

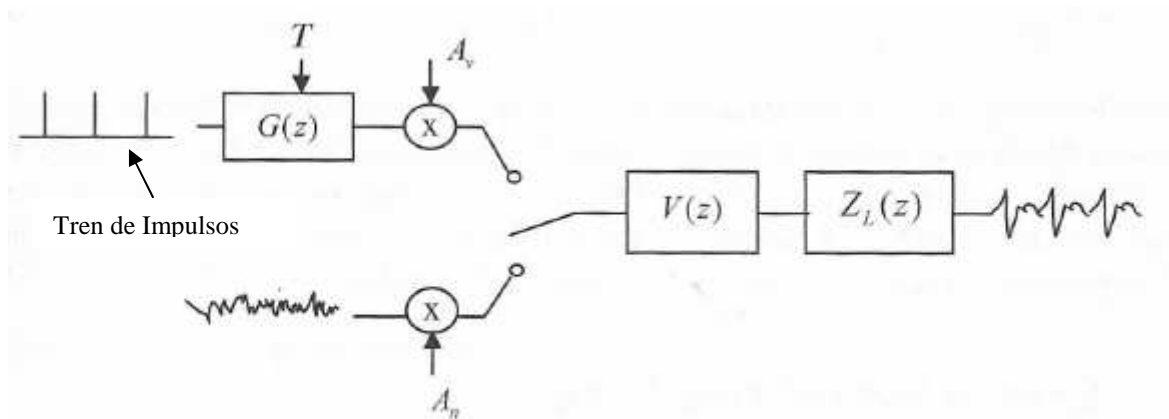
PROCESAMIENTO DIGITAL DE SEÑALES

El procesamiento digital de señales es una área que involucra el estudio de algoritmos y ciertas técnicas involucradas con el tratamiento de una señal digital, es decir una señal que fácilmente puede trabajarse en una computadora.

Como sabemos el habla es una señal sonora y es necesario procesarla para poder hacer su posterior reconocimiento, la necesidad de procesamiento surge pues sería ingenuo tratar de establecer algoritmos en la señal entera, para esto más bien se necesita capturar de alguna manera la información más importante de la señal en pocos datos.

Modelo Fuente Filtro

Podemos establecer un modelo para la producción del habla, a este modelo lo llamaremos el modelo fuente filtro, para esto podemos decir que el habla es una combinación del aire que sale de los pulmones y vibra en las cuerdas vocales para posteriormente entrar en un proceso de convolución con el tracto vocal que está formado por la cavidad resonante de la boca, como también se tendrá que tener en cuenta los efectos de los labios.



$G(z)$ = Efectos de la Glotis, generalmente las diferencias que ocurren en la forma de onda glotal (básicamente son diferentes montos de énfasis de bajas frecuencias), son una de las principales fuentes de diferencias entre los hablantes

A_v, T = Periodo y Amplitud para la excitación (fuente)

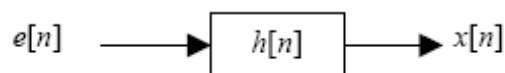
A_n = Amplitud para el ruido aleatorio

$V(z)$ = Efecto del tracto vocal, la longitud del tracto vocal tiene un efecto en la formación de las resonancias y es también un fuente para las diferencias entre hablantes.

$Z_L(z)$ = Efecto de los labios (énfasis en altas frecuencias)

El modelo es simplificado agrupando $G(z)$, $V(z)$ y $Z_L(z)$ para sonidos sonoros en $H(z)$ y $V(z)$ y $Z_L(z)$ en $H(z)$ para sonidos no sonoros

En general obtendremos un modelo fuente filtro de la siguiente manera



Donde $e[n]$ representará la fuente o excitación, es decir el flujo del aire en las cuerdas vocales, $h[n]$ será el filtro, es decir las resonancias del tracto vocal que cambian en el tiempo, $x[n]$ será la señal de salida en este caso el habla.

