 \otimes B_8)(I_4 \otimes B_4)(I_8 \otimes B_2)P_{16}^{T}x$$

$$F_{16} = (I_1 \otimes B_{16})(I_2 \otimes B_8)(I_4 \otimes B_4)(I_8 \otimes B_2)P_{16}^{T}$$

Transformada Rápida de Fourier
Versión Matricial

En general, se puede obtener la matriz DFT de orden n F_n (con elementos $[f_{i,j}] = w_n^{ij}$, con i,j variando entre 0 y n-1) mediante el siguiente producto de matrices:

$$F_n P_n = (I_1 \otimes B_n)(I_2 \otimes B_{n/2}) \cdots (I_{n/2} \otimes B_2)$$

Cada matriz del producto es DISPERSA (Sólo dos elementos distintos de 0 en cada fila)

Transformada Rápida de Fourier
Factorización de Cooley-Tukey (radio 2)

Si $n=2^t$ entonces:

$$F_n = A_t \cdots A_1 P_n^T$$

Donde P_n es la permutación "bit-reversal", y:

$$A_q = I_r \otimes B_L \qquad ; L = 2^q, r = n/L$$

$$B_L = \begin{bmatrix} I_{L^*} & \Omega_{L^*} \\ I_{L^*} & -\Omega_{L^*} \end{bmatrix}; \qquad L^* = L/2,$$

$$\Omega_{L^*} = diag\begin{pmatrix} 1 & \omega_L & \cdots & \omega_L^{L^*-1} \end{pmatrix}; \qquad \omega_L = \exp(-2\pi i / L)$$

Transformada Rápida de Fourier Decimación en tiempo y en frecuencia

La FFT de Cooley-Tukey empieza por permutar el vector (coste no trivial, 10-30% del coste total), todavía señal TEMPORAL. (Decimation-in-Time)

Existe la posibilidad de permutar el vector en el espacio de frecuencias: (decimation in frequency)

F_n es SIMÉTRICA

$$F_n = F_n^T \longrightarrow F_n x = F_n^T x$$

Transformada Rápida de Fourier
Decimación en tiempo y en frecuencia

Si $F_n = A_t \cdots A_1 P_n^T$ entonces

$$\boxed{F_n = F_n^T = P_n A_1^T \cdots A_t^T}$$

Factorización de **Gentleman-Sande**

$$A_q = I_r \otimes B_L \Rightarrow A_q^T = I_r^T \otimes B_L^T, \quad B_L^T = \begin{bmatrix} I_{L^*} & I_{L^*} \\ \Omega_{L^*} & -\Omega_{L^*} \end{bmatrix}$$

Si, por ejemplo, se utiliza la FFT para realizar la convolución de dos señales es posible eliminar por completo la fase de bit-reversal

Transformada Rápida de Fourier
Versión Matricial

Existen otras versiones de la FFT de radio 2:
Stockham, Stockam traspuesta, Pease, etc.

También existen otras versiones de buen rendimiento para potencias distintas de 2, especialmente cuando n es “altamente compuesto”.

Implementación de la FFT Cooley-Tukey

- Bit-Reversal.

- Cálculo de los pesos.

- Lema Dannielson-Lanczos (o aplicación de la matriz "Mariposa".

Consideraciones:

Coste a priori (cantidad de operaciones a efectuar).
Eficiencia en acceso a memoria (stride).
Error de redondeo.
Uso o no de "Espacio de Trabajo" y de "sobreescritura".

Implementación de la FFT
“Bit reversal”

Dado un entero n potencia de 2 ($n=2^t$), y dado k entero $0 \le k < n$, hallar el entero $r_n(k)$, $0 \le r_n(k) < n$, que nos da el valor “bit reversed” de k.

$$k = b_0 + b_1 2 + b_2 2^2 + \cdots + b_{t-2} 2^{t-2} + b_{t-1} 2^{t-1}$$

$$\Downarrow$$

$$r_n(k) = b_0 2^{t-1} + b_1 2^{t-2} + \cdots + b_{t-2} 2 + b_{t-1}$$

Hay que repetirlo para todo k, con $0 \le k \le n$,

Implementación de la FFT
“Bit reversal”

