``ANALISIS Y DISEÑO DE ALGORITMOS PARA LA COMPUTACIÓN CON ESTRUCTURAS CIRCULANTES

Tesis Presentada por:
Abraham H. Díaz-Pérez

Tesis sometida en cumplimiento parcial
de los requisitos para el grado de

MAESTRO EN CIENCIAS
en
INGENIERIA ELECTRICA
(Procesamiento Digital de Señales)

UNIVERSIDAD DE PUERTO RICO
RECINTO UNIVERSITARIO DE MAYAGÜEZ
2004

Aprobada por:

_____	_____
Manuel Jiménez Cedeño, Ph.D. Miembro, Comité Graduado	Fecha
_____	_____
Ramón Vásquez Espinosa., Ph.D. Miembro, Comité Graduado	Fecha
_____	_____
Domingo Rodríguez, Ph.D. Presidente, Comité Graduado	Fecha
_____	_____
Luis F. Cáceres, Ph.D. Representante de Estudios Graduados	Fecha
_____	_____
Jorge Ortíz Álvarez, Ph.D. Director del Departamento	Fecha

Contenido

Lista de Tablas

Lista de Figuras

Abstract

This dissertation proposal deals with the study of algorithms for computation with circulants structures. We study the different current algorithms for computation with circulant structures; specifically, for sequence convolutions and polynomial multiplications. Particularly, this work focuses on the arithmetic complexity of the matrix-vector product computations when they represent cyclic convolution operations in order to obtain efficient algorithms with low complexity in the multiplication sense.

Actually, two threads are used for the computation of the cyclic convolution: the *direct approach* which examines the structure of the system of equations describing the convolution operation, and the *transform approach* which maps the convolution operation into an alternative domain using the discrete Fourier transform (DFT) as tool.

We present an eclectic approach, using the intrinsic symmetry of the circulant matrices, and the roots of units of the monic polynomial $z^N - 1$, for the formulation of new algorithms for the cyclic convolution and for the products of polynomials of order $N = 2^s$, with s belonging to the positive whole numbers ($s \in Z^+$), which reach low multiplicative complexity, according to Winograd's theorem.

Resumen

Esta propuesta de disertación trata del estudio de algoritmos para el cómputo con estructuras circulantes. Estudiamos los diversos algoritmos que existen en la actualidad para el cómputo con las estructuras circulantes, específicamente para la convolución cíclica y las multiplicaciones de polinomios. En este trabajo nos enfocamos particularmente en la complejidad aritmética del producto de matrices circulantes por un vector, representando la operación de la convolución cíclica, para obtener algoritmos eficientes con baja complejidad desde el punto de vista de la multiplicación para la convolución cíclica.

Actualmente dos tipos de enfoques son usados para el cálculo de la convolución cíclica: el *enfoque directo*, el cual examina la estructura del sistema de ecuaciones que describen la convolución cíclica, y el *enfoque por transformación*, el cual lleva el proceso de convolución a un dominio alternativo usando la transformada de Fourier discreta (DFT) como herramienta.

En este trabajo usamos un enfoque ecléctico, aprovechando la simetría intrínseca de las matrices circulantes y las raíces de la unidad del polinomio mónico $z^N - 1$, para la formulación de nuevos algoritmos para la realización de la convolución cíclica de orden $N = 2^s$, con s perteneciendo a los enteros positivos ($s \in Z^+$), los cuales alcanzan baja complejidad multiplicativa de acuerdo al teorema de Winograd.

Dedicatoria

A Evariste Galois por hacernos detener en nuestras investigaciones debido a las suyas y por la sempiterna cita:

"El genio es condenado, por una organización social maliciosa, a una eterna negativa de justicia, en favor de la aduladora mediocridad"

Agradecimientos

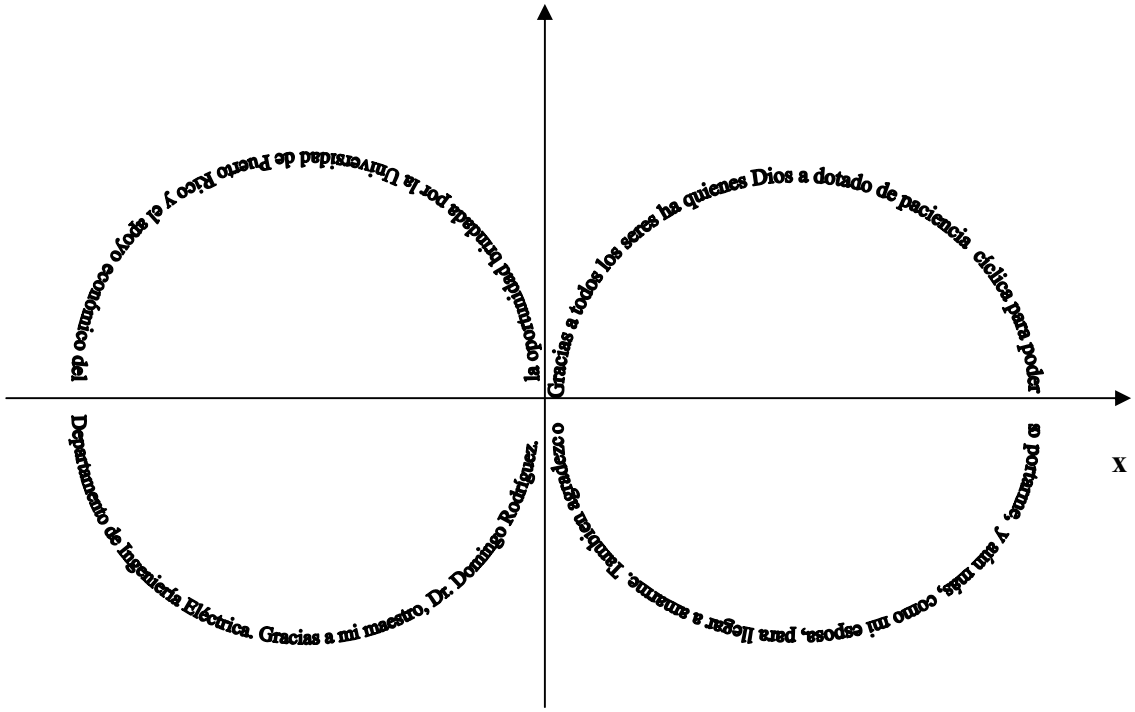


Figura 1. ***Lemniscata de Bernoulli*** $(x^2+y^2)^2=a^2(x^2-y^2)$

Introducción

Esta propuesta de disertación trata sobre el estudio de estructuras matemáticas denominadas circulantes y su posible realización, implementación o ejecución en estructuras digitales de computación en hardware electrónico.

El estudio persigue establecer relaciones correspondientes entre expresiones de matrices circulantes, definidas como *unidades funcionales matemáticas* y estructuras básicas en hardware, definidas como *unidades básicas de computación*.

Esto quiere decir, que algunas de las funciones básicas matemáticas sobre circulantes, serán decompuestas en términos más sencillos a los que llamaremos *unidades funcionales primitivas*, y se buscará entonces, que cada una de las expresiones de éste conjunto de estructuras algebraicas, tenga una aplicación correspondiente en las unidades básicas de computación.

Ampliando el contexto, se puede formular que para un operador de señales matemático dado, el cual actúa sobre una secuencia finita de datos escalares, digamos $x[.]$, para producir otra secuencia, llamémosla $X[.]$, un amplio número de diferentes estructuras, tanto algorítmicas como computacionales pueden existir. Para un operador de una aplicación (función) en particular, se busca obtener la implementación óptima referente a tiempo y recursos de sistema, en términos de costo de la arquitectura computacional. Esto no es una tarea fácil, ya que existe una gran variedad de diseños para un operador específico. Éste estudio buscará identificar mecanismos que puedan indicar el grado de eficiencia, referente a tiempo y utilización de recursos de las estructuras computacionales que se crearán para este trabajo. Otra parte fundamental de nuestra labor consistirá en expresar los operadores de funciones sobre las estructuras de computación de las circulantes, utilizando los tres operadores fundamentales en las que una estructura de hardware digital puede diseñarse y construirse, a saber: *elemento sumador, elemento multiplicador y elemento de almacenamiento y retraso.*

Nos referimos a estructuras algebraicas cuando tenemos uno o más conjuntos con una o varias operaciones binarias, que poseen unas determinadas propiedades y unos determinados elementos notables. Las estructuras algebraicas se representan agrupando bajo un paréntesis el conjunto y las operaciones que actúan sobre el, para el caso particular que nos concierne, una forma de representar las estructuras

algebraicas sobre circulantes puede ser $(\psi, +, *, Z^{-1})$, donde ψ, es el conjunto predefinido de las circulantes, + es la operación básica de adición, * es la operación fundamental de multiplicación y Z^{-1} es la operación elemental de almacenamiento y retraso. En el desarrollo del proyecto observaremos que estos elementos también son útiles para representar a los *filtros de respuesta finita* (FIR por sus siglas en inglés), por lo que las aplicaciones de circulantes pueden ser implementadas en *hardware* en esta clase de dispositivos.

Describimos las estructuras computacionales como el conjunto de todos los elementos de procesamiento que son utilizados para realizar un determinado algoritmo o aplicación mediante la definición de secuencias de control apropiadas.

Decimos que los elementos de procesamiento son *modulares* debido a que la metodología de síntesis explota la regularidad de la gráfica de flujo de señal del algoritmo. Esto quiere decir, que se buscará que las estructuras que agrupan elementos del problema, se encuentren altamente interrelacionadas, para que de ésta manera, puedan ser modularmente efectivas. En términos de medida, diremos que un sistema es modularmente más efectivo, si la suma de las relaciones funcionales entre diferentes módulos es mínima.

Se dice que un sistema es *escalable*, si conserva su efectividad cuando ocurre un incremento significativo en el número de recursos y el número de usuarios. En otras palabras, una arquitectura (de hardware o software) es escalable, si podemos aumentar sus recursos para soportar una mayor demanda de rendimiento y funcionalidad y/o disminuir sus recursos para reducir costos.

Nuestro trabajo está originalmente motivado en la necesidad de derivar un algoritmo específico para ser aplicado en un marco teórico para el trabajo con matrices circulantes.

La importancia teórica de los procesos cíclicos se reafirma, una vez se reconoce que todos los métodos de series en tiempo, que hacen uso de la transformada discreta de Fourier, están efectivamente basados en la asunción que la data proviene de un proceso circulante.

La disponibilidad de arquitecturas de multiprocesadores, hace deseable que se desarrollen algoritmos que puedan romper largos bloques de convoluciones cíclicas en bloques más pequeños de convoluciones cíclicas, dentro de una estructura en paralelo apropiada [Rodriguez2]. No solo la complejidad computacional es minimizada, si no que aún más importante, la estructura del algoritmo se intenta que sea lo más cercana a la aplicación de la arquitectura. Como resultado del proceso, un algoritmo altamente regular para la convolución cíclica, apropiado para arreglos de

VLSI (Very Large Scale Integration) e implementación en paralelo, es obtenido [Rodriguez3].

En su naturaleza fundamental, éste trabajo trata del estudio de la complejidad aritmética de la multiplicación de matrices circulantes por un vector. Explotaremos la simetría intrínseca de las matrices circulantes para la formulación de un nuevo algoritmo para la realización de la convolución cíclica de orden $N = 2^s$, $(\text{con } s \in Z^+)$ optimo en el sentido de alcanzar la mínima complejidad en la multiplicación.

La computación del producto matriz-vector, para una matriz densa, normalmente toma N^2 multiplicaciones; sin embargo, explotando la estructura especial de una matriz circulante, la operación puede ser substancialmente reducida para matrices largas. Una de las claves es usar la transformada rápida de Fourier (FFT), la cual hace posible computar el producto matriz-vector en solo $(N)log_2(N)$ multiplicaciones complejas. En este trabajo, realizaremos un enfoque ecléctico que nos permita la unificación de los conceptos de división en el tiempo con el de las raíces de la unidad de la transformada de Fourier discreta (del polinomio $z^N - 1$), para obtener un algoritmo eficiente, que solo necesita N multiplicaciones complejas para llevar a cabo este producto matriz-vector.

Trasfondo Histórico y Motivación

Las matrices circulantes hicieron su aparición en la literatura en el año de 1846, en un documento escrito por E. Catalán [Catalán] y desde ese entonces han sido objeto de una curiosidad matemática que aún no termina. La literatura sobre matrices circulantes, desde su introducción hasta 1920 fue resumida en cuatro documentos escritos por Muir [Muir1]-[Muir4]. Un tratado un poco más reciente, el cual contiene una bibliografía muy útil y excelente contexto matemático, es el proveído por Davis (1979) [Davis]; sin embargo, éste último libro no trata con problemas de análisis de series de tiempo ni su aplicación a estructuras computacionales.

Este proyecto utiliza la bien conocida técnica algorítmica llamada *"dividir para reinar"*

Definición 1:

Una técnica algorítmica para solucionar un problema en un caso de tamaño N, puede tener una solución que se encuentra por medio de dos formas; directamente porque el caso es fácil (típicamente, porque el caso N es pequeño) o cuando se divide en dos o más casos pequeños. Cada uno de estos casos más pequeños se soluciona recurrentemente, y las soluciones se combinan para producir una solución completa para el caso original.

Nota: El nombre de *"divide y conquista",* se da porque el problema es conquistado dividiéndolo en varios problemas más pequeños. Esta técnica presenta algoritmos elegantes, simples y a menudo absolutamente eficientes. Los ejemplos bien conocidos incluyen, multiplicación rápida de la matriz de Strassen, la transformada rápida de Fourier, Operaciones con Producto Kronecker, etc.

A continuación se presenta una lista del término en diversos idiomas:
Francés: diviser pour régner
Latín: divide et impera
Alemán: teile und herrsche
Inglés: Divide and conquer

Definición 2:

Una técnica algorítmica se dice que es recurrente, si una función, para lograr una tarea, se llama a si misma, para realizar una cierta parte de la labor.

Nota: Las soluciones recurrentes implican dos partes importantes:

- *Caso Simple:* En el cual el problema es bastante simple y puede ser solucionado directamente.

- *Caso Recurrente:* Un caso recurrente tiene tres componentes:

  1. Dividir el problema en unas o más partes más simples o más pequeñas.

  2. Llamar la función (recurrentemente) en cada parte.

  3. Combinar las soluciones de las piezas separadas, para entregar la solución completa del problema

El trabajo presente ésta motivado en la necesidad de tener un marco teórico para las aplicaciones que tengan que ver con matrices circulantes de sistemas lineales invariantes en el tiempo, y se presentarán soluciones que pueden ser representadas en estructuras computacionales que se obtienen de funciones derivadas para algoritmos en paralelo.

Otra motivación importante en cuanto a las aplicaciones del marco teórico propuesto en la presente disertación, consiste en que la solución de sistemas lineales de ecuaciones, relacionados con matrices circulantes puede dar resultado a un procedimiento eficiente de iteración para el problema de la convolución cíclica, de

éstas matrices circulantes con un vector. Para realizar la operación de convolución con un vector, la transformada rápida de Fourier puede ser utilizada, debido a que las circulantes pueden ser diagonalizadas por la DFT [Davis]. Por esto, la multiplicación por una matriz circulante, puede ser llevada a cabo con la ayuda de la transformada rápida de Fourier en dos dimensiones y la multiplicación por una matriz diagonal, lo cual reduce los costos de operaciones (sumas y multiplicaciones) a $O(n \log n)$.

Identificación del Problema y Contribución

Las matrices circulantes, son un tópico fascinante, cuyas aplicaciones son muy variadas, y cubren los campos de filtrado digital, procesamiento de imágenes, incluyen también los bien conocidos tópicos de algoritmos para la transformada rápida de Fourier, convolución cíclica, cálculo numérico de coeficientes y recientemente, están siendo usadas para estudiar los sistemas de ecuaciones diferenciales con control.

Este trabajo contiene resultados conocidos y estudiados por otros autores, y aportes propios, tales como la aplicación de los más recientes conceptos de álgebra lineal numérica para la creación de un marco teórico y un algoritmo novedoso para el computo del producto de una matriz circulante con un vector y su posible aplicación a estructuras de computación en hardware, por medio del uso de *primitivas computacionales*, las cuales brindan el soporte necesario para resolver importantes problemas de ciencias aplicadas.

Un aporte fundamental del presente trabajo, consiste en la reducción de las multiplicaciones necesarias para la realización del producto matriz circulante por un vector, así como también las formulaciones matemáticas que las describen. Otro factor de relevante importancia del presente trabajo radica en la flexibilidad de la posible aplicación en hardware, ya que ésta puede realizarse explotando el paralelismo intrínseco de la formulación algorítmica, conociendo que la ventaja de la computación en paralelo, reside en la rapidez de la ejecución del algoritmo, que se hace explicita en algunas tareas de procesamiento de señales o en aplicaciones de gran escala, como por ejemplo el trabajo con imágenes hiperespectrales.

Organización de la Tesis

En el presente documento, vamos a establecer un marco de trabajo teórico para la implementación de operaciones con matrices circulantes y su posible aplicación a estructuras de computación en hardware. Esto se realizará para modelos de computación en paralelo, recursivos en el tiempo.