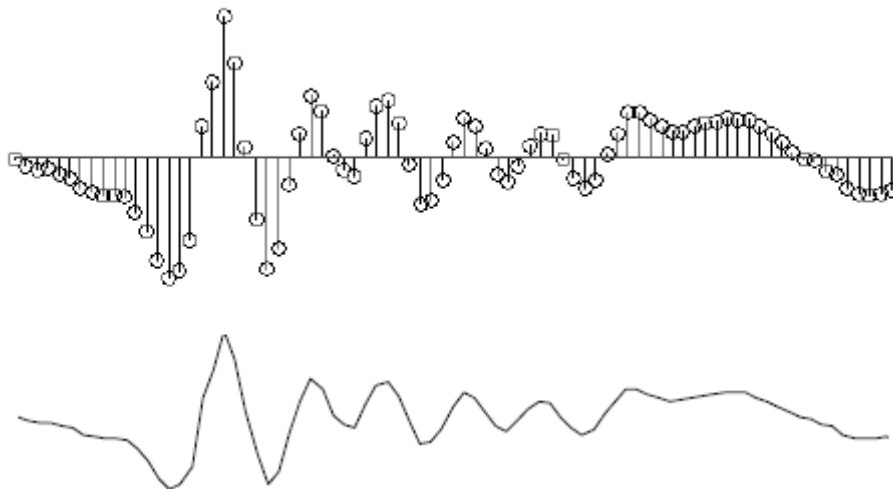
En general nos interesará calcular para nuestro estudio los valores en el tiempo de $e[n]$ y de $h[n]$ dado $x[n]$.

Señales Digitales

Si decimos que una señal continua es representada por $x_0(t)$, una señal digital la representaremos por $x[n]$

Donde la correspondencia entre la señal digital y la señal continua es la siguiente

$$x[n] = x_0(nT)$$



También definiremos frecuencia de muestreo como la inversa del periodo de muestreo de la siguiente manera:

$$F_s = 1/T$$

Una de las señales más importantes es el senoide

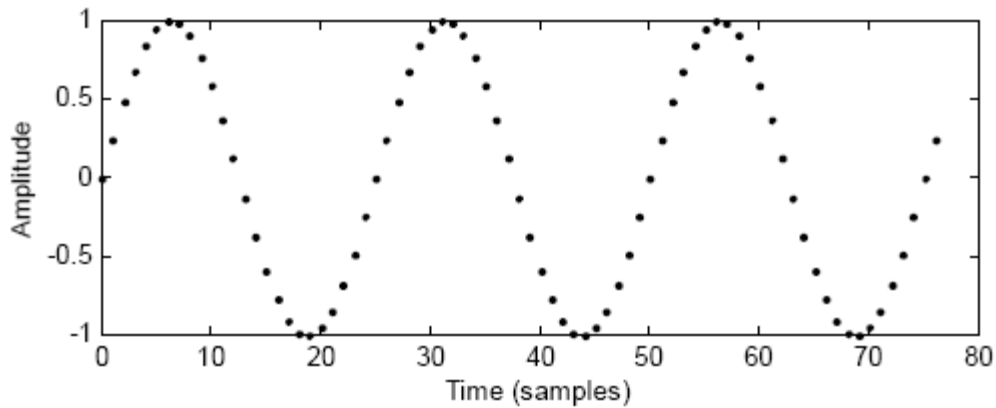
$$x_0[n] = A_0 \cos(\omega_0 n + \phi_0)$$

A_0 = amplitud

ω_0 = frecuencia angular

ϕ_0 = angulo de fase

Es necesario que la frecuencia de muestreo F_s sea por lo menos 2 veces la frecuencia más alta en la señal, a esto se le llama criterio Nyquist. (investigar el porque de este criterio), la frecuencia angular ω_0 está relacionada con la frecuencia lineal $0 \leq f_0 \leq 1$ (no confundir con la frecuencia de muestreo) $\omega_0 = 2\pi f_0$



El inconveniente surge cuando se quiere operar con identidades trigonométricas, para esto es mejor utilizar los números complejos

Un número complejo tiene la forma $z = x + jy$ donde x representa la parte real, y la parte imaginaria y j es la raíz cuadrada de -1 . En su forma polar esta representado como sigue

$$z = Ae^{j\phi}$$

Donde

$$e^{j\phi} = \cos \phi + j \sin \phi$$

Un senoide puede ser representado usando números complejos como sigue:

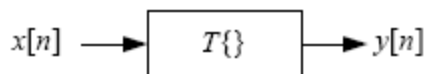
$$x_0[n] = A_0 \cos(\omega_0 n + \phi_0) = \text{Re}\{A_0 e^{j(\omega_0 n + \phi_0)}\}$$