$r_n(k)$ se puede interpretar como un polinomio de grado t-1 con coeficientes $(b_0, b_1, ..., b_{t-1})$ y evaluado en el punto 2

Se puede utilizar el algoritmo de Horner-Ruffini, para calcular j= $r_n(k)$

```
j=0;
for q=0:t-1
        Determinar el bit b_q
        j=2*j+b_q
end
```

Implementación de la FFT
"Bit reversal"

Cada bit se puede determinar dividiendo por dos (desplazando) y restando, para obtener el bit de la derecha, y repitiendo el proceso:

Ejemplo:

Dado $k=(0\ 1\ 0\ 1)_2$

$s=k/2 = (0\ 0\ 1\ 0)_2 \longleftarrow$ División entera

y $k-2s = (0\ 0\ 0\ 1)_2$

El bit de la derecha de k es 1

Implementación de la FFT "Bit reversal"

Algoritmo completo para, dado k, obtener $r_n(k)$

```
function j=bit_rev(k)
j=0; m=k;
for q=0:t-1
        s=floor(m/2)  % b_q  = m-2s
        j=2*j+(m-2*s)
        m=s
end
```

Coste $O(\log_2 n)$ operaciones

Implementación de la FFT “Bit reversal”

Algoritmo completo para obtener $r_n(k)$ para todo k, $0 \leq k < n$

```
for k=0:n-1
  j=bit_rev(k)
  if j>k
     swap(x(j), x(k))
  end
end
```

Coste $O(n\log_2 n)$ operaciones

Coste no trivial, puede ser 10-30% de tiempo total de cómputo.

Implementación de la FFT
Cálculo de los pesos ($W_n^{kj} = \exp(-2kj\pi i/n)$)

Existen varias posibilidades, de diferente coste y precisión:

1) Llamadas directas a la función seno, coseno

2) Multiplicación repetida $\omega_n^{kj} = \omega \cdot \omega_n^{(k-1)j}$

3) Escalado de subvector

4) Recursión con rotaciones de Givens

$\vdots$

Implementación de la FFT
Multiplicación por matriz "Mariposa"

Implementación del Producto y=Bz

$$\begin{bmatrix} y_t \\ y_b \end{bmatrix} = \begin{bmatrix} I_m & \Omega_m \\ I_m & -\Omega_m \end{bmatrix} \begin{bmatrix} z_t \\ z_b \end{bmatrix} = \begin{bmatrix} z_t + \Omega_m z_b \\ z_t - \Omega_m z_b \end{bmatrix}$$

```
m=L/2
for j=0:m-1
        τ=Ω_m(j)·z(j+m)
        y(j)=z(j)+ τ
        y(j+m)=z(j)- τ
end
```

No es posible "overwriting"

Implementación de la FFT
Multiplicación por matriz "Mariposa"

Calculando y(j+m) antes que y(j), es posible "overwriting"

```
m=L/2
for j=0:m-1
        τ=Ω_m(j)·z(j+m)
        z(j+m)=z(j)- τ
        z(j)=z(j)+ τ
end
```

Coste 10m flops

Implementación de la FFT
Actualización $x=A_q x$

Multiplicar $(I_r \otimes B_L)$ por $x \in C^n$, con $n=rL$

Versión “columnas” (“kj Butterfly update”)

```
r=n/L
m=L/2
for k=0:r-1
        for j=0:m-1
                τ=Ω_m(j)·x(kL+j+m)
                x(kL+j+m)=z(kL+j)- τ
                z(kL+j)=z(kL+j)+ τ
        end
end
```

Coste 5n flops

Implementación de la FFT Actualización $x=A_q x$

Multiplicar $(I_r \otimes B_L)$ por $x \in C^n$, con $n=rL$

Versión “filas” (“jk Butterfly update”)

```
r=n/L
m=L/2
for j=0:m-1
        for k=0:r-1
                τ=Ω_m(j)·x(kL+j+m)
                x(kL+j+m)=z(kL+j)- τ
                z(kL+j)=z(kL+j)+ τ
        end
end
```

Coste 5n flops

Implementación de la FFT
Actualización $x=A_q x$

Es posible calcular los pesos antes de las actualizaciones “Off-line paradigm” o dentro de los bucles “Online paradigm”.