El presente trabajo esta organizado como sigue: primero iniciaremos nuestro trabajo mostrando los fundamentos del problema en el estudio de los algoritmos para la convolución digital; segundo, realizaremos un paralelo entre los productos de los polinomios y el problema de la convolución; tercero, mostraremos como se desarrolló el algoritmo propuesto para el problema de la multiplicación de una matriz circulante por un vector (convolución cíclica), usando el marco conceptual desarrollado en las secciones previas y se mostrará una *gráfica de flujo de señal* que puede ser aplicada en arquitecturas en paralelo, por medio de VLSI; cuarto, se hará una introducción al problema de la convolución cíclica en dos dimensiones. Esta es una operación de sistemas multidimensionales, para las cuales se buscará explotar el hecho de que la representación por *"columna mayor"* de ésta operación, ofrece una regularización que puede asociarse sin mayores inconvenientes a las características de la multiplicación de una matriz circulante por bloques con bloques circulantes *(CBBC)* y un vector, para la cual un estudio sistemático tal como el empleado para la convolución cíclica en una dimensión puede ser aplicado con algunas variaciones, y obtener a su vez, la mínima complejidad multiplicativa que se alcanzó para el problema en una dimensión. Quinto, las conclusiones y contribuciones del trabajo son presentadas; por último se presentarán como apéndice las pruebas de algunos teoremas y funciones que se indicaron en el curso de nuestra presentación, y una copia del algoritmo propuesto realizado en Matlab® .

Capítulo I.

Marco de Trabajo Teórico

En éste capítulo, revisaremos la teoría existente sobre matrices circulantes para aprovechar éste conocimiento en la construcción de las herramientas necesarias para un diseño sistemático para arquitecturas sobre circulantes que sean recursivas en el tiempo.

En la sección 1, introduciremos algunas terminologías básicas sobre circulantes y su descomposición, que serán utilizadas en el desarrollo de la presente tesis.

En la sección 1.2, procederemos a explorar el marco teórico referente a las matrices circulantes, y las propiedades que este tipo de matrices poseen, tomando como base algunas de las definiciones presentadas en la sección 1.1, enfocando nuestra atención al problema de las matrices circulantes multidimensionales, en sistemas lineales invariantes en el tiempo.

1 Definiciones

1.1 Arquitecturas Computacionales

El término arquitectura computacional se refiere a un *diseño computacional*, el cual se puede representar tanto en hardware como en software, o una combinación de hardware y software. La arquitectura de un sistema define siempre sus esquemas generales, y puede definir también mecanismos exactos. Una arquitectura abierta permite que el sistema sea conectado fácilmente con los dispositivos y los programas hechos por otros fabricantes. Las arquitecturas abiertas utilizan componentes disponibles y se conforman con los estándares aprobados. Un sistema con una arquitectura cerrada, por otra parte, es uno en el que el diseño es exclusivo del propietario, haciéndolo difícil de conectar con otros sistemas.

1.2 Sistemas Digitales

Un sistema digital, es aquel que describe cualquier sistema basado en eventos o datos discontinuos. Las computadoras son máquinas digitales porque en su nivel más básico, pueden distinguir entre apenas dos valores, 0 y 1, o apagado y encendido. No hay manera simple de representar todos los valores, como por ejemplo 0.25. Todos los datos que una computadora procesa se deben codificar digitalmente, como serie de ceros y unos. El contrario de digital es análogo. Un dispositivo análogo

típico es un reloj en el cual las manecillas se mueven continuamente alrededor de la cara. Tal reloj es capaz de indicar cada hora posible. En contraste, un reloj digital es capaz de representar solamente un número finito de los instantes (cada décimas de un segundo, por ejemplo). En general, los seres humanos experimentan el mundo analógicamente. La visión, por ejemplo, es una experiencia análoga debido a que percibimos degradaciones infinitamente suaves de brillo y de colores. La mayoría de los acontecimientos análogos, sin embargo, se pueden simular digitalmente. Aunque las representaciones digitales son aproximaciones de acontecimientos análogos, son útiles porque son relativamente fáciles de almacenar y de manipular electrónicamente. El truco consiste en convertir de análogo a digital, y luego regresar otra vez a la forma análoga.

1.3 Modelado

Se denomina generalmente modelado al proceso de representar un objeto o un fenómeno del mundo real como sistema de ecuaciones matemáticas. Más específicamente, el término se utiliza a menudo para describir el proceso de representar objetos de 2 ó 3 dimensiones en una computadora. Todos los usos tridimensionales, incluyendo CAD/CAM y software de animación, realizan modelado.

1.4 Simulación

Se denomina simulación al proceso de imitar un fenómeno real mediante un sistema de fórmulas matemáticas, en otras palabras, la simulación es el modelo en acción. Los programas de computadora avanzados pueden simular las condiciones atmosféricas, las reacciones químicas, reacciones atómicas e incluso procesos biológicos. En teoría, cualquier fenómeno que se pueda reducir a datos y a ecuaciones matemáticas se puede simular en una computadora. En la práctica, sin embargo, la simulación es extremadamente difícil porque la mayoría de los fenómenos naturales están sujetos a un sinnúmero de influencias. Uno de los trucos para desarrollar simulaciones útiles, es por lo tanto, determinar cuáles son los factores de influencia más importantes. Además de los procesos de imitación de un fenómeno (para considerar cómo se comporta bajo diversas condiciones), las simulaciones también se utilizan para probar nuevas teorías. Después de crear una teoría de relaciones causales, el teórico puede codificar las relaciones en la forma de un programa de computadora. Si el programa entonces se comporta de la misma manera que el proceso verdadero, hay una buena opción de que las relaciones propuestas sean correctas.

1.5 Formulación del Nivel del Sistema del Marco de Trabajo Teórico

En esta formulación del nivel de sistema de nuestro marco de trabajo teórico deseamos describir las características de los sistemas lineales en una dimensión, que serán la base de las aplicaciones que se presentarán en los capítulos subsiguientes.

Hagamos que $\mu_R^{<Z>}$, sea el conjunto de todas las secuencias digitales de longitud *R*. Esto es, si $g \in \mu_R^{<Z>}$, entonces:

$$g = \{g[0], g[1], g[2], \ldots\ldots, g[R-1]\} \qquad \text{I-1}$$

Un sistema T_h es un sistema digital de una dimensión, si éste toma como entrada a una señal $x \in \mu_R^{<Z>}$, y produce a la salida otra señal, $y \in \mu_R^{<Z>}$. En forma de diagrama de bloque, esta operación se puede representar como:

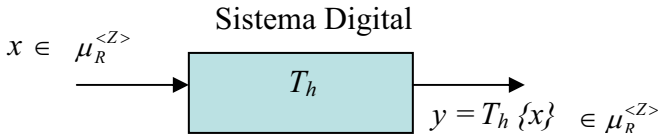


Figura 2. ***Diagrama de Bloque de Sistema Digital de una Dimensión***

Por ejemplo; si tenemos el sistema T_h con entrada $x \in \mu_2^{<Z>}$, y salida, $y \in \mu_3^{<Z>}$

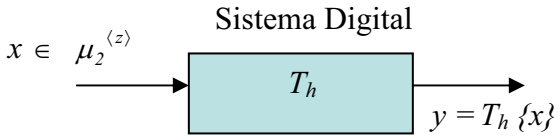


Figura 3. ***Ejemplo Particular de Sistema Digital de una Dimensión.***

Estamos interesados en describir las características de una clase especializada de sistema digital conocida como *filtro digital.*

Un *filtro digital* es cualquier sistema digital que satisface dos condiciones; la condición de *linealidad* y que además es *invariante al desplazamiento* (*shift invariance* en Inglés). Con el objetivo de establecer las condiciones de *linealidad* y de *invarianza al desplazamiento* como las características de los filtros digitales, comenzaremos asumiendo que todas nuestras señales tienen una longitud *N* o que cualquier señal general $g \in \mu_N^{<Z>}$.

Para establecer la condición de *linealidad,* el sistema tiene que satisfacer las siguientes dos condiciones; *superposición* y *homogeneidad*. Para que el principio de superposición se cumpla, los siguientes diagramas deben ser equivalentes:

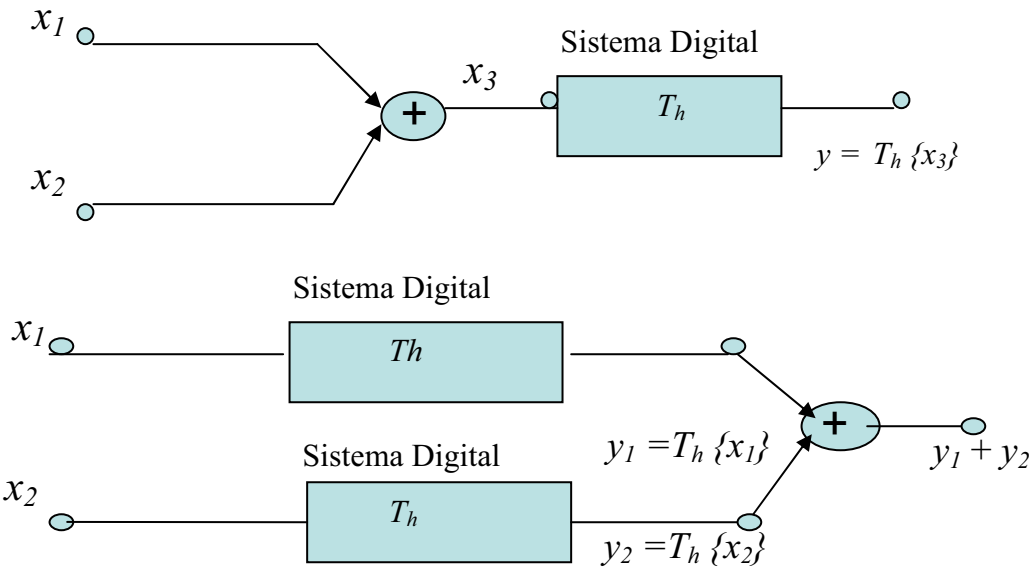


Figura 4. ***Diagramas de Bloques Para el Principio de Superposición.***

Ahora, para la condición de *homogeneidad,* los siguientes dos diagramas deben ser equivalentes:

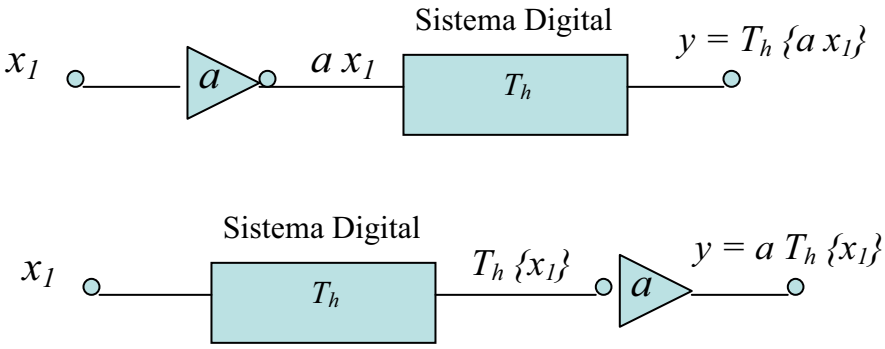


Figura 5. ***Diagramas de Bloques Para la Condición de Homogeneidad.***

Para establecer la condición de *invariante al desplazamiento,* definimos que un sistema es invariante al desplazamiento, si un desplazamiento en la secuencia de entrada, produce un desplazamiento proporcional en la secuencia de salida. En diagramas de bloques, esto se puede representar como:

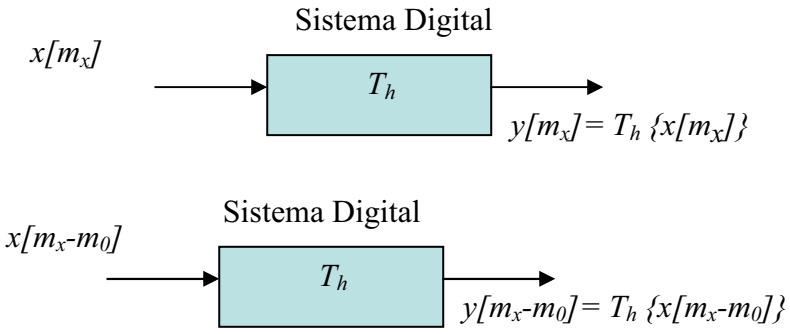


Figura 6. ***Diagramas de Bloques Para la Condición de Invariante al Desplazamiento.***

Introduciremos el *formato de diagrama de bloque de registro* para un sistema digital en la siguiente figura:

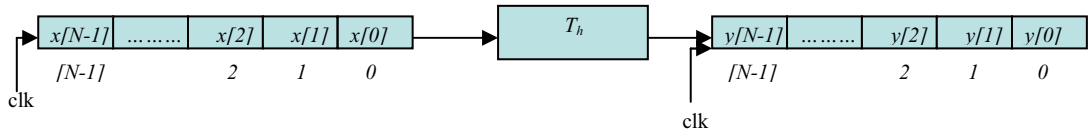


Figura 7. ***Formato de Diagrama de Bloque de Registro para un Sistema Digital***

Como podemos observar, los vectores de entrada y salida tienen la misma longitud, ésta es una característica de diseño. El hecho real es que escogeremos la longitud del vector de salida para que sea al menos *2N*, con *N* siendo la longitud del vector de entrada.

Iniciaremos nuestro estudio definiendo la *función de unidad delta* (*delta unit function* en Inglés) $\delta[n],\ n \in Z_N$. En forma de diagrama de bloque de registro tenemos:

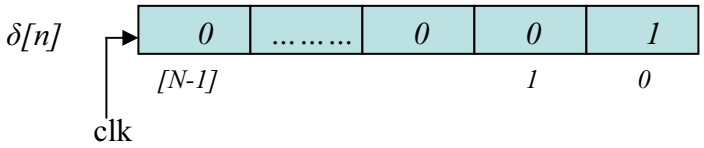


Figura 8. ***Diagrama de Bloque de Registro Función Unidad Delta***

Si deseáramos realizar la multiplicación de $a_0\delta[n],\ n \in Z_N$ y a_0 una constante arbitraria, procederemos entonces como sigue, en forma de diagrama de registro:

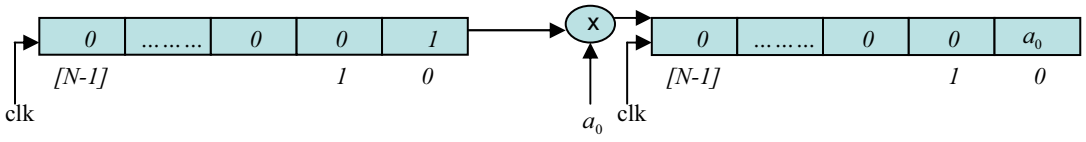
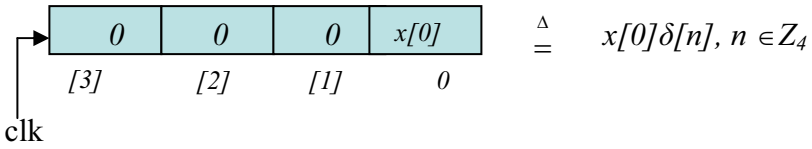


Figura 9. ***Diagrama de Bloque de Registro de la Multiplicación de una Constante por la Función Unidad Delta***

Podemos notar que cualquier secuencia digital $x \in V_N$ puede ser expresada como una combinación lineal de secuencias de funciones de unidad delta desplazadas, Esto es:

$$x[n] = \sum_{k=0}^{N-1} x[k]\delta[n-k],\ n \in Z_N \qquad \text{I-2}$$

Para hacer claridad en éstos conceptos, presentamos el siguiente ejemplo específico: Sea *N=4* entonces, $x[0]\delta[n]$ es representado en forma de diagrama de registro como sigue:



De la misma forma, podemos representar a $x[1]\delta[n-1]$ como:

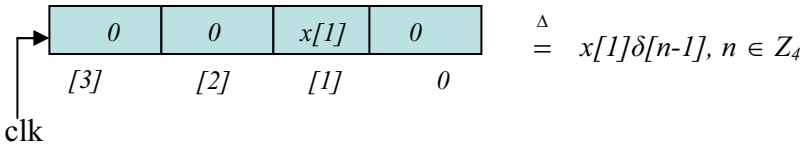


Figura 10.*Diagrama de Bloque de Registro para x[0]δ[n], y x[1]δ[n-1], n ∈ Z₄*

Dado que $x \in V_4$, es la secuencia digital $x[n] = x[0], x[1], x[2], x[3]$, podemos escribir x en forma de bloque de diagrama de registro como:

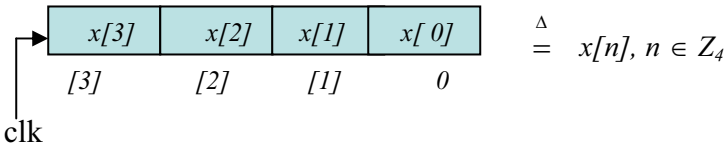


Figura 11. ***Diagrama de Bloque de Registro para x[n], n ∈Z₄***

Mediante el uso del diagrama de bloque de registro, podemos expresar $x \in V_4$ como una combinación lineal de factores multiplicados por secuencias de unidad delta desplazadas, de la siguiente manera:

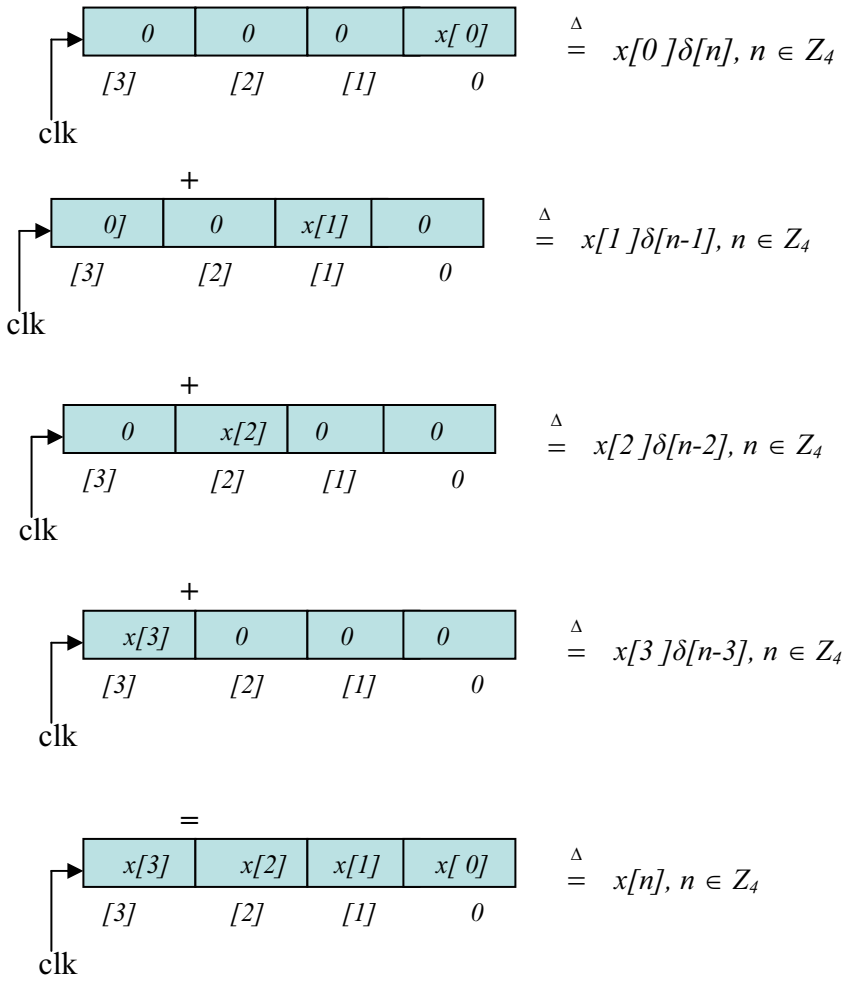


Figura 12. ***Representación de x[n], n ∈ Z4, como una combinación lineal de sus coeficientes multiplicados por secuencias de unidad delta desplazadas.***

La respuesta de un sistema digital a una *secuencia de unidad delta* caracteriza completamente el sistema. Esto es, para un operador de sistema T_h, la respuesta del sistema para una señal de entrada $\delta[n]$ esto es $h[n] = T_h\{\delta[n]\}$ es el sistema completamente caracterizado. En forma de diagrama de registro de bloques, tenemos:

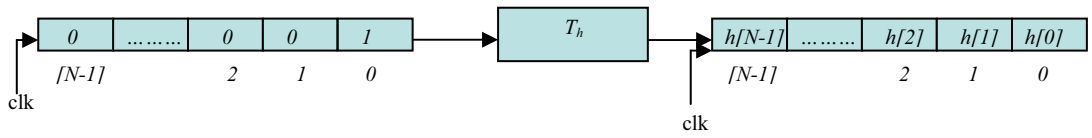


Figura 13.*Caracterización de un Sistema Mediante su Respuesta para una Señal de Entrada* $\delta[n]$**.**

Como pudimos observar anteriormente, un sistema es *invariante al desplazamiento*, si un desplazamiento en la secuencia de entrada, produce un desplazamiento proporcional en la secuencia de salida. Lo mismo ocurre cuando se desplaza la función delta, la respuesta del sistema, es decir, su caracterización, también se verá desplazada. Usando un diagrama de registro de bloques para un ejemplo específico, digamos $N=8$, obtendremos lo siguiente:

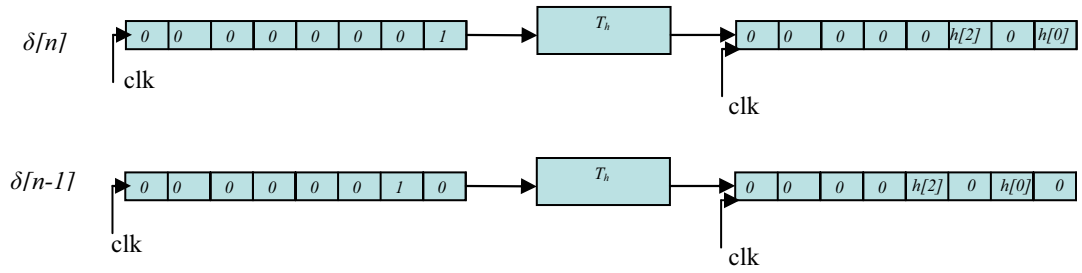


Figura 14. ***Desplazamiento de la Función de Caracterización de un Sistema Mediante el Desplazamiento de*** $\delta[n]$ ***a*** $\delta[n-1]$**.**

1.6 Convolución Lineal en una Dimensión

Una *señal discreta* es simplemente una secuencia de números. Una *señal discreta multidimensional* es un arreglo de números de *n-dimensiones*. Si el conjunto de números de los elementos de dicha secuencia o arreglo son finitos, entonces la señal es llamada *digital* [Myers]. Considere una señal $x[n]$ aplicada a algún procesador digital de señales con respuesta al impulso $h[n]$. Esto puede relacionarse como la suma de un conjunto infinito de señales unitarias delta, cada una corrida de forma tal que la j^{esima} de dichas señales, no es cero solamente en la posición j, y cada una está escalada por la magnitud de la muestra de $x[n]$ en la posición j. La secuencia de salida del sistema $y[n]$ se define como la convolución de la entrada y el sistema, y es nuevamente un conjunto infinito de respuestas delta unitaria, cada una escalada y corrida en la misma forma, esto es, a una posición en particular de la secuencia j, $y[j]$ consiste en la suma de los componentes:

$$y[j] = x[j]h[0] + x[j-1]h[1] + x[j-2]h[2] + \ldots$$

En una forma más compacta, la salida del procesador a cualquier posición de la secuencia n, está dada por:

$$y[n] = x[n] * h[n]$$

símbolo $*$ denota el operador de convolución, que se define como:

$$y[n] = \sum_{m=0}^{\infty} h[m] \cdot x[n-m] = \sum_{m=0}^{\infty} h[n-m] \cdot x[m] \; ; \; n = (-\infty, \infty) \qquad \text{I-4}$$

Esta describe la *convolución digital*

Hay otra propiedad importante que un sistema real necesita poseer, ésta es la *estabilidad.* Un sistema es estable si una entrada limitada produce una salida limitada. Un sistema lineal, invariante en el tiempo, causal, con respuesta al impulso $h[n]$ es estable, si y solo si:

$$\sum_{m=0}^{\infty} |h[m]| < \infty \qquad \text{I-5}$$

En forma de diagrama de bloques, la convolución lineal puede representarse como:

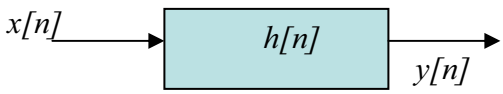


Figura 15. ***Diagrama de Bloques para la Convolución Lineal de la Entrada y el Sistema***

Cinco casos particulares de convolución digital son de gran importancia, estos son:

1.6.1 *x[n] Finita y h[n] Finita:*

Si $x[n]$ es una secuencia con solo N términos y $h[n]$ es una secuencia con solo M términos, entonces, su convolución digital está dada por:

$$y[n] = x[n] * h[n]$$

$$y[n] = \sum_{m=0}^{n} h[m] \cdot x[n-m] = \sum_{m=0}^{n} h[n-m] \cdot x[m] \; ; \qquad \text{I-6}$$

$$n = 0,1,2...N+M-2$$

Con $x[n]$ y $h[n]$ siendo ceros por fuera de sus rangos definidos. Este tipo de convolución se denomina *lineal* o *aperiódica.*

1.6.2 $x[n]$ *Infinita y* $h[n]$ *Finita:*

Si $x[n]$ es una señal infinita de términos que no son todos ceros, pero $h[n]$ es una secuencia con solo N términos, entonces, su convolución digital está dada por:

$$y[n] = x[n] * h[n]$$

$$y[n] = \sum_{m=0}^{N-1} h[m] \cdot x[n-m] = \sum_{m=0}^{N-1} h[n-m] \cdot x[m]\ ; \qquad \text{I-7}$$

$$n = (-\infty, \infty)$$

1.6.3 $x[n]$ *Finita y* $h[n]$ *Infinita:*

Si $x[n]$ es una señal finita de solo N términos, pero $h[n]$ es una secuencia infinita de términos que no son todos ceros, entonces, su convolución digital está dada por:

$$y[n] = x[n] * h[n]$$

$$y[n] = \sum_{m=0}^{N-1} h[m] \cdot x[n-m] = \sum_{m=0}^{N-1} h[n-m] \cdot x[m] \qquad \text{I-8}$$

$$n = (0, -\infty)$$

1.6.4 $x[n]$ *Infinita y* $h[n]$ *Infinita:*

Asumiendo un sistema causal, tenemos:

$$y[n] = x[n] * h[n]$$

$$y[n] = \sum_{m=0}^{\infty} h[m] \cdot x[n-m] = \sum_{m=0}^{\infty} h[n-m] \cdot x[m] \qquad \text{I-9}$$

$$n = (-\infty, \infty)$$

1.6.5 $x[n]$ *Infinita y Periódica,* $h[n]$ *Infinita y Periódica:*

Este caso es un caso particular de 1.6.4; consideremos dos señales infinitas $h[n]$ y $x[n]$ ambas distinguidas por ser periódicas con un periodo de N muestras. *La convolución Periódica* o *cíclica* de esas señales, denotada por $y[n] = x[n] \otimes_N h[n]$, también con periodo fundamental N es computada como:

$$y[n] = x[n] \otimes_N h[n] \qquad \text{I-10}$$

$$y[n] = \sum_{m=0}^{N-1} h[m] \cdot x[\langle n-m \rangle_N] = \sum_{m=0}^{N-1} h[m] \cdot x[\langle n-m \rangle_N]$$
$$n = 0,1,2...N-1$$

La notación del operador $\langle P \rangle_N$, denota el modulo de P, y es el residuo de P, luego de ser dividida por *N,* o $\langle P \rangle_N = residuo(P/N)$.

La convolución cíclica es una herramienta muy notable en el estudio de la convolución digital. Podemos obtener una relación entre la convolución cíclica y la convolución aperiódica, debido a que las estructuras matemáticas de la convolución cíclica y la aperiódica son similares. Por eso, es posible expresar la convolución aperiódica como una convolución cíclica de la siguiente manera:

Hagamos: $Q = N + M - 1$

Ahora definamos dos secuencias de Q puntos cada una $h'[n]$ y $x'[n]$ derivadas al llenar con ceros $h[n]$ y $x[n]$ de la siguiente manera:

$$h'[n] = \begin{cases} h[n] \text{ para } n = 0,1,2,...,M-1 \\ 0 \text{ para } n = M, M+1,...,Q-1 \end{cases}$$

$$x'[n] = \begin{cases} x[n] \text{ para } n = 0,1,2,...,N-1 \\ 0 \text{ para } n = N, N+1,...,Q-1 \end{cases} \qquad \text{I-11}$$

Cada periodo de la nueva convolución cíclica es:

$$y'[n] = x'[n] \otimes_Q h'[n]$$
$$y'[n] = \sum_{m=0}^{Q-1} h'[m] \cdot x'[\langle n-m \rangle_Q] = \sum_{m=0}^{Q-1} h'[m] \cdot x'[\langle n-m \rangle_Q] \quad n = 0,1,2...Q-1$$

Lo cual es igual a la convolución aperiódica deseada.

Vamos a presentar un ejemplo específico para demostrar el concepto de la convolución:

$$\text{Sea } x[n] = \begin{cases} 4-n, & 0 \le n \le 4 \\ 0 & de\,otra\,forma \end{cases}, \text{ y } h[n] = \begin{cases} 2, & n = 1 \\ 1, & 2 \le n \le 4 \\ 0, & de\,otra\,forma \end{cases} \qquad \text{I-12}$$

Entonces, $x[n] * h[n]$ es representada gráficamente como sigue:

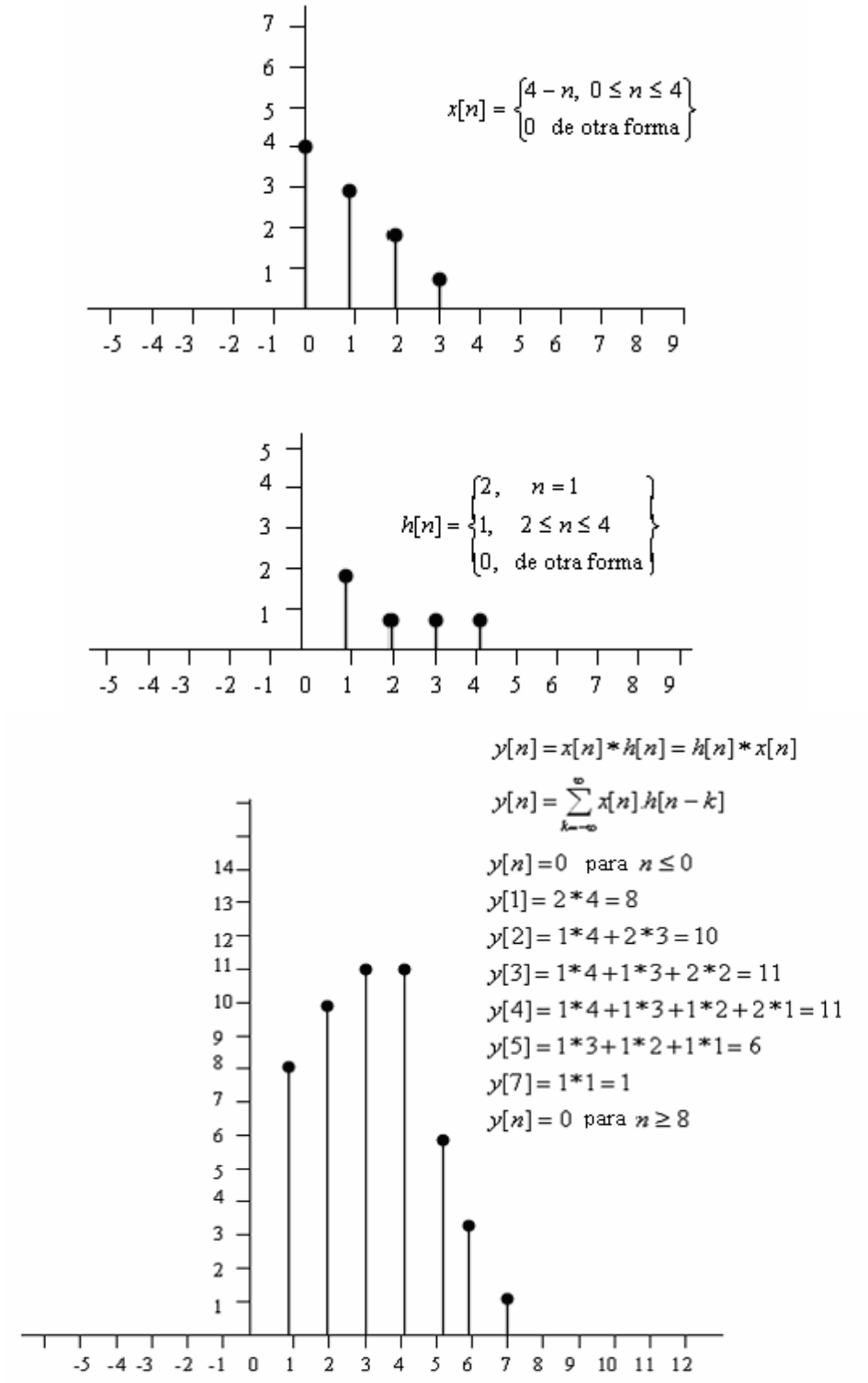


Figura 16. ***Grafica para la Convolución Lineal de la Entrada y el Sistema para un Ejemplo Específico***

Introduciremos ahora la forma matricial para la realización de la convolución cíclica, esto se realizará por medio del ejemplo específico que se mostró para explicar el concepto de la convolución lineal;

$$\text{Sea } x[n] = \begin{Bmatrix} 4-n, & 0 \le n \le 4 \\ 0 & de\,otra\ forma \end{Bmatrix}, \text{ y } h[n] = \begin{Bmatrix} 2, & n = 1 \\ 1, & 2 \le n \le 4 \\ 0, & de\,otra\ forma \end{Bmatrix} \qquad \text{I-13}$$

En forma vectorial, nuestras señales pueden verse como:

$$x[n] = [\underset{\uparrow}{4},3,2,1] \text{ y } h[n] = [\underset{\uparrow}{0},2,1,1,1]; \qquad \text{I-14}$$

Para realizar el proceso de la convolución cíclica en forma de multiplicación matriz vector, la matriz ciclica resultante debe tener la misma longitud que el vector de entrada, lo que se hace es ajustar con ceros la longitud de las señales, para que cada una tenga una longitud de *(N+M-1)*, en nuestro caso *N=4, M=5,* entonces, la longitud de cada secuencia será de *(4+5-1)=8.* Nuestra función de transferencia está representada por *h[n],* por lo tanto, esta secuencia será la escogida para realizar la matriz circulante, también pudo realizarse escogiendo a *x[n],* puesto que la operación de convolución es conmutativa, pero para tener en cuenta los sistemas físicos, asumiremos la respuesta del sistema como *h[n].*

Nuestros nuevos vectores son entonces:

$$x[n] = [\underset{\uparrow}{4},3,2,1,0.0,0,0] \text{ y } h[n] = [\underset{\uparrow}{0},2,1,1,1,0,0,0], \qquad \text{I-15}$$

y la convolución entre ellos;

$$y[n] = h[n] * x[n] \qquad \text{I-16}$$

Esto puede representarse matricialmente como sigue:

$$y[n] = h[n] * x[n] = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 2 \\ 2 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 2 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 2 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 0 \end{bmatrix} \times \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \text{I-17}$$

La representación dada por I-17, es nuestro primer acercamiento a las matrices circulantes, y desde ahora podemos apreciar su importancia para definir la salida de sistemas cuya función de transferencia es conocida.