Sistemas digitales lineales e invariantes en el tiempo

Un sistema será digital si dada una entrada $x[n]$, genera una salida $y[n]$

$$y[n] = T\{x[n]\}$$

esta relación puede ser vista como



es lineal si cumple

$$T\{a_1 x_1[n] + a_2 x_2[n]\} = a_1 T\{x_1[n]\} + a_2 T\{x_2[n]\}$$

y es invariante en el tiempo si cumple

$$y[n - n_0] = T\{x[n - n_0]\}$$

Operador convolución

Uno de los operadores más importantes es el operador convolución y describe a los sistemas lineales e invariantes en el tiempo

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = x[n] * h[n]$$





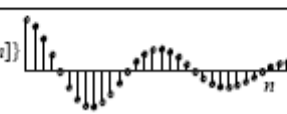
cuyas propiedades asociadas se muestran en esta tabla:

Commutative	$x[n] * h[n] = h[n] * x[n]$
Associative	$x[n] * (h_1[n] * h_2[n]) = (x[n] * h_1[n]) * h_2[n] = x[n] * h_1[n] * h_2[n]$
Distributive	$x[n] * (h_1[n] + h_2[n]) = x[n] * h_1[n] + x[n] * h_2[n]$

Ejemplo de convolución

$$x = [1, 3, 7, 3, 1] * h = [1, 3, 1] = y[1, 6, 17, 27, 17, 6, 1]$$

Algunas señales digitales de importancia son las siguientes:

<i>Kronecker delta, or unit impulse</i>	$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}$	
<i>Unit step</i>	$u[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$	
<i>Rectangular signal</i>	$\text{rect}_N[n] = \begin{cases} 1 & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$	
<i>Real exponential</i>	$x[n] = a^n u[n]$	
<i>Complex exponential</i>	$x[n] = a^n u[n] = r^n e^{j\omega_0 n} u[n] = r^n (\cos n\omega_0 + j \sin n\omega_0) u[n]$	

Pero que sucederá si de entrada $x[n]$ ponemos un senoide y lo convolucionamos con un filtro

Si $x[n] = e^{j\omega n}$ entonces:

$$y[n] = \sum_{k=-\infty}^{\infty} e^{j\omega(n-k)} h[k] = e^{j\omega n} \sum_{k=-\infty}^{\infty} e^{-j\omega k} h[k] = H(e^{j\omega}) e^{j\omega n}$$

Lo cual nos indica que si la entrada al sistema es un exponencial complejo, la salida es el mismo exponencial complejo escalado y ajustado en fase.

Trasformada de Fourier

El término $H(e^{j\omega})$ es la trasformada de Fourier de una señal discreta, siendo definido como:

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h[n] e^{-j\omega n}$$

con magnitud $|H(e^{j\omega})|$

y fase $e^{j \arg[H(e^{j\omega})]}$

Es decir si la entrada al sistema es la señal digital $h[n]$, ésta señal entrará en convolución con el senoide $e^{-j\omega n}$ y como $H(e^{j\omega})$ tiene como argumento ω que es la frecuencia angular, la señal $h[n]$ entrará en convolución con sinusoides de frecuencia determinada por el argumento de entrada ω .

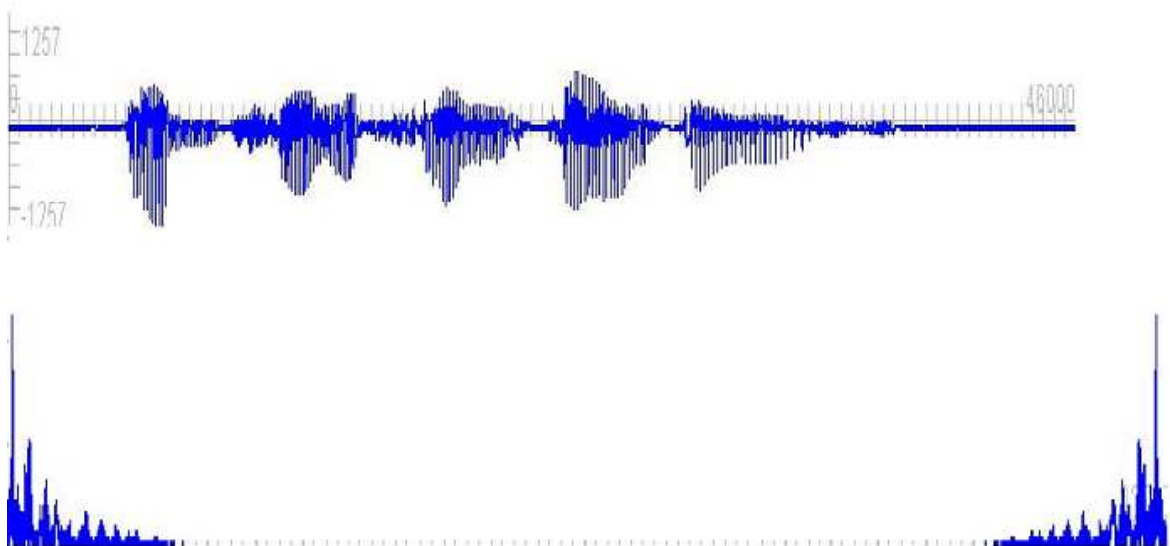
Lo que obtendremos será transformar nuestra señal de entrada del dominio del tiempo al dominio de la frecuencia, la transformada de Fourier es recuperable del dominio de la frecuencia al dominio del tiempo y esto se logra gracias a que la transformada de Fourier tiene inversa