Transformada Rápida de Fourier Comentarios sobre el algoritmo:

1) Sea n=p·m;
   La matriz DFT de orden n se puede expresar en función de las DFTs de orden p y la DFT de orden m.

Esto se puede generalizar para cualquier tamaño de matriz y número de divisores.

Dado un entero n, la DF de orden n será tanto más eficiente cuanto mas pequeños sean todos los divisores de n. (Cuando n es “highly composite”)

El peor caso es cuando n es grande y primo; en ese caso, el coste es O(n^2)

Transformada Rápida de Fourier
Altas prestaciones

1) Multiple FFT

Versión columnas: Fn·X, o Versión Filas X·Fn

Es posible reordenar todas las filas (o columnas) a mismo tiempo, amortizando coste

Es posible realizar la FFT de un vector muy grande convirtiéndolo en una matriz:

$$\left[F_n x\right]_{n_1 \times n_2} = \left\lfloor F_n(0:n_1-1, 0:n_2-1).*\left(F_{n_1} x^T_{n_2 \times n_1}\right)\right\rfloor F_{n_2}$$

Transformada Rápida de Fourier
Altas prestaciones

2) Transposición de matrices;
Muy importante para problema de vector grande y para FFTs multidimensionales.

La dificultad es el acceso por filas/columnas, sobre todo para matrices grandes cuyo tamaño es potencia de dos; se hace a bloques

Transformada Rápida de Fourier
Altas prestaciones

Sea n = p·m, 1<p<n; F_n se puede expresar en función de F_m y F_p

$$F_n\ \Pi_{p,n} = \left(F_p \otimes I_m\right) diag\ \left(I_m, \Omega_{p,m}\ , \cdots,\ \Omega_{p,m}^{p-1}\right)\left(\ I_p \otimes F_m\ \right)$$

Donde $\Omega_{p,n}$ =diag (1, w_n,…,w_n^{m-1})

Y $\Pi_{p,n}$ es la permutación «barajado perfecto»

$$\Pi_{p,n}\mathrm{x}=\begin{pmatrix} x(0{:}\,p{:}\,n-1) \\ x(1{:}\,p{:}\,n-1) \\ \vdots \\ x(p-1{:}\,p{:}\,n-1) \end{pmatrix}$$

Transformada Rápida de Fourier
Altas prestaciones

Ejemplo; sea n = 96, p=4, m=24

$$\Pi^T_{4,24}\ \mathrm{x} = \begin{pmatrix} x(0:4:95) \\ x(1:4:95) \\ x(2:4:95) \\ x(3:4:95) \end{pmatrix}$$

$$(\ I_4 \otimes F_{24}\ )\ \ \Pi^T_{4,24}\ \mathrm{x} = \begin{pmatrix} F_{24}x(0:4:95) \\ F_{24}x(1:4:95) \\ F_{24}x(2:4:95) \\ F_{24}x(3:4:95) \end{pmatrix}$$

Transformada Rápida de Fourier
Altas prestaciones

$$diag\left(I_{24}, \Omega_{4,24}, \cdots, \Omega_{4,24}^{3}\right)(I_4 \otimes F_{24})\ \Pi_{4,24}^{T}\ \mathrm{x} =$$

$$\begin{pmatrix} F_{24}x(0:4:95) \\ \Omega_{4,24} \cdot F_{24}x(1:4:95) \\ \Omega_{4,24}^{2} \cdot F_{24}\, x(2:4:95) \\ \Omega_{4,24}^{3} \cdot F_{24}x(3:4:95) \end{pmatrix}$$

Este vector columna se reordena como un vector con p=24 filas y m=4 columnas; A cada fila de la matriz resultante, se le aplica la transformada por filas de orden 4 ➔

$$(F_4 \otimes I_{24})$$

Transformada Rápida de Fourier
Altas prestaciones

Algoritmo FFT recursiva de radio general n

Funcion y=**genfft**(x,n)

w_n=exp(-2*pi*i/n);

si n primo

y=Fn*x;

si no

{ selecciona p, divisor no trivial de n; m=n/p;

Ω =diag (1, w_n,…,w_n^{m-1})