La operación de multiplicación de la matriz por el vector nos resulta en:

$$y[n] = h[n] * x[n] = \begin{bmatrix} 0 \\ 8 \\ 10 \\ 11 \\ 11 \\ 6 \\ 1 \\ 0 \end{bmatrix} \qquad \text{I-18}$$

Que como podemos apreciar, es el mismo resultado que el obtenido en la figura 16.

1.7 Algoritmos Para la Convolución Cíclica

El objetivo del presente trabajo es presentar un algoritmo eficiente para determinar la convolución cíclica, óptimo en el sentido de alcanzar el mínimo número de multiplicaciones necesarias para dicha operación. En la actualidad dos tipos de enfoques para este problema son utilizados, el enfoque directo, el cual examina las ecuaciones para la convolución cíclica y el enfoque por transformación, el cual realiza una aplicación del proceso de convolución a un dominio matemático más simple usando la transformada rápida de Fourier como herramienta.

Consideremos la ecuación para convolución cíclica de orden N

$$y[n]=\sum_{m=0}^{N-1}h[m]\cdot x[\langle n-m\rangle_N]=\sum_{m=0}^{N-1}x[m]\cdot h[\langle n-m\rangle_N] \qquad \text{I-19}$$
$$n=0,1,2...N-1$$

Esta describe completamente un ciclo de la función periódica $y[n]$. En forma matricial, ésta ecuación puede representarse como:

$$\begin{bmatrix} y[0] \\ y[1] \\ . \\ . \\ . \\ y[N-1] \end{bmatrix} = \begin{bmatrix} h[0] & h[N-1] & . & . & . & h[1] \\ h[1] & h[0] & . & . & . & h[2] \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ h[N-1] & h[N-2] & . & . & . & h[0] \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ . \\ . \\ . \\ x[N-1] \end{bmatrix} \qquad \text{I-20}$$

O consecuentemente:

$$\begin{bmatrix} y[0] \\ y[1] \\ . \\ . \\ . \\ y[N-1] \end{bmatrix} = \begin{bmatrix} x[0] & x[N-1] & . & . & . & x[1] \\ x[1] & x[0] & . & . & . & x[2] \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ x[N-1] & x[N-2] & . & . & . & x[0] \end{bmatrix} \begin{bmatrix} h[0] \\ h[1] \\ . \\ . \\ . \\ h[N-1] \end{bmatrix} \qquad \text{I-21}$$

Definición I-1:

Una matriz A es *Toeplitz*, si los elementos de cada una de las diagonales son los mismos, esto es, si:

$$a_{ij}=a_{p,q} \quad para \;\; i-p=j-q \qquad \text{I-22}$$

Si la matriz A es $N\times N$ y cada fila es igual a la precedente circularmente rotada, entonces la matriz A es llamada *circulante* [Davis]. La matriz en la convolución cíclica es *circulante* y *Toeplitz*.

1.8 Matrices Circulantes

Una matriz circulante de orden *N*, o simplemente circulante es una matriz cuadrada *C*, en el conjunto $\ell^2(Z_N)$,[1] para la cual, los elementos de cada fila son idénticos a los de la fila previa, pero están movidos una posición a la derecha y luego enrollados alrededor de si misma [Davis], por ejemplo:

$$C= circ\ (c_0, c_1, c_2, \ldots\ldots\ldots c_{N-1}) \qquad \text{I-23}$$

[1] $\ell^2(Z_N)$, *es un espacio de Hilbert*

$$C=\begin{pmatrix} c_0 & c_1 & c_2 & . & c_{N-1} \\ c_{N-1} & c_0 & c_1 & . & c_{N-2} \\ . & . & . & . & . \\ . & . & . & & . \\ c_1 & c_2 & c_3 & . & c_0 \end{pmatrix} \qquad \text{I-24}$$

En la anterior matriz se puede observar que el efecto circulante también se realiza en forma de columnas, es decir, se puede tener una representación circulante de un vector columna, tal como se aprecia en la siguiente figura:

$$C= circ\ (c_{0,}\, c_{1,}\, c_{2,}\, \ldots\ldots\ldots\ldots\ c_{N-1})^T \qquad \text{I-25}$$

$$C=\begin{pmatrix} c_0 & c_{N-1} & c_{N-2} & . & c_{N-3} \\ c_1 & c_0 & c_{N-1} & . & c_{N-2} \\ . & . & . & . & . \\ . & . & . & & . \\ c_{N-1} & c_{N-2} & c_{N-3} & . & c_0 \end{pmatrix} \qquad \text{I-26}$$

Se infiere de las anteriores representaciones que $c_{(i,j)} = c<_{(i+1,j+1)}>_{N,}$ donde la operación $<_{(i+1,j+1)}>_{N,}$ implica la operación modulo, definida como $<i>_{N} =$ residuo de (i/N).

Las matrices circulantes cumplen con la propiedad de linealidad, por lo tanto, en ellas se pueden verificar las propiedades de superposición y homogeneidad, por esta razón podemos decir:

1.8.1 Superposición

Sean A y B dos vectores circulantes, entonces:

$$circ\ (a_{0,}\, a_{1,}\, a_{2,}\, \ldots\ a_{N-1}) + circ\ (b_{0,}\, b_{1,}\, b_{2,}\ldots..\ b_{N-1}) = circ\ (c_{0,}\, c_{1,}\, c_{2,}\, ..\ c_{N-1}), \qquad \text{I-27}$$

$$c_{0=}\ (a_0 + b_0),\ c_{1=}\ (a_1 + b_1),\ c_{2=}\ (a_2 + b_2)\ \ldots\ldots,\ c_{N-1\,=}\ (a_{N-1} + b_{N-1}) \qquad \text{I-28}$$

1.8.2 Homogeneidad

Sea α un escalar, entonces:

$$\alpha \cdot circ\ (a_{0,}\, a_{1,}\, a_{2,}\, \ldots\ldots\ldots\ a_{N-1}) = \quad circ\ (\alpha\, a_{0,}\, \alpha\, a_{1,}\, \alpha\, a_{2,}\, \ldots\ldots\ldots\ \alpha\, a_{N-1}) \qquad \text{I-29}$$

1.8.3 Matrices de Uso Común en la Teoría de Matrices Circulantes

Dos matrices de uso común en la teoría de matrices circulantes son:

$$\prod = \begin{bmatrix} 0 & 1 & . & . & . & 0 \\ 0 & 0 & . & . & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & 0 & 1 \\ 1 & 0 & 0 & . & 0 & 0 \end{bmatrix} \quad \text{y} \quad \Gamma = \begin{bmatrix} 1 & 0 & . & . & . & 0 & 0 \\ 0 & 0 & . & . & . & 0 & 1 \\ 0 & 0 & . & . & . & 1 & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 1 & . & . & . & 0 & 0 \end{bmatrix} \qquad \text{I-30}$$

Ambas de orden N. Es un hecho claro que estas matrices satisfacen:

$$\prod^N = I_N; \qquad y \qquad \Gamma^2 = I;$$
$$\prod^T = \prod^* = \prod^{-1} = \prod^{N-1} \qquad \Gamma^* = \Gamma^T = \Gamma = \Gamma^{-1} \qquad \text{I-31}$$

Donde los súper-índices T y $*$ denotan la transpuesta y la transpuesta conjugada de la matriz respectivamente.

Proposición I-1.

Sea A una matriz de tamaño NxN, y $\prod$ la matriz de orden N, anteriormente definida. Entonces A es circulante si y solo si $\prod A = A \prod$

Prueba: Sea $A=(a_{ij})$, entonces $\prod A \prod^* = \prod A \prod^{-1} = a_{(i+1)(j+1)}$ (donde los subíndices son tomados modulo N) y ésta matriz es igual a $A=(a_{ij})$, solo si A es circulante.

Es claro de ésta proposición que A es circulante siempre y cuando A^* lo sea.

1.8.4 Propiedades de las Matrices Circulantes

1. Como Polinomios en la misma matriz conmutan respecto al producto, entonces, el producto de matrices circulantes conmuta. Más aún, el producto de circulantes, es de nuevo una circulante.
2. Como C y C^* conmutan, toda matriz circulante es normal.
3. De 1. y 2. se obtiene que si C es circulante y $k \in Z^+$, entonces C^k también es circulante.

Una matriz circulante *C* de orden $N = R\text{x}S$ es automáticamente circulante por bloques, y cada bloque es una matriz de tipo *Toeplitz*. Los bloques son de orden *S*, en un arreglo de $R\text{x}S$ bloques.

Ejemplo: Consideremos una circulante de orden 6; la podemos dividir en 3x3 bloques de orden 2 cada uno, convirtiéndola en una matriz circulante por bloques:

$$CB = \begin{bmatrix} \begin{bmatrix} a & b \\ f & a \end{bmatrix} \begin{bmatrix} c & d \\ b & c \end{bmatrix} \begin{bmatrix} e & f \\ d & e \end{bmatrix} \\ \begin{bmatrix} e & f \\ d & e \end{bmatrix} \begin{bmatrix} a & b \\ f & a \end{bmatrix} \begin{bmatrix} c & d \\ b & c \end{bmatrix} \\ \begin{bmatrix} c & d \\ b & c \end{bmatrix} \begin{bmatrix} e & f \\ d & e \end{bmatrix} \begin{bmatrix} a & b \\ f & a \end{bmatrix} \end{bmatrix} \quad \text{ó} \quad CB = \begin{bmatrix} A1 & A2 & A3 \\ A3 & A1 & A2 \\ A2 & A3 & A1 \end{bmatrix} \qquad \text{I-32}$$

Nótese que un bloque circulante, no es necesariamente una matriz circulante. Denotaremos el conjunto de matrices circulantes por bloques de tipo *(m,n),* por $CB_{(m,n)}$, donde *CB*, denota circulante por bloques, *n,* es el orden de los bloques y *m* es el número de bloques de la matriz. En el ejemplo dado en [I.32], *m=3, n=2.*

Claramente observamos que los bloques no necesariamente son circulantes, para que esto sea cierto, por ejemplo, en el primer bloque *b* debería ser igual a *f.*

Teorema I-1

$A \in CB_{(m,n)}$ si y sólo si

$$A(\Pi_m \otimes I_n) = (\Pi_m \otimes I_n)A$$

Prueba: $(\Pi_m \otimes I_n) \in CB_{(m,n)}$ y está dada por:

$$(\Pi_m \otimes I_n) = \begin{bmatrix} 0_n & I_n & 0_n & . & . & . & 0_n \\ 0_n & 0_n & I_n & . & . & . & 0_n \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0_n & 0_n & 0_n & . & . & . & I_n \\ I_n & 0_n & 0_n & . & . & . & 0_n \end{bmatrix}_{mn \times mn} \qquad \text{I-33}$$

Por otro lado $A \in CB_{(m,n)}$ si y sólo si

$$A = \begin{bmatrix} A_1 & A_2 & . & . & . & A_m \\ A_m & A_1 & . & . & . & A_{m-1} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ A_2 & A_3 & . & . & . & A_1 \end{bmatrix} \qquad \text{I-34}$$

Usaremos ahora la multiplicación formal (por bloques) de matrices para obtener la igualdad [Lara]:

$$(I_m \otimes A_1) = diag(A_1, A_1 ..., A_1)$$

$$(\Pi_m \otimes A_2) = circ(0, A_2 ..., 0)$$

$$(\Pi_m^2 \otimes A_3) = circ(0,0, A_3, 0..., 0) ....$$

Con lo cual podemos escribir $circ(A_1, A_2, A_3 ..., A_m)$ como:

$$circ(A_1, A_2, A_3 ..., A_m) = \sum_{j=0}^{m-1} \Pi_m^j \otimes A_{j+1} \qquad \text{I-35}$$

Definición I-2

Sea *A* una matriz por bloques del *tipo (m,n)*

$$A = \begin{bmatrix} A_{11} & A_{12} & . & . & . & A_{1m} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ A_{m1} & A_{m2} & . & . & . & A_{mm} \end{bmatrix} \qquad \text{I-36}$$

Con mxm bloques de orden n cada uno. Si cada bloque A_{ij} es a su vez una matriz circulante, diremos que A es una matriz con bloques circulantes y la denotaremos como $BC_{(m,n)}$. Vamos ahora a dar unos resultados como los que fueron dados para $CB_{(m,n)}$.

Teorema I-2

$A \in BC_{(m,n)}$ si y sólo si

$$A(I_m \otimes \Pi_n) = (I_m \otimes \Pi_n)A$$

Prueba:

$(I_m \otimes \Pi_n) \in BC_{(m,n)}$ y $(I_m \otimes \Pi_n) = diag(\Pi_n, \Pi_n, ..., \Pi_n)$, entonces $A(I_m \otimes \Pi_n) = (A_{jk}\Pi_n)_{1 \le j,k \le m}$ y $(I_m \otimes \Pi_n)A = (\Pi_n A_{jk})_{1 \le j,k \le m}$.

Así:

$A(I_m \otimes \Pi_n) = (I_m \otimes \Pi_n)A$, si y sólo si,

$(A_{jk}\Pi_n) = (\Pi_n A_{jk})$ *para* $j,k = 1,2,...,n$ y A_{jk} es un bloque de orden *n*; pero ésta igualdad es solo válida de acuerdo con la proposición [I.1], si A_{jk} es circulante, y como $1 \le j,k \le n$ son arbitrarios, cada bloque es circulante.

Teorema I-3

$A \in BC_{(m,n)}$ si y sólo si *A* es de la forma $A = \sum_{k=0}^{n-1} (A_{k+1} \otimes \Pi_n^k)$, donde A_{j+1} son matrices cuadradas de orden *m.*

Prueba: $A = A_{j,k} \in BC_{(m,n)}$, si y sólo si, para $1 \le j,k \le m$, se tiene:

$A_{jk} = circ(a_1^{(jk)}, a_2^{(jk)}, ..., a_n^{(jk)}) = \sum_{i=1}^{n} a_i^{(jk)} \Pi_n^{i-1}$, entonces:

$$A=\begin{bmatrix} a_1^{(11)}I_n & . & . & . & a_1^{(1m)}I_n \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ a_1^{(m1)}I_n & . & . & . & a_1^{(mm)}I_n \end{bmatrix}+\begin{bmatrix} a_2^{(11)}\Pi_n & . & . & . & a_2^{(1m)}\Pi_n \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ a_2^{(m1)}\Pi_n & . & . & . & a_2^{(mm)}\Pi_n \end{bmatrix}+\ldots$$
$$\ldots+\begin{bmatrix} a_m^{(11)}\Pi_n^{n-1} & . & . & . & a_m^{(1m)}\Pi_n^{n-1} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ a_m^{(m1)}\Pi_n^{n-1} & . & . & . & a_m^{(mm)}\Pi_n^{n-1} \end{bmatrix} \qquad \text{I-37}$$

Si $A_1=(a_1^{(jk)})_{1\le j,k\le m}$, $A_2=(a_2^{(jk)})_{1\le j,k\le m},\ldots,$ $A_n=(a_n^{(jk)})_{1\le j,k\le m}$, entonces $A=A_1\otimes I_n+A_2\otimes \Pi_n+\ldots+A_n\otimes \Pi_n^{n-1}$

Note que cada matriz $A_1,\ldots,A_n$ es de tamaño mxm. Si $A \in BC_{(m,n)}$, $A_1=(A_{jk})$ $1\le j,k\le m$. y cada bloque A_{jk} es circulante, entonces cada bloque puede ser diagonalizado, ver sección 1.8.6.