$$h[n] = 1/(2\pi) \int_{-\pi}^{\pi} H(e^{j\omega}) e^{j\omega n} d\omega$$

Generalizando la transformada de Fourier es generalizada a la transformada z como sigue:

$$H(z) = \sum_{n=-\infty}^{\infty} h[n] z^{-n}$$

Donde z es cualquier variable compleja, la transformada de Fourier es la transformada z evaluada en $z = e^{j\omega}$



Propiedad Fundamental de la Convolución

Una de las propiedades fundamentales de la convolución es que una convolución en el dominio del tiempo es igual a una multiplicación en el dominio de la frecuencia y viceversa, esto es importante por varias razones, una de ellas es la complejidad computacional, es menos costoso realizar una multiplicación en el dominio de la frecuencia que realizar una convolución en el dominio del tiempo, a pesar de que primero tenemos que aplicar una transformada de Fourier a las señales para llevarla del dominio del

tiempo al dominio de la frecuencia, luego realizar las multiplicaciones respectivas y luego volverlas a llevar al dominio del tiempo, esto es posible gracias a algoritmos que realizan el proceso de calcular la transformada de Fourier de una señal en tiempo logaritmico
A continuación la demostración de esta propiedad

$$\begin{aligned} Y(z) &= \sum_{n=-\infty}^{\infty} y[n]z^{-n} = \sum_{n=-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} x[k]h[n-k] \right) z^{-n} \\ &= \sum_{k=-\infty}^{\infty} x[k] \left(\sum_{n=-\infty}^{\infty} h[n-k]z^{-n} \right) = \sum_{k=-\infty}^{\infty} x[k] \left(\sum_{n=-\infty}^{\infty} h[n]z^{-(n+k)} \right) \\ &= \sum_{k=-\infty}^{\infty} x[k]z^{-k} H(z) = X(z)H(z) \end{aligned}$$

Transformada Discreta de Fourier

Transforman una señal discreta en el tiempo en una señal discreta en la frecuencia, la Transformada Discreta de Fourier de una señal esta definida por:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-\frac{j2\pi nk}{N}} = \sum_{n=0}^{N-1} x[n]W_N^{kn}, \quad k = 0, 1, \dots, N-1$$

$$W_N = e^{-\frac{j2\pi}{N}}$$

El factor W es llamado también factor mariposa, el cual es una función de N términos de frecuencia, con argumento nk La Transformada Inversa Discreta de Fourier esta dada por

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]e^{\frac{j2\pi nk}{N}} = \sum_{k=0}^{N-1} X[k]W_N^{-kn} \quad n = 0, 1, \dots, N-1$$

Utilizar la Transformada Discreta de Fourier tiene un costo computacional elevado, cuando se trabaja con muchos datos, para ello un algoritmo propuesto por Cooley y Tukey, en 1965 hizo que el cálculo de la Transformada Discreta se realizara en forma mas eficiente, este es el algoritmo de la Transformada Rápida de Fourier, el aporte está en el hecho de que la Transformada Discreta de Fourier necesita N^2 operaciones mientras que la Transformada Rápida de Fourier solo necesita $N \log_2 N$ operaciones, donde N es el número de muestras, por ejemplo si $N = 1024$, la Transformada Discreta de Fourier necesitaría $1024^2 = 1,048,576$ operaciones entre multiplicaciones y sumas como mínimo, mientras que la Transformada Rápida de Fourier solo necesitaría $1024 \log_2 1024 = 10,240$ operaciones entre multiplicaciones y sumas como mínimo; la ventaja del segundo algoritmo es evidente. Existen muchos algoritmos que implementan la Transformada Rápida de Fourier y estos se basan en el paradigma divide y conquista [Cooley and Tukey,