/* z=$diag\left(I_m, \Omega_{p,m}, \cdots, \Omega_{p,m}^{p-1}\right)\left(I_p \otimes F_m\right)\Pi_{p,n}^{T}$x */

for j=0:p-1

z(j·m:(j+1)·m-1)= Ω^j **genfft**(x(j:p:n-1),m)

end

Transformada Rápida de Fourier
Altas prestaciones

```
/*     y=( Fp ⊗ Im )z ⇔ ym,p =zm,p Fp    */
        for j=0:m-1
              y(j:m:n-1)=  genfft(x(j:m:n-1),p)
        end
End   /* del if */
End /*de genfft */
```

$$y = (F_p \otimes I_m) z \Leftrightarrow y_{m,p} = z_{m,p} F_p$$

En el segundo bucle hay una fft de multiples filas

Transformada Rápida de Fourier de vector muy grande

Supongamos tamaño del vector n=n1*n2; aplicando la fórmula para m=n·p, y reorganizando, teníamos:

$$\left[F_n x\right]_{n_1 \times n_2} = \left\lfloor F_n(0:n_1-1, 0:n_2-1).*\left(F_{n_1}\, x^T_{n_2 \times n_1}\right)\right\rfloor F_{n_2}$$

Que requiere una FFT por filas, una por columnas, un escalado y una transposición

Transformada Rápida de Fourier de vector muy grande, por columnas (metodo de 6 pasos)

Transponiendo la parte derecha, obtenemos

$$[F_n x]_{n_1 \times n_2} = \left[F_{n_1} \left[F_n(0{:}\,n_1 - 1, 0{:}\,n_2 - 1) .* \left(F_{n_1} x^T_{n_1 \times n_2} \right) \right]^T \right]^T$$

Que es la base para el método de 6 pasos:

$x_{n_1 \times n_2} = x^T_{n_2 \times n_1}$

$x_{n_1 \times n_2} = F_{n_1} x_{n_1 \times n_2}$

$x_{n_1 \times n_2} = F_n(0{:}\,n_1 - 1, 0{:}\,n_2 - 1) .* \, x_{n_1 \times n_2}$

$x_{n_2 \times n_1} = x^T_{n_1 \times n_2}$

$x_{n_2 \times n_1} = F_{n_2} x_{n_2 \times n_1}$

$x_{n_1 \times n_2} = x^T_{n_2 \times n_1}$

Transformada Rápida de Fourier de vector muy grande, por filas (metodo de 4 pasos)

Utilizando la simetría de las matrices DFT

$$[F_n x]_{n_2 \times n_1} = \left[F_n(0{:}\,n_1 - 1, 0{:}\,n_2 - 1) .* \left(x_{n_1 \times n_2} F_{n_2}\right)\right]^T F_{n_1}$$

Que es la base para el método de 4 pasos:

$x_{n_1 \times n_2} = x_{n_1 \times n_2} F_{n_2}$

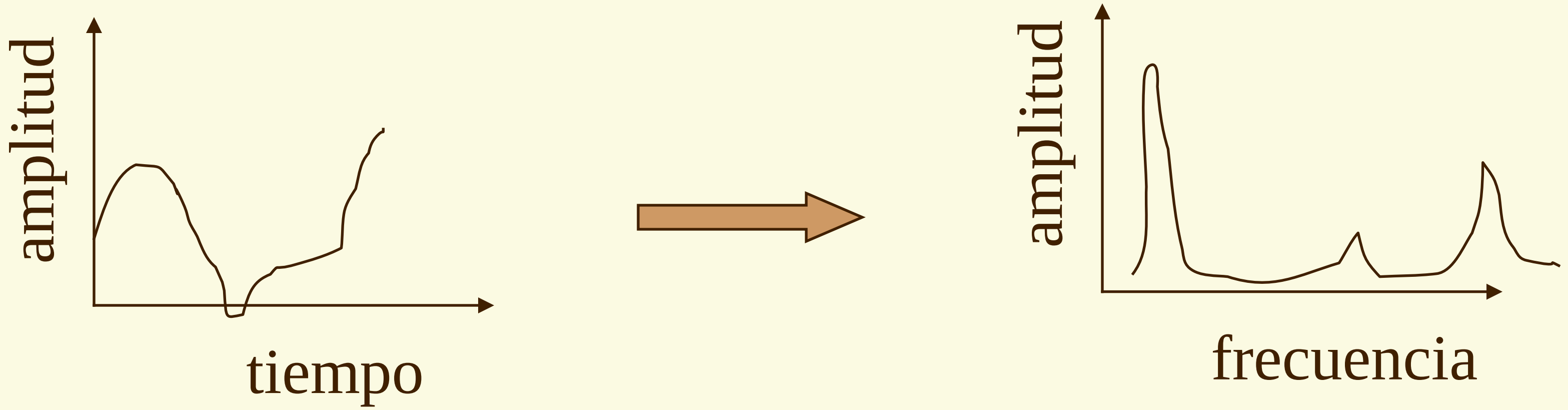
$x_{n_1 \times n_2} = F_n(0{:}\,n_1 - 1, 0{:}\,n_2 - 1) .* x_{n_1 \times n_2}$