Combinemos las dos ideas expuestas en la presentación de matrices circulantes por bloques $CB_{(m,n)}$, y matrices con bloques circulantes $BC_{(m,n)}$.

Definición I-3

Sea A una matriz del tipo *(m,n),* si A es circulante por bloques y cada bloque es a su vez circulante, diremos que A es de clase $CBBC_{(m,n)}$.

Ejemplo:

$$A=\begin{bmatrix} a & b & c & d & e & f \\ b & a & d & c & f & e \\ e & f & a & b & c & d \\ f & e & b & a & d & c \\ c & d & e & f & a & b \\ d & c & f & e & b & a \end{bmatrix} \qquad \text{I-38}$$

Una matriz $CBBC_{(m,n)}$, no es necesariamente ella misma una matriz circulante. Esta matriz A puede definirse como:

$$A=\sum_{j=0}^{m-1}(\Pi_m^j \otimes A_{j+1}) \qquad \text{I-39}$$

1.8.5 Una segunda Representación de Circulantes

Aprovechando la estructura de las matrices de permutación Π^k , $k= 0,1,...N-1,$ es claro que:

$$C=circ(c_1, c_2,...,c_N)=c_1 I+c_2\Pi+....+c_N \Pi^{N-1} \qquad \text{I-40}$$

Por lo anterior, C es una circulante si y solo si $C= p(\Pi)$. Para algún polinomio $p(z)$ asociado con la n-tupla $\gamma=(c_1, c_2,...,c_N)$, el polinomio:

$$p_\gamma(z)=c_1+c_2 z+....+c_N z^{N-1} \qquad \text{I-41}$$

es llamado el *representante de la circulante*, la asociación $\gamma \leftrightarrow p_\gamma(z)$, es una asociación lineal, por esto:

$$C=circ\ \gamma=p_\gamma(\Pi) \qquad \text{I-42}$$

1.8.6 Diagonalización de Circulantes

Para la demostración de la diagonalización de las circulantes, partiremos del caso particular de la diagonalización de la matriz circulante básica Π.

Definición I-4 Sea N un entero fijo, mayor o igual a 1, sea $w = exp\ (2\pi/N)$, la matriz Ω se define como:

$$\Omega=(\Omega_N)=diag(1,w,w^2,...,w^{N-1}) \qquad \text{notese que:}$$
$$\Omega^k= diag(1,w^k,w^{2k},...,w^{(N-1)k}), \qquad \text{I-43}$$

Teorema I-4

$$\Pi=F^* \Omega F \qquad \text{I-44}$$

Prueba: La j-ésima fila de F^* es:

$$\left(\frac{1}{\sqrt{N}}\right)\left[w^{(j-1)0}, w^{(j-1)1}, w^{(j-1)2}, \ldots, w^{(j-1)(N-1)}\right] \qquad \text{I-45}$$

Por lo tanto, la j-ésima fila de $F^{*}\Omega$ es:

$$\left(\frac{1}{\sqrt{N}}\right)\left[w^{(j-1)r}.w^{r}\right]=\left(\frac{1}{\sqrt{N}}\right)\left[w^{(j)r}\right];\ r = 0,1,2\ldots,N-1 \qquad \text{I-46}$$

La k-ésima columna de F es:

$$\left(\frac{1}{\sqrt{N}}\right)\left[w^{-(k-1)r}\right];\quad r = 0,1,2\ldots,N-1 \qquad \text{I-47}$$

Por lo tanto el (j,k) ésimo elemento de $F^{*}\,\Omega F$ es:

$$\left(\frac{1}{N}\right)\sum_{r=0}^{N-1}\left[w^{(j)r}\,w^{-(k-1)r}\right]=\left(\frac{1}{N}\right)\sum_{r=0}^{N-1}\left[w^{r(j-k+1)}\right]=\begin{cases}1 \text{ if } j=k-1, \\ \\ 0 \text{ if } j\neq k-1,\end{cases} \quad mod\ N \qquad \text{I-48}$$

Con lo cual se comprueba I-45

Ahora de I-43 se tiene que:

$$\begin{gathered} C=circ\ \gamma=p_{\gamma}(\Pi)=p_{\gamma}(F^{*}\,\Omega F) \\ C=F^{*}\,p_{\gamma}(\Omega)F \\ C=F^{*}\,diag(p_{\gamma 0}(I), p_{\gamma 1}(\Omega), p_{\gamma 2}(\Omega^{2}),\ldots, p_{\gamma N-1}(\Omega^{(N-1)}))F \\ C=F^{*}\,(diag(F.p\gamma))F \end{gathered} \qquad \text{I-49}$$

Con lo que se llega al teorema fundamental

Teorema I-5

Si C es una circulante, esta es diagonalizada por F.

1.9 Convolución Lineal en dos Dimensiones

Sean x y h dos señales arbitrarias de dimensión Z, con dominio el conjunto $ZxZ= \{(n_o, n_1): n_o \in Z,\ n_1 \in Z\}$, la convolución lineal de estas dos señales, es una

nueva señal *y*, denotada por $y = x**h$, la cual es calculada como se muestra a continuación:

$$y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h[k_0, k_1].x[n_0 - k_0, n_1 - k_1]) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x[k_0, k_1].h[n_0 - k_0, n_1 - k_1]) \quad \text{I-50}$$

Como se puede observar en la anterior formulación, la operación de convolución en dos dimensiones, también es conmutativa.

1.10 Transformada de Fourier Discreta en el Tiempo (DTFT siglas en inglés)

La Transformada de Fourier Discreta en el Tiempo (DTFT) es la transformada de Fourier en frecuencia-continua de una señal discreta en tiempo, esto es:

$$X(e^{jw}) = \sum_{n=-\infty}^{\infty} x[n].e^{-jwn} \quad \text{I-51}$$

En donde $\left|X(e^{jw})\right|$ es periódica con periodo 2π

La Transformada Inversa de Fourier Discreta en el Tiempo (IDTFT) esta dada por:

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{jw}).e^{jwn} dw \quad \text{I-52}$$

1.11 Convolución de la Transformada Discreta de Fourier en el Tiempo

La DTFT de la convolución de dos señales es el producto de sus DTFT's, esto es:

$$y[n] = x[n] * h[n] \overset{DTFT}{\Leftrightarrow} Y(e^{jw}) = X(e^{jw}).H(e^{jw}) \quad \text{I-53}$$

Y la respuesta en frecuencia de un sistema lineal invariante en el tiempo (LTI) es:

$$h[n] \overset{DTFT}{\Leftrightarrow} H(e^{jw}) = \frac{Y(e^{jw})}{X(e^{jw})} \quad \text{I-54}$$

1.12 La Transformada de Fourier Discreta (DFT)

La transformada de Fourier de frecuencia discreta de una señal discreta, está limitada al rango $0 \le n \le N-1$, y se denota:

$$X[k] = \sum_{n=0}^{N-1} x[n].e^{-j\frac{2\pi kn}{N}}; \quad 0 \le k \le N-1 \qquad \text{I-55}$$

Algunas veces, es a su vez expresada como:

$$X[k] = \sum_{n=0}^{N-1} x[n].W_N^{kn}; \quad 0 \le k \le N-1; \qquad W_N = e^{-j\frac{2\pi}{N}} \qquad \text{I-56}$$

La Transformada Inversa de Fourier Discreta esta dada por:

$$x[n] = \frac{1}{N}\sum_{n=0}^{N-1} X[k].e^{+j\frac{2\pi kn}{N}}; \quad 0 \le n \le N-1 \qquad \text{I-57}$$

O en forma análoga:

$$x[n] = \sum_{n=0}^{N-1} X[k].W_N^{-kn}; \quad 0 \le n \le N-1; \qquad W_N = e^{-j\frac{2\pi}{N}} \qquad \text{I-58}$$

1.13 Convolución Cíclica y la DFT

Similarmente a como vimos en la DTFT, los productos de la DFT's de dos señales es la DFT de su convolución cíclica de *N*-puntos, la cual es representada así:

$$x_1[n] o_N x_2[n] = \sum_{m=0}^{N-1} x_1[m].x_2[\langle (n-m) \rangle_N] \overset{DFT}{\Leftrightarrow} X_1[k]X_2[k] \qquad \text{I-59}$$

Como ya se ha comprobado en el Teorema I-5, las matrices circulantes son diagonalizadas por la transformada de Fourier, ésta es la base para la reducción de la complejidad multiplicativa del proceso de convolución en el dominio de la frecuencia y de allí su gran utilidad.
Si una secuencia $y[n]$ es la convolución de dos secuencias $h[n]$ y $x[n]$, por la propiedad anterior en el dominio de la transformada z tenemos:

$$Y[z] = H[z] \cdot X[z] \qquad \text{I-60}$$

Donde $Y[z]$, $H[z]$ y $X[z]$ son las transformadas z de las secuencias $y[n]$, $h[n]$ y $x[n]$ respectivamente. Por ésta razón, métodos eficientes para realizar convolución de secuencias, lo son también para realizar multiplicación de polinomios y viceversa.

Lo que queremos decir cuando realizamos una transformación, no es nada más que una operación similar a la transformada de Fourier (DFT). Dada una señal periódica digital $x[n]$, hemos observado que su par en la transformada de Fourier discreta es:

$$X[k] = \sum_{n=0}^{N-1} x[n].W^{kn}$$
$$x[n] = -\frac{1}{N}\sum_{k=0}^{N-1} X[k].W^{-nk} \qquad \text{I-61}$$

Donde $N =$ número de muestras en un período de $x[n]$, y $W = exp(-\frac{j2\pi}{N})$.

La *propiedad de la convolución cíclica* relaciona la convolución $y[n]$ de dos señales periódicas y discretas $x[n]$ y $h[n]$ por medio de su DFT de la siguiente manera:

$$Y[k] = H[k] \cdot X[k] \quad k = 0,1,..., N-1 \qquad \text{I-62}$$

Donde $Y[k], H[k]$ y $X[z]$ son las DFT's de $y[n]$, $h[n]$ y $x[n]$ respectivamente, por eso, una forma alternativa de definir la convolución es:

$$y[n] = -\frac{1}{N}\sum_{k=0}^{N-1} H[k] \cdot X[k] \cdot W^{-nk} \quad n = 0,1,2...N-1 \qquad \text{I-63}$$

$H[k]$ y $X[k]$ pueden ser computadas en paralelo y luego realizarse los N productos $H[k] \cdot X[k]$.

En éste proyecto proponemos un enfoque ecléctico para lo cal haremos uso de la simetría de las matrices circulantes y las raíces de la unidad de la transformada de Fourier como herramientas para crear un algoritmo que mejore los resultados en el número de cómputos a realizar para el proceso de convolución, hasta el límite establecido por el teorema de Winograd, el cual establece:

Sea F algún campo polinomial. Hagamos que $X[z]$ y $H[z]$ sean dos polinomios de grado N definidos sobre ese campo. Entonces:

$$Y[z] = \langle H[z] \cdot X[z] \rangle_{P[z]} \qquad \text{I-64}$$

Requiere al menos $2N-k$ multiplicaciones, donde k es el número de factores irreducibles de $P[z]$ sobre el campo F. Si $P[z]$ es primo, entonces k es 1. Si $P[z] = (z^N - 1)$ (el caso de la convolución cíclica), entonces $k = N$ y el mínimo número de multiplicaciones es $2N - k = 2N - N = N$. Éste es el número de multiplicaciones que se logró obtener en el algoritmo que se mostrará en el capítulo 3.

1.14 La Transformada Rápida de Fourier (FFT)

La FFT es un algoritmo eficiente para calcular la transformada de Fourier Discreta (DFT) en una dimensión. La DFT de una señal discreta arbitraria en una dimensión x*(n),* de longitud *N*, está dada por:

$$X(k) = \sum_{n=-\infty}^{\infty} x[n] e^{\frac{-j(2\pi kn)}{N}} \; ; k = 0, 1, 2, 3 \ldots N\text{-}1; \quad j = \sqrt{-1} \qquad \text{I-65}$$

Expresando la función $F_N = [W_N^{kn}]$, donde $W_N = e^{\frac{-j(2\pi)}{N}}$, y *k, n* = 0, 1, 2,….. *N-1,* podemos obtener la representación matriz-vector de la operación de la DFT en una dimensión como sigue:

$$X(k) = F_N \times x[n] \qquad \text{I-66}$$

Como un ejemplo, tomaremos el caso para *N=4*:

$$X(k) = \sum_{n=0}^{3} x[n] W_4^{kn} = x(0) + x(1)\ W_4^k + x(2)\ W_4^{2k} + x(3)\ W_4^{3k} \qquad \text{I-67}$$

Esta puede ser representada en forma matricial como:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4 & W_4^2 & W_4^3 \\ 1 & W_4^2 & 1 & W_4^2 \\ 1 & W_4^3 & W_4^2 & W_4 \end{bmatrix} \times \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix} \qquad \text{I-68}$$

Para *N=4,* se tiene $W_4 = e^{\frac{-j2\pi}{4}} = -j$, por lo tanto,

$$F_4 = [(-j)^{(n.k)}] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \qquad \text{I-69}$$

Para dos dimensiones podemos llegar también a una forma análoga a I.67, la cual se representa de la siguiente forma:

$$X[j,k] = F_M \times (F_N \times x[n,m])^T \qquad \text{I-70}$$

con $m, j \in Z_M$ y $n, k \in Z_N$, donde m y n son las filas y columnas de la matriz a la cual se le desea obtener su transformada de Fourier en dos dimensiones. La prueba de ésta formula, se expondrá en el apéndice A.

1.15 Álgebra del Producto Kronecker o Producto Tensor

Describiremos a continuación algunas de las propiedades básicas de la operación con productos Kronecker. Procedemos entonces con la descripción de la matriz de DFT como una composición de matrices más pequeñas, algunas de ellas, expresadas en forma de producto Kronecker. La propiedad de poder descomponer a la matriz de Fourier F_N en matrices más pequeñas, es uno de los principales factores por los que se pueden conseguir implementaciones eficientes. Usaremos matrices cuadradas para definir el producto Kronecker, pero la definición generalmente aplica a matrices de cualquier dimensión.

Por ejemplo, sean A y B dos matrices de orden R y S, respectivamente, el producto Kronecker de A y B, es una operación binaria de la cual resulta una nueva matriz de orden $N = RxS$, denotada por $C = A \otimes B$, y que esta dada por:

$$A \otimes B = [a_{kl} B]_{k,l=0,1,\ \ ,R-1} \qquad \text{I-71}$$

Por ejemplo, tomando $R=S=2$, y haciendo

$$A = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix},\ B = \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} \qquad \text{I-72}$$

$$C = A \otimes B = [a_{kl} B] = \begin{bmatrix} a_{00}B & a_{01}B \\ a_{10}B & a_{11}B \end{bmatrix} = \begin{bmatrix} a_{00}b_{00} & a_{00}b_{01} & a_{01}b_{00} & a_{01}b_{01} \\ a_{00}b_{10} & a_{00}b_{11} & a_{01}b_{10} & a_{01}b_{11} \\ a_{10}b_{00} & a_{10}\,b_{01} & a_{11}b_{00} & a_{11}b_{01} \\ a_{10}b_{10} & a_{10}b_{11} & a_{11}b_{10} & a_{11}b_{11} \end{bmatrix} \qquad \text{I-73}$$

$$D = B \otimes A = [B_{kl} A] = \begin{bmatrix} b_{00}A & b_{01}A \\ b_{10}A & b_{11}A \end{bmatrix} = \begin{bmatrix} b_{00}a_{00} & b_{00}a_{01} & b_{01}a_{00} & b_{01}a_{01} \\ b_{00}a_{10} & b_{00}a_{11} & b_{01}a_{10} & b_{01}a_{11} \\ b_{10}a_{00} & b_{10}\,a_{01} & b_{11}a_{00} & b_{11}a_{01} \\ b_{10}a_{10} & b_{10}a_{11} & b_{11}a_{10} & b_{11}a_{11} \end{bmatrix}, \qquad \text{I-74}$$

Podemos notar claramente que $A \otimes B \neq B \otimes A$, lo cual nos dice que la operación no es conmutativa.