1965], los cuales primero dividen el problema en dos o más subproblemas de pequeño tamaño, solucionan el mismo subproblema recursivamente por el mismo algoritmo, aplicando condiciones de límite para terminar la recursión cuando el tamaño de los subproblemas son suficientemente pequeños y obtienen la solución al problema original, combinando las soluciones de los subproblemas.

Existen muchos algoritmos Rápidos de Fourier y su aplicación depende de criterios, como el tamaño de datos a tratar, si el algoritmo va a ser implementado en computadoras con un procesador o en computadoras con varios procesadores, etc.; nosotros vamos a estudiar el algoritmo Radix – 2 FFT, el cual tiene dos variantes habitualmente usadas [Chu and George, 2000], los cuales son algoritmos secuenciales es decir se van a implementar en computadoras con un solo procesador, estos algoritmos son: algoritmo Radix–2 con Decimación en el Tiempo y algoritmo Radix – 2 con Decimación en la Frecuencia, solamente varían en el modo en como los dos subproblemas son definidos, no existiendo ninguna variación en complejidad computacional.

Algoritmo Radix-2 con Decimación en Frecuencia y reordenamiento en la salida de bits mezclados

Los primeros algoritmos rápidos fueron propuestos en forma independiente por Cooley y Tukey en 1965 en su forma de decimación en el tiempo y en su forma de decimación en frecuencia por Gentleman y Sande en 1966 A continuación se describe el algoritmo Radix-2 con decimación en frecuencia y reordenamiento en la salida de bits.

Tenemos la Transformada Discreta de Fourier definida de la siguiente manera:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi nk}{N}} = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad k = 0, 1, \dots, N-1$$

donde

$$W_N = e^{\frac{-j2\pi}{N}}$$

a partir de acá se puede deducir lo siguiente:

$$W_N^{\frac{N}{2}} = -1$$

$$W_{\frac{N}{2}} = W_N^2$$

$$W_N^N = 1$$

donde N es necesariamente un valor que sea el resultado de una potencia de dos, para que el algoritmo pueda aplicar eficientemente el principio de divide y conquista.

El algoritmo radix-2 con decimación en la frecuencia, es obtenido por definir dos subproblemas de la forma:

$$\{X(2k)|k = 0, \dots, \frac{N}{2} - 1\}$$

que son ls frecuencias diezmadadas de los índices pares y

$$\{X(2k + 1)|k = 0, \dots, \frac{N}{2} - 1\}$$

las frecuencias diezmadadas de los índices impares

usarendo como notación X_k por $X(K)$ y X_n por $x(n)$

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} x_n W_N^{kn} + \sum_{n=\frac{N}{2}}^{N-1} x_n W_N^{kn}$$

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} x_n W_N^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x_{n+\frac{N}{2}} W_N^{k(n+\frac{N}{2})}$$

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} (x_n + x_{n+\frac{N}{2}} W_N^{k\frac{N}{2}}) W_N^{kn}, \quad k = 0, 1, \dots, N - 1$$

escogiendo los indices pares tenemos:

$$X_{2k} = \sum_{n=0}^{\frac{N}{2}-1} (x_n + x_{n+\frac{N}{2}} W_N^{kN}) W_N^{2kn}$$

$$X_{2k} = \sum_{n=0}^{\frac{N}{2}-1} (x_n + x_{n+\frac{N}{2}}) W_N^{kn}, \quad k = 0, 1, \dots, \frac{N}{2} - 1$$

definiendo $Y_k = X_{2k}$ y $y_n = x_n + x_{n+\frac{N}{2}}$, $k = 0, 1, \dots, \frac{N}{2} - 1$, tenemos la primera

mitad del problema

$$Y_k = \sum_{n=0}^{\frac{N}{2}-1} y_n W_{\frac{N}{2}}^{kn}, \quad k = 0, 1, \dots, \frac{N}{2} - 1$$

similarmente para los índices impares tenemos:

$$X_{2k+1} = \sum_{n=0}^{\frac{N}{2}-1} (x_n + x_{n+\frac{N}{2}} W_N^{(2k+1)\frac{N}{2}}) W_N^{(2k+1)n}$$