$x_{n_2 \times n_1} = x^T_{n_1 \times n_2}$

$x_{n_2 \times n_1} = x_{n_2 \times n_1} F_{n_1}$

Short-Time Fourier Transform
Transformada de Gabor

Transformada Discreta de Fourier
Inconvenientes



La FT y DFT son herramientas válidas para examinar señales ESTACIONARIAS.

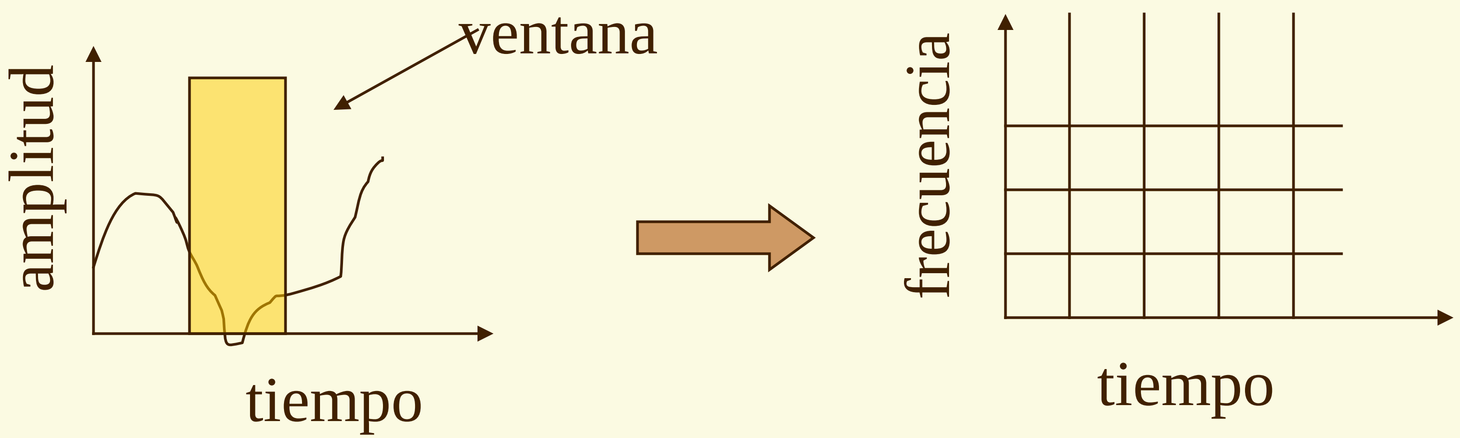
Transformada Discreta de Fourier
Inconvenientes

La DFT o la FT nos dicen si una frecuencia determinada aparece en una señal, pero no dicen CUANDO aparecen:

Buena resolución de frecuencia, pero ninguna resolución temporal.
(Ejemplo del tutorial).

Necesitamos una herramienta para detectar cuando aparecen fenómenos no estacionarios en la señal

Short -Time Transformada Discreta de Fourier: Ventanas



Se multiplica la función “ventana” por la señal a analizar, y al resultado se le aplica la FT o DFT.

La función ventana debe ser 0 salvo en un pequeño intervalo