Ejemplo:

Sean $$A = I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad ; \quad B = F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \text{I-75}$$

Por lo tanto, $$C = A \otimes B = \begin{bmatrix} F_2 & 0 \\ 0 & F_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \qquad \text{I-76}$$

En general, $I_R \otimes F_S$ puede verse como una operación en paralelo, ya que los elementos que no son ceros, es decir, las matrices F_s, aparecen a lo largo de la diagonal. La matriz $I_R \otimes F_S$, es una matriz dispersa, y su implementación favorece una arquitectura en paralelo, esto puede ser demostrado con el siguiente ejemplo:

Sea $R = 4$ y $S = 2$, si calculamos $b = (F_s \otimes I_R)\, a$, que es una operación de multiplicación matriz-vector tenemos:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} \qquad \text{I-77}$$

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} I_4 & I_4 \\ I_4 & -I_4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} \qquad \text{I-78}$$

Podemos observar que las entradas en la matriz cuadrada son en si mismas matrices identidad de orden 4 cada una, entonces, ésta multiplicación matriz-vector puede realizarse en términos de segmentos de submatriz-vector como:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} I_4 \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} + I_4 \begin{bmatrix} a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} \\ I_4 \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} - I_4 \begin{bmatrix} a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} \\ \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} - \begin{bmatrix} a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} \end{bmatrix} \qquad \text{I-79}$$

En general, la operación $b = (F_s \otimes I_R)\, a$ puede ser computada en una máquina con capacidades de procesamiento de vectores. El vector de entrada a puede ser dividido en *S* segmentos de longitud *R* cada uno.

1.16 Formulación de algoritmos de FFT Usando el Producto Kronecker

Una matriz de DFT puede descomponerse en una secuencia de submatrices. Si *N* es el orden de la matriz de DFT es muy deseable que *N* sea un número compuesto, siendo $N = 2^M$ la mejor composición de números para la formulación del producto Kronecker [Rodriguez1]- [Rodriguez2].

La esencia para realizar la DFT de una señal discreta de *N*-puntos, es primero tomar $N = R \times S$, y luego expresar la suma de la DFT, en términos de *R* y *S*, como se muestra a continuación:

$$\tilde{y}[\ell] = \sum_{k=0}^{N-1} \tilde{x}[k] e^{\frac{-j2\pi\ell k}{N}} ; \quad k = 0,1.....N-1, \quad and\ j = \sqrt{-1} \qquad \text{I-80}$$

Esta función de DFT se puede expresar como:

$$\tilde{y} = (F_S \otimes I_R) T_{N,S} (I_S \otimes F_R) P_{N,S}\, \tilde{x} \qquad \text{I-81}$$

La matriz $T_{N,S}$ es una matriz especial llamada *"twiddle factor",* y está definida como:

$$T_{N,S} = diag[I_R, D_{R,N}, ...(D_{R,N})^{S-1}] \qquad \text{I-82}$$

Donde *diag,* significa diagonal, I_R es la matriz identidad de orden R y $D_{R,N}$ esta definida por:

$$D_{R,N} = diag[1, w_N, ...(w_N)^{R-1}] \qquad \text{I-83}$$

La derivación formal de ésta formula se encuentra en el apéndice A.

1.17 Producto Cartesiano en dos Dimensiones

Sean A y B dos conjuntos cualesquiera, el producto cartesiano de A y B en dos dimensiones es un nuevo conjunto conformado de la siguiente manera:

$$A \times B = \{(a,b) : a \in A \quad y \quad b \in B\} \qquad \text{I-84}$$

Los productos cartesianos formados por conjuntos de indexación natural de la forma $Z_n = \{1,2,3,...N-1\}$, juegan un papel importante en el presente trabajo. A continuación, describimos algunas propiedades asociadas a los productos cartesianos.

Una relación $\rho : A \to B$, asocia a todo elemento del conjunto A, con otro elemento del conjunto B, en éste caso, el conjunto A se denomina *dominio* y el conjunto B se denomina *Co-dominio.*

De ésta manera, podemos expresar la relación ρ como un subconjunto del producto cartesiano $A \times B$; esto es, $\rho \subset A \times B$.

Definimos una función como una relación en donde la primera entrada de cada elemento par-ordenado, de la relación aparece una sola vez. Por ejemplo, podemos decir que la función $x : Z_N \to C$, es una relación $x \subset Z_N \times C$, donde $x = \{n, x[n]\} : n \in Z_N, x[n] \in C$.

Los productos cartesianos se utilizan para describir operadores lineales, como por ejemplo, la convolución cíclica y la transformada discreta de Fourier, en espacios lineales como los espacios $\ell^2(Z_N)$ y $\ell^2(Z_{N0}, Z_{N1})$.

Todo operador lineal satisface las condiciones de superposición y homogeneidad, lo mismo ocurre para todo espacio lineal. En éste trabajo clasificamos los operadores lineales en *operadores unarios* y *operadores binarios.* A continuación damos ejemplos de los operadores más utilizados en la formulación de éste trabajo.

1.18 Representación Binaria del Operador de Convolución Cíclica de dos Dimensiones

$$N_0 \times N_1;\ \ell^2(Z_{N0} \times Z_{N1}) \times \ell^2(Z_{N0} \times Z_{N1}) \to \ell^2(Z_{N0} \times Z_{N1}) \qquad \text{I-85}$$
$$(x,h) \mapsto y = x *_{N0xN1} h$$

El operador binario $*_{N0xN1}$ convierte al espacio lineal $\ell^2(Z_{N0} \times Z_{N1})$, en el álgebra lineal de las convoluciones cíclicas de dos dimensiones. Decimos que $\ell^2(Z_{N0} \times Z_{N1})$ es un espacio lineal porque satisface las siguientes condiciones:

1) Superposición:
   Para $x_1, x_2 \in \ell^2(Z_{N0} \times Z_{N1})$, tenemos que $x_1 + x_2 \in \ell^2(Z_{N0} \times Z_{N1})$.
2) Homegenidad:
   Para $x_i \in \ell^2$ y $a \in C$, entonces:
   $ax_i \in \ell^2(Z_{N0} \times Z_{N1})$

Este espacio lineal se denomina espacio lineal de señales discretas finitas de dos dimensiones, o espacio de arreglos de dos dimensiones, también se le llama espacio vectorial de arreglos de dos dimensiones. Un arreglo implica un conjunto de valores con una estructura preescrita y la estructura es ordenada.

1.19 Representación Matricial del Operador Unario de la Convolución Cíclica de dos Dimensiones

La convolución cíclica de dos dimensiones admite una representación matricial partiendo de la siguiente formulación en términos de un operador unario.

Sea T_h el operador unario de la convolución cíclica en dos dimensiones, entonces tenemos la siguiente representación de su acción en el espacio lineal de señales de arreglos finitos de dos dimensiones. Un arreglo finito implica un arreglo discreto.

$$T_h : \ell^2(Z_{N0} \times Z_{N1}) \to \ell^2(Z_{N0} \times Z_{N1})$$
$$x \mapsto y = T\{x\} \quad donde$$
$$y[n_0,n_1] = \sum_{k_1 \in Z_{N1}} \sum_{k_0 \in Z_{N0}} h[k_0,k_1].x[<n_0-k_0>_{N0},<n_1-k_1>_{N1}] \ \ ó \qquad \text{I-86}$$
$$y[n_0,n_1] = \sum_{k_1 \in Z_{N1}} \sum_{k_0 \in Z_{N0}} x[k_0,k_1].h[<n_0-k_0>_{N0},<n_1-k_1>_{N1}]$$

La representación matricial del operador unario de la convolución cíclica admite dos formulaciones dependiendo si el arreglo de dos dimensiones que representa cada señal se expresa en forma de vector-columna, utilizando los formatos de "columna mayor" o "fila mayor", por ejemplo, el arreglo de dos dimensiones:

$$a = \begin{bmatrix} a_{[0,0]} & a_{[0,1]} & a_{[0,2]} \\ a_{[1,0]} & a_{[1,1]} & a_{[1,2]} \end{bmatrix} \qquad \text{I-87}$$

El cual representa a una señal finita cualquiera de dos dimensiones, admite las siguientes formulaciones:

1) *Formulación "Columna Mayor"*

Se produce un vector columna tomando en orden las columnas del arreglo a:

$$a_v = \begin{bmatrix} a_{[0,0]} \\ a_{[1,0]} \\ a_{[0,1]} \\ a_{[1,1]} \\ a_{[0,2]} \\ a_{[1,2]} \end{bmatrix}$$ Esta formulación se denomina *antilexicográfica* I-88

2) *Formulación "Fila Mayor"*

Se produce un vector columna tomando en orden las filas del arreglo a:

$$a_v = \begin{bmatrix} a_{[0,0]} \\ a_{[0,1]} \\ a_{[0,2]} \\ a_{[1,0]} \\ a_{[1,1]} \\ a_{[1,2]} \end{bmatrix}$$ Esta formulación se denomina *lexicográfica* I-89

Ejemplo: Convolución Cíclica en el espacio lineal $\ell^2(Z_2 \times Z_3)$:

$$\begin{aligned} &T_h : \ell^2(Z_2 \times Z_3) \to \ell^2(Z_2 \times Z_3) \\ &\qquad x \mapsto y = T\{x\} \quad donde \\ &y[n_0, n_1] = \sum_{k_1 \in Z_3} \sum_{k_0 \in Z_2} x[k_0, k_1].h[< n_0 - k_0 >_{N0}, < n_1 - k_1 >_{N1}] \end{aligned} \qquad \text{I-90}$$

$$y[n_0,n_1] = \sum_{k_1=0}^{2}\sum_{k_0=0}^{1} x[k_0,k_1].h[<n_0-k_0>_2,<n_1-k_1>_3]$$

Expandiendo las sumatorias tenemos:

$$y[n_0,n_1] = \sum_{k_1=0}^{2} x[0,k_1].h[<n_0>_2,<n_1-k_1>_3] + x[1,k_1].h[<n_0-1>_2,<n_1-k_1>_3]$$

$$\begin{aligned} y[n_0,n_1] = {} & x[0,0].h[<n_0>_2,<n_1>_3] + x[1,0].h[<n_0-1>_2,<n_1>_3] \\ & + x[0,1].h[<n_0>_2,<n_1-1>_3] + x[1,1].h[<n_0-1>_2,<n_1-1>_3] \\ & + x[0,2].h[<n_0>_2,<n_1-2>_3] + x[1,2].h[<n_0-1>_2,<n_1-2>_3]; \end{aligned}$$

El anterior producto representado en forma de columna mayor puede observarse como:

$$\begin{bmatrix} y[0,0] \\ y[1,0] \\ y[0,1] \\ y[1,1] \\ y[0,2] \\ y[1,2] \end{bmatrix} = \left[\begin{array}{cc|cc|cc} h[0,0] & h[1,0] & h[0,2] & h[1,2] & h[0,1] & h[1,1] \\ h[1,0] & h[0,0] & h[1,2] & h[0,2] & h[1,1] & h[0,1] \\ \hline h[0,1] & h[1,1] & h[0,0] & h[1,0] & h[0,2] & h[1,2] \\ h[1,1] & h[0,1] & h[1,0] & h[0,0] & h[1,2] & h[0,2] \\ \hline h[0,2] & h[1,2] & h[0,1] & h[1,1] & h[0,0] & h[1,0] \\ h[1,2] & h[0,2] & h[1,1] & h[0,1] & h[1,0] & h[0,0] \end{array}\right] \times \begin{bmatrix} x[0,0] \\ x[1,0] \\ x[0,1] \\ x[1,1] \\ x[0,2] \\ x[1,2] \end{bmatrix} \quad \text{I-91}$$

Es posible apreciar que la convolución cíclica en dos dimensiones cuando se expresa en forma de *"columna mayor",* resulta en una multiplicación de una matriz circulante por bloques, con bloques circulantes, por un vector. No ocurre lo mismo para la representación *"fila mayor",* tal como se observa a continuación.

$$\begin{bmatrix} y[0,0] \\ y[0,1] \\ y[0,2] \\ y[1,0] \\ y[1,1] \\ y[1,2] \end{bmatrix} = \begin{bmatrix} h[0,0] & h[1,0] & h[0,2] & h[1,2] & h[0,1] & h[1,1] \\ h[0,1] & h[1,1] & h[0,0] & h[1,0] & h[0,2] & h[1,2] \\ h[0,2] & h[1,2] & h[0,1] & h[1,1] & h[0,0] & h[1,0] \\ h[1,0] & h[0,0] & h[1,2] & h[0,2] & h[1,1] & h[0,1] \\ h[1,1] & h[0,1] & h[1,0] & h[0,0] & h[1,2] & h[0,2] \\ h[1,2] & h[0,2] & h[1,1] & h[0,1] & h[1,0] & h[0,0] \end{bmatrix} \times \begin{bmatrix} x[0,0] \\ x[0,1] \\ x[0,2] \\ x[1,0] \\ x[1,1] \\ x[1,2] \end{bmatrix} \quad \text{I-92}$$

Esta última representación no presenta ninguna característica que pueda ser asociada a las matrices circulantes. Es importante notar que las dos representaciones mostradas en las ecuaciones I-91 e I-92, corresponden a la misma operación de convolución cíclica.

Capítulo II.

2 La computación de la transformada rápida de Fourier.

En éste capítulo examinaremos las bases de la transformada rápida de Fourier. Haremos una introducción de cómo se derivaron los métodos para la realización de algoritmos eficientes para la transformada de Fourier discreta. Se prestará especial atención a la transformada de Fourier en base 2, que servirá como guía para la realización del algoritmo que se presentará en el capitulo 3.

2.1 Sección de cómputo elemental de la transformada rápida de Fourier

El término "Transformada rápida de Fourier" ha sido ampliamente usado para denotar algoritmos para la transformada de Fourier discreta (DFT), los cuales requieren menos operaciones, particularmente multiplicaciones, de lo que requiere la implementación directa de:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot W^{kn} \qquad k = 0,1,.....N-1 \qquad \text{II-1}$$

Esta terminología será la que usaremos de aquí en adelante para describir los algoritmos que serán investigados.

Consideremos entonces la ecuación para la transformada de Fourier discreta (DFT). Excepto si N es primo, puede siempre factorizarse en:

$$N = M \cdot L \qquad \text{II-2}$$

Ahora, realicemos una aplicación de $x(n)$ en un arreglo $x(u,v)$ el cual tiene M filas y L columnas. Esto es, en:

$$\begin{array}{llll} x(0) & x(1) & ..... & x(L-1) \\ x(L) & x(L+1) & ..... & x(2L-1) \\ . & & & \\ . & & & \\ x((M-1)\cdot L) & x((M-1)\cdot L+1) & ..... & x((M-1)\cdot L+L-1) \end{array} \qquad \text{II-3}$$

Matemáticamente, esta aplicación es obtenida mediante la transformación del indice n de acuerdo a:

$$n = u \cdot L + v$$
$$\text{donde } 0 \le n \le N-1,\ 0 \le u \le M-1,\ 0 \le v \le L-1. \qquad \text{II-4}$$

Una aplicación similar de $X(k)$ en un arreglo $X(s,r)$ puede ser definida. Aquí, es conveniente definir ésta aplicación como:

$$k = r \cdot M + s$$
$$\text{donde } 0 \le k \le N-1,\ 0 \le r \le L-1,\ 0 \le s \le M-1. \qquad \text{II-5}$$

Substituyendo estas transformaciones en la ecuación de la transformada de Fourier discreta (DFT), tenemos:

$$X(s,r) = \sum_{u=0}^{M-1} \sum_{v=0}^{L-1} x(u,v) \cdot W^{(rM+s)(uL+v)} \qquad \text{II-6}$$

Debido a que W es periodica en N, esta notación será cambiada a W_N para reflejar ésta propiedad. Entonces:

$$\begin{aligned} W_N^{(rM+s)(uL+v)} &= W_N^{ruML+rvM+suL+sv} \\ &= W_N^{ruN+(rvM+suL+sv)} \\ &= W_N^{rvM+suL+sv} \end{aligned} \qquad \text{II-7}$$

Por lo tanto:

$$X(s,r) = \sum_{u=0}^{M-1} \sum_{v=0}^{L-1} x(u,v) \cdot \ W_N^{rvM+suL+sv}$$

Esta última ecuación puede ser descompuesta en tres acciones separadas, llamemoslas:

$$X(s,r) = \sum_{v=0}^{L-1} W_N^{rvM} \left[W_N^{sv} \cdot \left[\sum_{u=0}^{M-1} x(u,v) W_N^{suL} \right] \right] \qquad \text{II-8}$$

Estas tres etapas son como se describen a continuación:

ETAPA 1

$$P(s,v) = \sum_{u=0}^{M-1} x(u,v) W_N^{suL}$$
$$\text{donde } \ 0 \le s \le M-1,\ 0 \le v \le L-1 \qquad \text{II-9}$$

Ahora:

$$W_N^{suL} = \left[exp\left(-\frac{j2\pi}{N} \right) \right]^{suL}$$

$$W_N^{suL} = \left[exp\left(-\frac{j2\pi}{ML} \right) \right]^{suL}$$

$$W_N^{suL} = \left[exp\left(-\frac{j2\pi}{M} \right) \right]^{su}$$

$$= W_M^{su}$$

Por lo tanto:

$$P(s,v) = \sum_{u=0}^{M-1} x(u,v) W_M^{su}$$

Esto describe un conjunto de transformadas de Fourier a lo largo de las columnas del arreglo $x(u,v)$. Los cómputos de cada uno de éstas ecuaciones requieren $(M^2 - 2M + 1)$ multiplicaciones complejas y $M \cdot (M-1)$ adiciones complejas. Hay L de tales transformadas de Fourier para computar, por lo tanto ésta etapa requiere $(M^2 - 2M + 1) \cdot L$ multiplicaciones complejas y $M \cdot L \cdot (M-1)$ adiciones complejas.