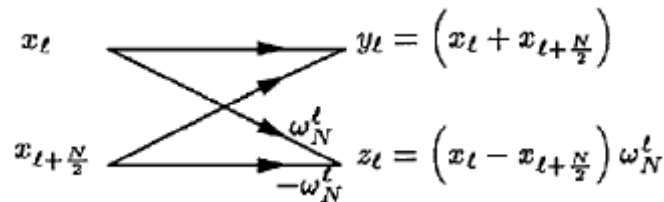
$$X_{2k+1} = \sum_{n=0}^{\frac{N}{2}-1} ((x_n - x_{n+\frac{N}{2}}) W_N^n) W_{\frac{N}{2}}^{kn}, \quad k = 0, 1, \dots, \frac{N}{2} - 1$$

definiendo $Z_k = X_{2k+1}$ y $z_n = (x_n - x_{n+\frac{N}{2}}) W_N^n$, $k = 0, 1, \dots, \frac{N}{2} - 1$, obtenemos la segunda mitad del problema

$$Z_k = \sum_{n=0}^{\frac{N}{2}-1} z_n W_{\frac{N}{2}}^{kn}, \quad k = 0, 1, \dots, \frac{N}{2} - 1$$

El cálculo de y_n y de z_n en el paso de la subdivisión es definida en la literatura como: la mariposa Gentleman-Sande.

La salida de este procedimiento obtiene los valores de frecuencia pero en un orden mezclado, es decir si se tiene los valores de entrada x almacenados en un array a , la salida cuando el algoritmo Radix-2 con Decimación en Frecuencia sea computado, obtendremos los valores X en el array a pero de una manera mezclada.



La mariposa Gentleman-Sande.

El reordenamiento de estos valores se hace teniendo en cuenta lo siguiente: las posiciones ordenadas de los valores de salida serán obtenidos al hacer un operación de bits reverso a las posiciones del vector de salida. Si la notación binaria de un número A es la siguiente:

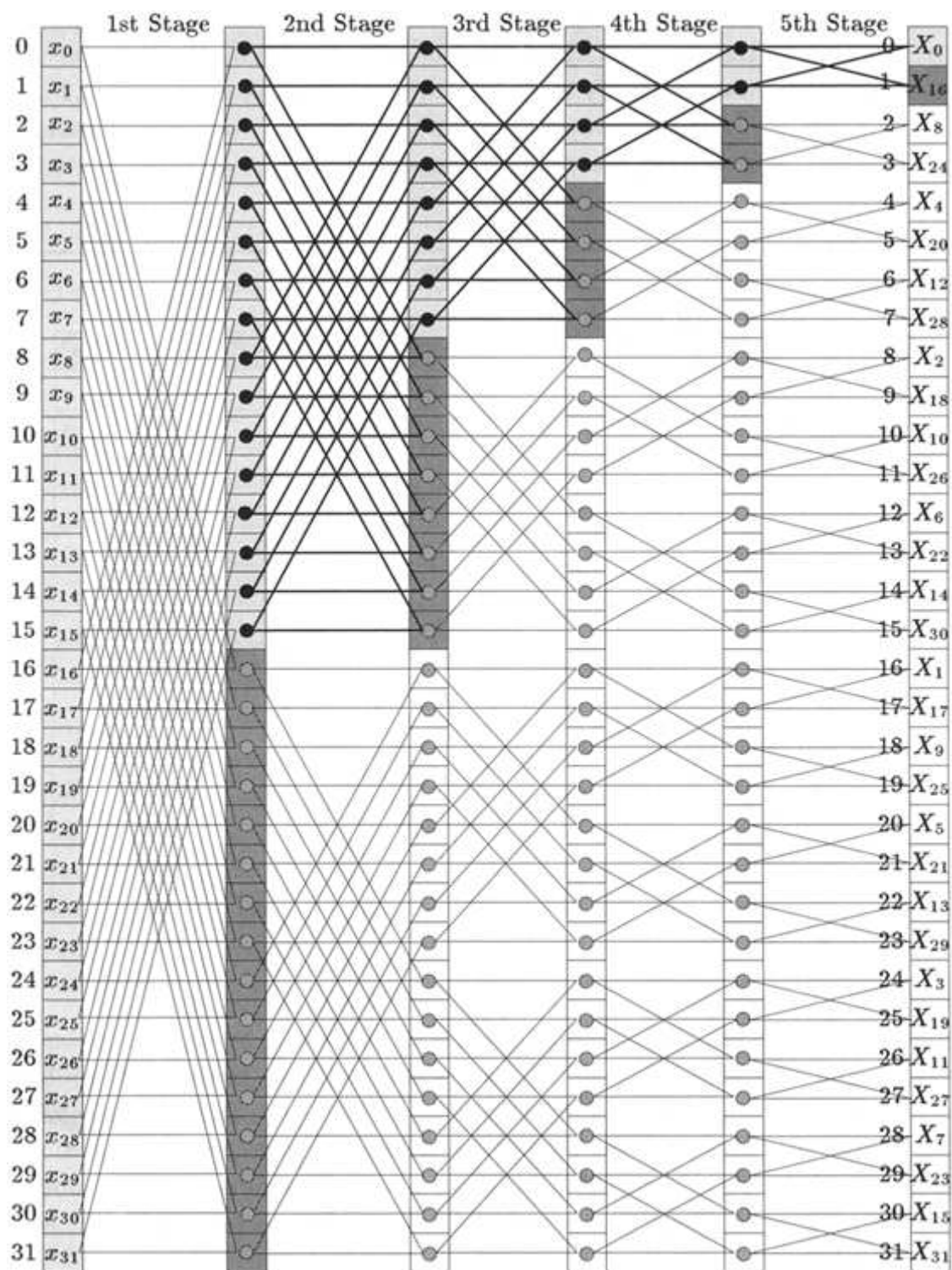
$$A = abcd...yz$$

donde $abcde...yz$ son valores que corresponden a los números 0 o 1, su bits reverso sera:

$$A = zywx....a$$

que corresponden a intercambiar la última posición de su representación binaria por la primera, la penúltima por la segunda, la antepenúltima por la tercera y así sucesivamente.

Entrada :	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7
	$a[0]$	$a[1]$	$a[2]$	$a[3]$	$a[4]$	$a[5]$	$a[6]$	$a[7]$
Salida :	X_0	X_4	X_2	X_6	X_1	X_5	X_3	X_7
	$a[0]$	$a[1]$	$a[2]$	$a[3]$	$a[4]$	$a[5]$	$a[6]$	$a[7]$



DIF FFT				
0	ω^0	0		0
1	ω^1	1		1
2	ω^2	2		2
3	ω^3	3		3
4	ω^4	4		4
5	ω^5	5		5
6	ω^6	6		6
7	ω^7	7		7
8	ω^8	8		8
9	ω^9	9		9
10	ω^{10}	10		10
11	ω^{11}	11		11
12	ω^{12}	12		12
13	ω^{13}	13		13
14	ω^{14}	14		14
15	ω^{15}	15		15
16	ω^0	16	ω^0	16
17	ω^2	17	ω^2	17
18	ω^4	18	ω^4	18
19	ω^6	19	ω^6	19
20	ω^8	20	ω^8	20
21	ω^{10}	21	ω^{10}	21
22	ω^{12}	22	ω^{12}	22
23	ω^{14}	23	ω^{14}	23
24	ω^0	24	ω^0	24
25	ω^4	25	ω^4	25
26	ω^8	26	ω^8	26
27	ω^{12}	27	ω^{12}	27
28	ω^0	28	ω^0	28
29	ω^8	29	ω^8	29
$N - 2 = 30$	ω^0	30		30

vector largo (arriba) y pequeño (abajo), para almacenar los factores mariposa

DIF FFT				
0	ω^0	0	ω^0	0
1	ω^1	1		1
2	ω^2	2	ω^2	2
3	ω^3	3		3
4	ω^4	4	ω^4	4
5	ω^5	5		5
6	ω^6	6	ω^6	6
7	ω^7	7		7
8	ω^8	8	ω^8	8
9	ω^9	9		9
10	ω^{10}	10	ω^{10}	10
11	ω^{11}	11		11
12	ω^{12}	12	ω^{12}	12
13	ω^{13}	13		13
14	ω^{14}	14	ω^{14}	14
$\frac{N}{2} - 1 = 15$	ω^{15}	15		15

Trasformada discreta del Coseno

Es una trasformada similar a la trasformada discreta de Fourier, pero en lugar de usar cosenos y senos, solo utiliza cosenos, es decir trabaja solamente con números reales, existen 8 variantes de la trasformada discreta del coseno, siendo 4 de ellas las más comunes