ETAPA 2

$$Q(s,v) = W_N^{sv} \cdot P(s,v)$$

donde $0 \le s \le M-1,\ 0 \le v \le L-1$ II-10

El arreglo resultante de la primera etapa ésta multiplicado por un conjunto de factores W_M^{sv}. Debido a que estos términos todos tienen una magnitud de la unidad pero varían sus fases, estos términos son llamados *factores de rotación de fases.* Otro término mucho más poético y más ampliamente usado es *factores de vuelta ligera* (*twiddle factors en inglés)* con lo que a ésta etapa se le denomina etapa de *giramiento* (*twiddling en inglés).* Solo multiplicaciones complejas están involucradas en ésta etapa, y hay al menos $(M \cdot L - M - L + 1)$ de ellas.

ETAPA 3

$$X(s,r) = \sum_{v=0}^{L-1} Q(s,v) W_N^{rvM}$$

donde $0 \le s \le M-1,\ 0 \le r \le L-1$ II-11

Siguiendo la derivación vista en la etapa 1 tenemos:

$$W_N^{rvM} = W_L^{rv}$$ II-12

Por lo tanto:

$$X(s,r) = \sum_{v=0}^{L-1} Q(s,v) \cdot W_L^{rv}$$

Esto no es más que la computación de transformadas de Fourier discretas de L-puntos, a lo largo del arreglo $Q(s,v)$ resultante de la etapa 2 y una aplicación del resultado dentro del arreglo $X(s,r)$. Esto requiere en total $M \cdot (L^2 - 2L + 1)$ multiplicaciones complejas y $M \cdot L \cdot (L-1)$ adiciones complejas.
La ecuación original para la transformada de Fourier discreta de N-puntos requería $(N^2 - 2N + 1)$ multiplicaciones complejas y $N \cdot (N-1)$ adiciones complejas. Una sumatoria de las operaciones a lo largo de éste nuevo algoritmo nos da que éste requiere solo:

$$\begin{array}{ll} (M+L) \cdot N - 3N + 1 & \textit{multiplicaciones complejas} \\ N \cdot (M + L - 2) & \textit{adiciones complejas} \end{array} \qquad \text{II-13}$$

Hay claramente un número menor de operaciones. De hecho, cuando N se hace lo suficientemente grande, éste algoritmo requiere tanto para multiplicaciones complejas y adiciones complejas, la fracción:

$$\frac{2}{\sqrt{N}} \quad \text{si:} \qquad \text{II-14}$$

$$M \approx L \approx \sqrt{N}$$

Una revisión del procedimiento anterior nos muestra que pueden existir formulaciones alternativas. La aplicación inicial para el arreglo pudo haber sido de la siguiente forma:

$$\begin{array}{lll} x(0) & x(M) & \ldots\ldots x(M\cdot(L-1)) \\ x(L) & x(M+1) & \ldots\ldots x(M\cdot(L-1+1)) \\ . & & \\ . & & \\ . & & \\ x((M-1) & x(2M-1) \ldots\ldots & x(M\cdot(L-1)+M-1) \end{array} \qquad \text{II-15}$$

En éste caso, la aplicación pudo haberse escogido como:

$$n = u + v \cdot M \qquad \text{II-16}$$

donde $0 \le n \le N-1,\ 0 \le u \le M-1,\ 0 \le v \le L-1$, y similarmente:

$$k = r + s \cdot L$$

Donde $0 \le k \le N-1,\ 0 \le r \le L-1,\ 0 \le s \le M-1$. Lo cual resulta en:

$$X(s,r) = \sum_{u=0}^{M-1} W_M^{su} \left[W_N^{ru} \cdot \left[\sum_{v=0}^{L-1} x(u,v) W_L^{rv} \right] \right]$$

Analizando ésta nueva formulación se puede observar que ésta requiere exactamente el mismo número de operaciones que requería la forma previa. Consecuentemente, estos procedimientos pueden ser descritos colectivamente como *secciones de cómputo elemental de la transformada rápida de Fourier*.

Un comentario importante para realizar en este punto, es que los algoritmos deseables son aquellos que no tienen necesidad de ningún almacenamiento adicional más que el requerido para la entrada de la data. Estos algoritmos se denominan *en el lugar (in-place)*. Dichos algoritmos están caracterizados por cálculos en los que en cada etapa los términos de entrada y de salida están caracterizados por computaciones como:

$$\begin{aligned} X(n) &= x(n) + x(n+1) \\ X(n+1) &= x(n) - x(n+1) \end{aligned} \qquad \text{II-17}$$

2.2 Transformada rápida de Fourier en Base-2

Además de haber sido las primeras en ser desarrolladas, las más importantes transformadas de Fourier son aquellas en que el orden N es igual a una potencia de 2. Hay tres razones principales para su popularidad; Primero, N igual a una potencia de 2 es un número conveniente para computar. La otra es que todas las multiplicaciones complejas en base 2 de la transformada rápida de Fourier tienen factores de vuelta ligera (twiddle factors) como:

$$\begin{aligned} X(0) &= x(0) + x(1) \\ X(1) &= x(0) - x(1) \end{aligned} \qquad \text{II-18}$$

Por último, las operaciones de ordenamiento resultan mucho más simples.

Se puede expresar el algoritmo de 2.1 en base-B como un algoritmo de base-2 de la siguiente manera. El conjunto inicial de DFT están dadas por:

$$\begin{aligned} X_1(2I_1) &= x(2I_1) + x(2I_1+1) \\ X_1(2I+1_1) &= x(2I_1) - x(2I_1+1) \\ \text{donde} \quad I_1 &= 0,1,\ldots,\frac{N}{2}-1 \end{aligned} \qquad \text{II-19}$$

Todas las etapas subsecuentes involucran factores de giro y transformadas discretas en base-2. Para la L-ézzsima etapa tenemos:

$$X_L(k+2^{L-1}+2^L\cdot I_2)=W_{2L}^{KI_1}\cdot X_{L-1}(k+2^{L-1}\cdot I_1+2^L\cdot I_2)$$

donde

$$k=0,1,....,2^{L-1}-1$$

$$I_2=0,1,....,\frac{N}{2^L}-1$$

II-20

El número de DFT's de dos puntos es $\frac{N}{2}\cdot Log_2 N$ y por lo tanto, el número de adiciones es $N\cdot Log_2 N$. Solamente los factores de giro (twiddles factors) contribuyen a las multiplicaciones, por esto, el número de éstas son $\left(\frac{N}{2}\cdot Log_2 N-N+1\right)$.

Todos los algoritmos basados en potencias de 2 tienen un número de multiplicaciones en las etapas de giro (twiddling) por j solamente. Encontrar y ejecutar por separado éstas multiplicaciones, introduce gastos indirectos en el software que compensa a menudo las ganancias. La primera etapa de twiddling en un algoritmo de base-2 a menudo involucra solamente una multiplicación y esta es por j. Por ésta razón, existe alguna ventaja en la implementación de la DFT inicial más la primera etapa separadamente del resto del algoritmo (como una especie de preprocesamiento).

Los primeros algoritmos para la transformada rápida de Fourier en base-2 fueron desarrollados siguiendo los siguientes lineamientos. Dividir el conjunto N de puntos de la data $x(n)$ en dos secuencias de $\frac{N}{2}$ puntos cada uno, llamémoslas $x_1(n)$ y $x_2(n)$ donde:

$$X_1(n)=x(2n)$$
$$X_2(n)=x(2n+1)$$

II-21

De aquí proviene el término de decimación (diezmar) en tiempo. Para el algoritmo de decimación en la frecuencia la división es solo:

$$X_1(n)=x(n) \qquad n=0,1,.....,\frac{N}{2}-1$$

$$X_2(n)=x(n) \qquad n=\frac{N}{2},.....,N-1$$

II-22

Y quizás no es tan obvio el porque del nombre.

La ecuación para la DFT viene siendo:

$$\begin{aligned} X(k) &= \sum_{n=0}^{\frac{N}{2}-1} x(2n)\cdot W_N^{k\cdot 2n} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)\cdot W_N^{k\cdot(2n+1)} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x_1(n)\cdot W_N^{k\cdot 2n} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_2(n)\cdot W_N^{k\cdot(2n)} \end{aligned} \qquad \text{II-23}$$

Como W es periódica en N entonces W^2 es periódica en $\frac{N}{2}$. Por lo tanto, la ecuación se convierte en:

$$X(k) \quad = X_1(k) + W_N^k \cdot X_2(k) \qquad \text{II-24}$$

Donde $X_1(k)$ es la DFT de $\frac{N}{2}$-puntos de $x_1(n)$ y $X_2(k)$ es la DFT de $\frac{N}{2}$-puntos de $x_2(n)$, y estas pueden ser interpretadas como:

$$\begin{aligned} X(k) &= X_1(k) \quad + W_N^k \cdot X_2(k) \qquad k = 0,1,\ldots\ldots,\frac{N}{2}-1 \\ &= X_1\left(k-\frac{N}{2}\right) + W_N^k \cdot X_2\left(k-\frac{N}{2}\right) \qquad k = \frac{N}{2},\ldots\ldots,N-1 \end{aligned} \qquad \text{II-25}$$

Debido a la periodicidad de W_N, esta puede ser expresada como:

$$\begin{aligned} X(k) &= X_1(k) \quad + W_N^k \cdot X_2(k) \qquad k = 0,1,\ldots\ldots,\frac{N}{2}-1 \\ &= X_1\left(k-\frac{N}{2}\right) - W_N^{\left(k-\frac{N}{2}\right)} \cdot X_2\left(k-\frac{N}{2}\right) \quad k = \frac{N}{2},\ldots\ldots,N-1 \end{aligned} \qquad \text{II-26}$$

Esta sección de cómputo de los elementos describe un algoritmo in-place. Debido a una descripción gráfica este es llamado *mariposa* (*butterfly)*. El nombre se torna mucho más obvio cuando la sección es expresada como:

$$
\begin{aligned}
X(k) &= X_1(k) + W_N^k \cdot X_2(k) \\
X\left(k+\frac{N}{2}\right) &= X_1(k) - W_N^k \cdot X_2(k) \\
&for\ k = 0,1,....,\frac{N}{2}-1.
\end{aligned}
\qquad \text{II-27}
$$

La siguiente tabla muestra el número de multiplicaciones que son necesarias para calcular la DFT en base-2 y en base-4:

N	*Base-2 FFT*	*Base-4 FFT*
4	0	0
8	5	-
16	17	9
32	49	-
64	129	81
128	321	-
256	769	513

Tabla 1. Número de multiplicaciones para una DFT en base-2 y en base-4 para diferentes valores de N.

Capítulo III.

Antes de comenzar con la derivación de nuestro algoritmo, presentaremos un par de definiciones como introducción a este capítulo.

Algoritmo

Un algoritmo aritmético es una secuencia finita de pasos que computa exactamente un conjunto deseado de salidas de un conjunto dado de entradas.

Campo de constantes

Algunas multiplicaciones y divisiones son triviales, tales como la multiplicación y división por la unidad. Para prevenir que éstas sean contadas un conjunto base (*ground set)* G, algunas veces llamado el *campo de las constantes,* es especificado. Cualquier multiplicación o división por un elemento de G, no será tomada en el conteo de las multiplicaciones/divisiones incluidas en nuestro algoritmo.

3 Desarrollo del Algoritmo.

En el desarrollo del presente algoritmo, examinaremos en principio la complejidad multiplicativa del producto de una matriz circulante de orden $N = 2^s$ por un vector (Problema de la convolución cíclica), para realizar esto, empezaremos por el caso básico de orden $N = 2$, e iremos aumentando el orden (siempre en potencias de 2), para de esta manera obtener el algoritmo recursivo que define el proceso de convolución de la manera más eficiente. Aunque es viable seguir un tipo de enfoque similar al establecido en el capítulo 2 para la decimación en tiempo de la transformada de Fourier, preferimos el camino que se presentará a continuación por parecernos que puede ofrecer una mayor claridad al lector.

Sea $\mathbf{y} = \mathbf{X} \cdot \mathbf{h}$ una multiplicación matriz circulante- vector, donde $N = 2$ es el orden de las secuencias, entonces:

$$\mathbf{y} = \begin{bmatrix} x_1 & x_2 \\ x_2 & x_1 \end{bmatrix} \cdot \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} x_1 h_1 + x_2 h_2 \\ x_1 h_2 + x_2 h_1 \end{bmatrix} \qquad \text{III-1}$$

La computación directa es realizada por medio de 4 multiplicaciones y 2 sumas. Winograd [Winograd] demostró que para realizar $CF(Z_2)$,[2] el siguiente algoritmo puede ser usado, el cual necesita solo 2 multiplicaciones y 6 sumas:

[2] $CF(Z_2)$ es campo complejo orden 2

$$\mathbf{y} = \begin{bmatrix} x_1 & x_2 \\ x_2 & x_1 \end{bmatrix} \cdot \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} m_1 + m_2 \\ m_1 - m_2 \end{bmatrix},$$

donde:

$$m_1 = \frac{1}{2}(x_1 + x_2)(h_1 + h_2), \ ;$$

$$m_2 = \frac{1}{2}(x_1 - x_2)(h_1 - h_2), \ entonces \qquad \text{III-2}$$

$$\mathbf{y} = \begin{bmatrix} x_1 h_1 + x_2 h_2 \\ x_1 h_2 + x_2 h_1 \end{bmatrix}$$

Es nuestro interés reducir la complejidad de las multiplicaciones debido a que éstas requieren mayor costo computacional que las sumas. Por otro lado, las multiplicaciones por las constantes $\frac{1}{2}$ y -1, no serán tenidas en cuenta en el conteo de las multiplicaciones necesarias para realizar la operación debido al hecho de que no son parte de las secuencias a convolucionar, por esto, ellas se tomarán como pertenecientes a nuestro campo de constantes, o conjunto base G, que escogeremos por conveniencia (que se hará evidente en las subsiguientes demostraciones), como el campo de los números complejos [Winograd].

Aunque a primera vista nuestro algoritmo no es tan eficiente debido a que el número de sumas se ve incrementado comparativamente a la computación directa, demostraremos más adelante que para orden de secuencias $N \geq 16$ el número de sumas son significativamente menores que la multiplicación directa.

Este algoritmo de división es la base de nuestro trabajo, la implementación de nuestro algoritmo de convolución cíclica se realiza sobre éste concepto.

Con éste primer ejemplo no es muy sencillo comenzar a ver relaciones, sin embargo, con un poco de astucia se puede ver que las constantes empleadas en nuestro algoritmo no son otras que las raíces de la unidad del polinomio $z^N - 1$, en nuestro caso $z^2 - 1 \ (z = 1 \, y \, z = -1)$, y que el valor de la constante $\frac{1}{2}$ no es otra cosa que $\frac{1}{N}$.

También aquí comenzamos a observar la relación con los algoritmos en base-2 para la transformada de Fourier estudiados en la sección 2.2, para éste caso, las constantes son las mismas, e incluso sus diagramas de *mariposa (butterfly)* son esencialmente iguales.

La siguiente figura presenta una manera de como éste algoritmo puede ser aplicado en una estructura de hardware en paralelo:

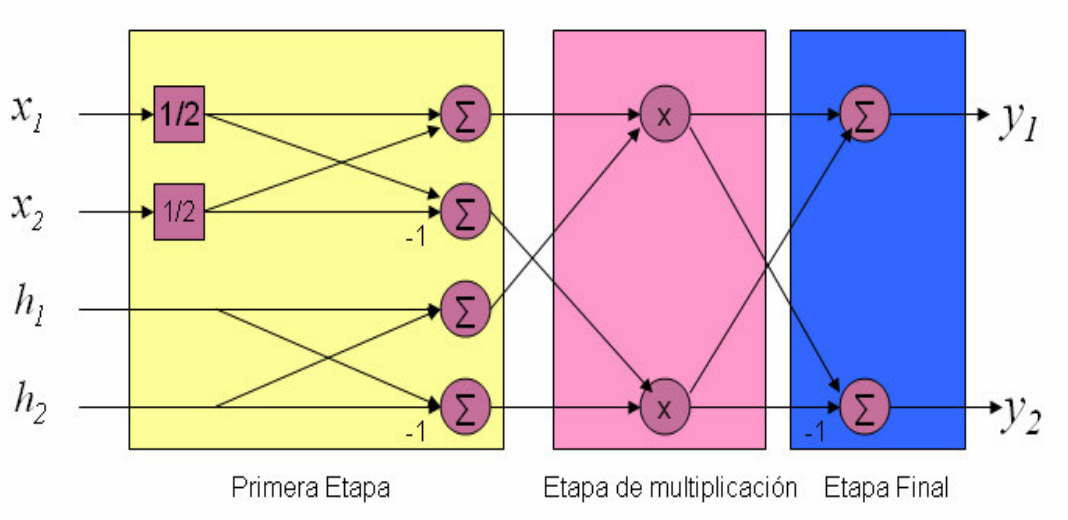


Figura 17. ***Aplicación para convolución cíclica para un orden N=2.***

Podemos observar en la figura 17 como en la primera etapa las secuencias de entrada $x[n]$ y $h[n]$ son computadas en paralelo. La multiplicación por la constante $\frac{1}{2}$ ($\frac{1}{N}$) pudo haber sido aplicada a $x[n]$ ó a $h[n]$, o incluso dejarla al final de todo el proceso y el resultado sería el mismo.