Definición :

La trasformada discreta del coseno es una función lineal e invertible de $\mathbb{R}^N \rightarrow \mathbb{R}^N$, donde los N números reales $x_0 \dots x_{N-1}$ son convertidos a los N números reales $x_0 \dots x_{N-1}$ mediante las siguientes fórmulas

DTC I

$$X_k = \frac{1}{2}(x_0 + (-1)^k x_{N-1}) + \sum_{n=1}^{N-2} x_n \cos \left[\frac{\pi}{N-1} nk \right] \quad k = 0, \dots, N-1.$$

se multiplica por el factor de escalamiento $\sqrt{2/(N-1)}$

DTC II

$$X_k = \sum_{n=0}^{N-1} x_n \cos \left[\frac{\pi}{N} \left(n + \frac{1}{2} \right) k \right] \quad k = 0, \dots, N-1.$$

se multiplica por el factor de escalamiento $\sqrt{2/N}$

DTC III

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n \cos \left[\frac{\pi}{N} n \left(k + \frac{1}{2} \right) \right] \quad k = 0, \dots, N-1.$$

se multiplica por el factor de escalamiento $\sqrt{2/N}$

DCTIV

$$X_k = \sum_{n=0}^{N-1} x_n \cos \left[\frac{\pi}{N} \left(n + \frac{1}{2} \right) \left(k + \frac{1}{2} \right) \right] \quad k = 0, \dots, N-1.$$

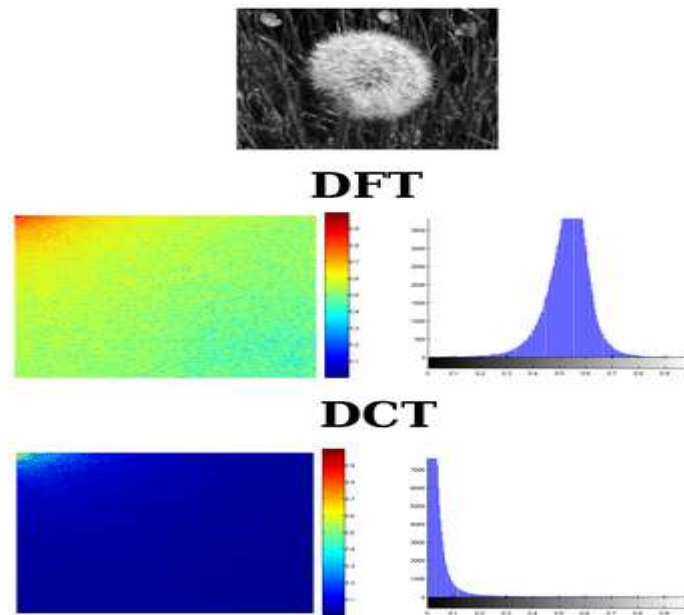
se multiplica por el factor de escalamiento $\sqrt{2/N}$

MDTC Transformada Discreta del Coseno Modificada

$$X_k = \sum_{n=0}^{2N-1} x_n \cos \left[\frac{\pi}{N} \left(n + \frac{1}{2} + \frac{N}{2} \right) \left(k + \frac{1}{2} \right) \right]$$

IMDTC

$$y_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cos \left[\frac{\pi}{N} \left(n + \frac{1}{2} + \frac{N}{2} \right) \left(k + \frac{1}{2} \right) \right]$$



Referencias bibliograficas

- [1] Huang, X., Acero, A., and Hon, H. 2001 *Spoken Language Processing: a Guide to Theory, Algorithm, and System Development*. 1st. Prentice Hall PTR.
- [2] E. Chu and A. George, *Inside the FFT Black Box." Serial and Parallel Fast Fourier Transform Algorithms*. Boca Raton, FL: CRC Press, 2000
- [3] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform", IEEE Trans. Computers, 90-93, Jan 1974.
- [4] John P. Princen and Alan B. Bradley, "Analysis/synthesis filter bank design based on time domain aliasing cancellation," IEEE Trans. Acoust. Speech Sig. Proc. ASSP-34 (5), 1153-1161 (1986).
- [5] Princen J, Johnson A, Bradley, A, en Subband/Transform Coding Using Filter Bank Designs Based on Time Domain Aliasing Cancellation, Proc. of the ICASSP 1987, pp 2161-2164.