Con base en la figura 17, podemos realizar una comparación de nuestro algoritmo con aquellas implementaciones de algoritmos de convolución que usan la FFT base-2 presentada en la sección 2.2. Aquí se notan claramente las tres etapas que posee nuestro algoritmo, una primera etapa que es muy similar al proceso de la transformada de Fourier, una segunda etapa que puede asemejarse a la multiplicación de las secuencias "transformadas" y una tercera etapa que es similar al proceso de transformada inversa. Las etapas de transformación (primera y final) de éste algoritmo pueden ser obtenidas mediante un enfoque similar al descrito en la sección 2.2, para la transformada de Fourier Discreta (DFT) en base-2. Nuestro algoritmo presenta S etapas [3] de multiplicación por las raíces del polinomio $z^N - 1$ y S etapas que son similares al proceso de transformada inversa mostrados en 2.2. Para éste primer caso la etapa es solo $S = 1$ para la multiplicación por las raíces $z^2 - 1$ $(z = 1 \, y \, z = -1)$. Sin embargo, el hecho más importante radica en que solo necesitamos una etapa de multiplicación de tamaño $N = 2$.

[3] Donde S se ha definido como la potencia del orden de las secuencias $N=2^S$

Nuevamente resulta bastante sencillo verificar en la figura 17 que el número total de multiplicaciones son solo dos y el número total de sumas son 6.

Como habíamos mencionado anteriormente se puede notar en la figura 17 la característica de mariposa (butterfly), sin embargo debe quedar claro en este punto que el proceso que estamos siguiendo es un proceso de decimación en tiempo y no el enfoque de la convolución que hace uso de la DFT.

Podemos sacar algunas características que se repetirán a lo largo de las sucesivas derivaciones de los algoritmos, las cuales corroboraremos en los órdenes subsiguientes, y estas son:

1) *(3)(S)* etapas de multiplicación por las raíces del polinomio $z^N - 1$, y de la constante *(1/N)* de nuestro campo de constantes (no se cuentan como multiplicaciones dentro de nuestra derivación por pertenecer al campo de nuestras constantes).

Para el caso de *N=2*, tenemos 3 etapas de multiplicación por las raíces y por la constante 1/2.

2) (*3*)*(S)* etapas de sumas, cada una de orden *N,* por lo tanto (*3)(N)(log2(N))* sumas. Para el caso de *N=2*, tenemos 3 etapas de sumas cada una de orden 2, entonces tenemos 6 sumas en total.

3) 1 etapa de multiplicación de orden *N* por los elementos de las secuencias que están siendo procesadas, es decir, *N* multiplicaciones (son las únicas a tener en cuenta en cuanto a número de multiplicaciones de los algoritmos).
Para el caso *N=2*, solo 2 multiplicaciones a realizar.

En éste punto incrementaremos el orden del polinomio a la próxima potencia de 2, digamos $N = 2^2 = 4$, con el objetivo de mostrar de una manera más clara como ocurre el proceso de decimación en tiempo de nuestro algoritmo:

$$\mathbf{y} = \left[\begin{array}{cc|cc} x_1 & x_4 & x_3 & x_2 \\ x_2 & x_1 & x_4 & x_3 \\ \hline x_3 & x_2 & x_1 & x_4 \\ x_4 & x_3 & x_2 & x_1 \end{array}\right] \cdot \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} \qquad \text{III-3}$$

Podemos representar a la matriz circulante $\mathbf{X}$ y al vector $\mathbf{h}$ como:

$$\mathbf{X}=\begin{bmatrix}\mathbf{X_1} & \mathbf{X_2}\\ \mathbf{X_2} & \mathbf{X_1}\end{bmatrix},\quad \mathbf{h}=\begin{bmatrix}\mathbf{h_1}\\ \mathbf{h_2}\end{bmatrix},$$

$$\mathbf{y}=\mathbf{Xh} \qquad \text{III-4}$$

$$\mathbf{y}=\begin{bmatrix}\mathbf{X_1h_1}+\mathbf{X_2h_2}\\ \mathbf{X_1h_2}+\mathbf{X_2h_1}\end{bmatrix}$$

Y ahora, aunque estamos tratando con matrices y vectores podemos utilizar el mismo algoritmo presentado en III-2 para resolver la anterior multiplicación, es decir:

$$\mathbf{y}=\begin{bmatrix}\mathbf{y_1}\\ \mathbf{y_2}\end{bmatrix}=\begin{bmatrix}\mathbf{X_1} & \mathbf{X_2}\\ \mathbf{X_2} & \mathbf{X_1}\end{bmatrix}.\begin{bmatrix}\mathbf{h_1}\\ \mathbf{h_2}\end{bmatrix}=\begin{bmatrix}\mathbf{m_1}+\mathbf{m_2}\\ \mathbf{m_1}-\mathbf{m_2}\end{bmatrix},$$

donde:

$$\mathbf{m_1}=\frac{1}{2}(\mathbf{X_1}+\mathbf{X_2})(\mathbf{h_1}+\mathbf{h_2}),\ y \qquad \text{III-5}$$

$$\mathbf{m_2}=\frac{1}{2}(\mathbf{X_1}-\mathbf{X_2})(\mathbf{h_1}-\mathbf{h_2})$$

Ahora bien, los vectores $\mathbf{m_1}$ y $\mathbf{m_2}$ se encuentran realizando las siguientes computaciones:

Para $\mathbf{m_1}$ llamaremos

$$\begin{bmatrix}x_1' & x_2'\\ x_2' & x_1'\end{bmatrix}=\begin{bmatrix}x_1+x_3 & x_4+x_2\\ x_2+x_4 & x_1+x_3\end{bmatrix}$$

$$\begin{bmatrix}h_1'\\ h_2'\end{bmatrix}=\begin{bmatrix}h_1+h_3\\ h_2+h_4\end{bmatrix}, \text{tenemos } 4 \text{ sumas.} \qquad \text{III-6}$$

Ahora para la multiplicación para obtener $\mathbf{m_1}$

$$\mathbf{m_1}=\frac{1}{2}\begin{bmatrix}x_1' & x_2'\\ x_2' & x_1'\end{bmatrix}.\begin{bmatrix}h_1'\\ h_2'\end{bmatrix}$$

Tiene las mismas propiedades que III-2, por lo tanto puede ser realizado usando 2 multiplicaciones y 6 sumas.

Para el vector $\mathbf{m_2}$; seguimos un procedimiento similar:

$$\begin{bmatrix} x_3' & -x_4' \\ x_4' & x_3' \end{bmatrix} = \begin{bmatrix} x_1 - x_3 & x_4 - x_2 \\ x_2 - x_4 & x_1 - x_3 \end{bmatrix}, \text{entonces}$$

$$\begin{bmatrix} h_3' \\ h_4' \end{bmatrix} = \begin{bmatrix} h_1 - h_3 \\ h_2 - h_4 \end{bmatrix}, \text{ 4 sumas.} \qquad \text{III-7}$$

Ahora la multiplicación para obtener $\mathbf{m_2}$

$$\mathbf{m_2} = \frac{1}{2}\begin{bmatrix} x_3' & -x_4' \\ x_4' & x_3' \end{bmatrix}.\begin{bmatrix} h_3' \\ h_4' \end{bmatrix}$$

En éste punto aparece por primera vez una estructura diferente, sin embargo, la misma ésta estrechamente relacionada con una matriz circulante. Para resolver éste tipo de ecuación en la practica diferentes enfoques pueden ser usados, normalmente ésta clase de multiplicación con matrices relacionadas a las Toeplitz, es resuelta por medio de un enfoque con circulantes, la matriz problema se hace encajar en una matriz circulante de tamaño *2N* por *2N*, y se resuelve la multiplicación por medio de los algoritmos rápidos para la transformada de Fourier FFT. Nosotros no usaremos esta clase de procedimiento, ya que podemos resolver ésta multiplicación mediante un nuevo algoritmo de la siguiente manera [Winograd]:

$$\mathbf{m_2} = \frac{1}{2}\begin{bmatrix} x_3' & -x_4' \\ x_4' & x_3' \end{bmatrix}.\begin{bmatrix} h_3' \\ h_4' \end{bmatrix}$$

$$\mathbf{m_2} = \frac{1}{2}\begin{bmatrix} (r_1 + r_2) \\ (r_1 - r_2)/i \end{bmatrix}, \text{ d}onde \qquad \text{III-8}$$

$$r_1 = \frac{1}{2}(x_3' + ix_4')(h_3' + ih_4')$$

$$r_2 = \frac{1}{2}(x_3' - ix_4')(h_3' - ih_4')$$

Como podemos apreciar el computo de las sumas de las matrices $(\mathbf{X_1} + \mathbf{X_2})$, y $(\mathbf{X_1} - \mathbf{X_2})$ son realizadas mediante solo dos sumas individuales. Lo mismo ocurre con el calculo de la suma de los vectores $(\mathbf{h_1} + \mathbf{h_2})$ y $(\mathbf{h_1} - \mathbf{h_2})$, Otra cosa de especial interés es que la primera parte de nuestro desarrollo fue realizado por medio de la multiplicación por las constantes *1* y *-1*, y ésta parte final fue realizada mediante la multiplicación por las constantes *i* e *-i*.

Es importante apreciar el empleo de la constante *i*, la cual puede ser encontrada al tomarle la raíz cuadrada a la división de $\frac{-x_4}{x_4}$, es decir, $\sqrt{\frac{-x_4}{x_4}} = \sqrt{-1} = i$, este hecho es de fundamental importancia ya que en los subsiguientes ordenes se observará la

aparición de nuevas constantes para nuestros algoritmos, pero que parten de la misma premisa anterior, y que se puede describir como, "la raíz de la constante que debe multiplicarse al elemento de la matriz para volverla circulante".

Antes de continuar relacionando los resultados obtenidos, veamos que sucede con la suma de los vectores para el cálculo del resultado general:

$$\mathbf{v} = \begin{bmatrix} \mathbf{v_1} \\ \mathbf{v_2} \end{bmatrix} = \begin{bmatrix} \mathbf{m_1} + \mathbf{m_2} \\ \mathbf{m_1} - \mathbf{m_2} \end{bmatrix}, \text{ llamando:}$$

$$\mathbf{m_1} = \begin{bmatrix} x_1'' \\ x_2'' \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_1' & x_2' \\ x_2' & x_1' \end{bmatrix} \cdot \begin{bmatrix} h_1' \\ h_2' \end{bmatrix}.$$

y

$$\mathbf{m_2} = \begin{bmatrix} x_3'' \\ x_4'' \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_3' & -x_4' \\ x_4' & x_3' \end{bmatrix} \cdot \begin{bmatrix} h_3' \\ h_4' \end{bmatrix} \qquad \text{III-9}$$

Entonces,

$$\mathbf{y} = \begin{bmatrix} \mathbf{y_1} \\ \mathbf{y_2} \end{bmatrix} = \begin{bmatrix} \mathbf{m_1} + \mathbf{m_2} \\ \mathbf{m_1} - \mathbf{m_2} \end{bmatrix} = \begin{bmatrix} x_1'' + x_3'' \\ x_2'' + x_4'' \\ x_1'' - x_3'' \\ x_2'' - x_4'' \end{bmatrix}$$

Realizado por medio de 4 sumas.

Es muy importante notar dos aspectos relevantes de nuestro desarrollo. Primero, la multiplicación por la constante $\frac{1}{2}$ fue realizada en dos ocasiones, por lo tanto, éstas dos multiplicaciones pueden ser realizadas por medio de una sola constante de valor $\frac{1}{4}$, y como podemos intuir ese valor no es otro que $\frac{1}{N}$, y segundo, las otras constantes empleadas son las raíces del polinomio $z^N - 1$, en este caso $z^4 - 1$ cuyos valores son $[(z-1)(z+1)(z-i)(z+i)]$. Sin embargo, el resultado más importante obtenido es que la multiplicación de una matriz circulante por un vector de orden $N = 2^2 = 4$ es realizado mediante el uso de 4 multiplicaciones y 24 sumas.

Un poco de análisis nos permite observar que nuevamente, las tres características mostradas por el algoritmo para *N=2*, se ponen de manifiesto ahora para *N=4*, estas son:

1) Para el caso de *N=4*, tenemos *(3)(S)=(3)(2) =6* etapas de multiplicación por las raíces y por la constante 1/4.

2) *(3)(N)(log2(N))* sumas. Para el caso de *N=4*, tenemos *(3)(4)(2)=24* sumas en total.

3) 1 etapa de multiplicación de orden *N=4* por los elementos de las secuencias que están siendo procesadas.

Estos resultados pueden observarse con mejor claridad en la aplicación presentada en la figura 18, la cual presenta además, un ejemplo práctico de cómo es el flujo de la data, para el proceso de convolución de las siguientes dos secuencias:

$$x = [0,10,0]'$$
$$h = [0,1,0,0]'$$
$$y = x * h = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad \text{III-10}$$

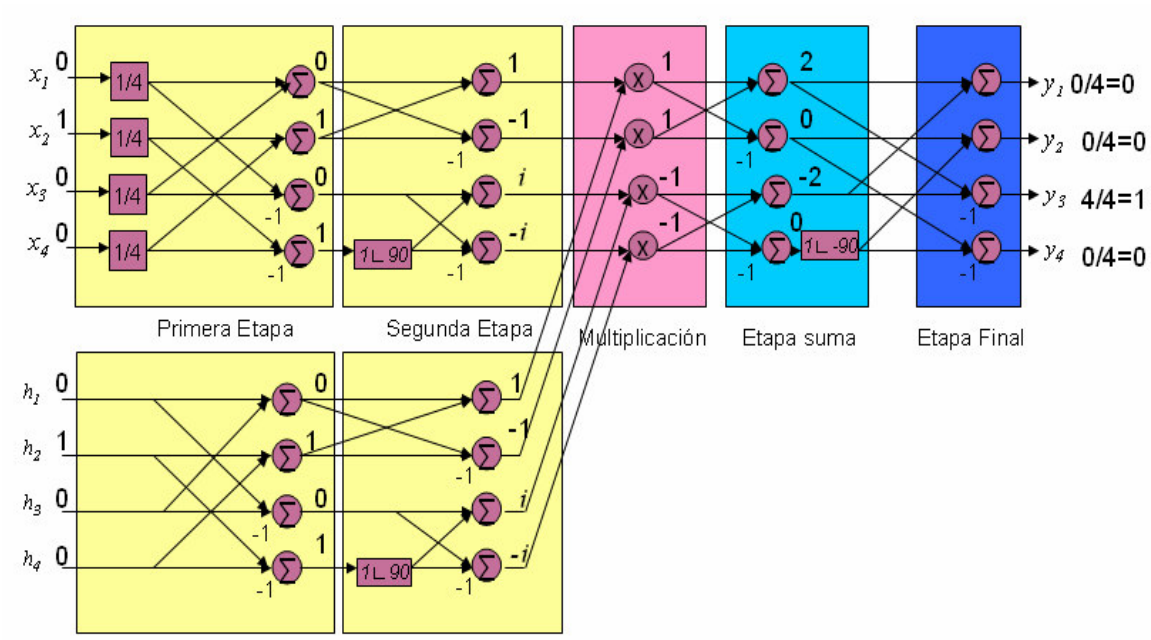


Figura 18. ***Aplicación para convolución cíclica para un orden N=4.***

Con base en la figura 18, podemos realizar una comparación de nuestro algoritmo con aquellas implementaciones de algoritmos de convolución que usan la FFT base-2 presentada en la sección 2.2. Aquí se notan claramente las cinco etapas que posee nuestro algoritmo, nuestras dos primeras etapas son similares al proceso de la transformada de Fourier, una tercera etapa que puede asemejarse a la

multiplicación de las secuencias "transformadas" y dos etapas finales que son similares al proceso de transformada inversa. Las etapas de transformación (las dos primeras y las dos últimas) de éste algoritmo pueden ser obtenidas mediante un enfoque similar al descrito en la sección 2.2 para la DFT en base-2. Nuestro algoritmo presenta S etapas de multiplicación por las raíces del polinomio $z^N - 1$ y S etapas que son similares al proceso de transformada inversa mostrados en 2.2. Para éste primer caso las etapas son $S = 2$ para la multiplicación por las raíces $z^4 - 1$ $[(z-1)(z+1)(z-i)(z+i)]$. Sin embargo, el hecho más importante radica en que solo necesitamos una etapa de multiplicación de tamaño $N = 4$.

Resulta sencillo observar en la figura 18 que el número total de multiplicaciones son solo 4 y que el número total de sumas son 24.

Una vez más, aumentaremos el orden de las secuencias a la próxima potencia de 2, es decir, $N=8$, para confirmar las apreciaciones que hemos venido realizando para $N=2$ y $N=4$.

$$\mathbf{y} = \left[\begin{array}{cccc|cccc} x_1 & x_8 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 \\ x_2 & x_1 & x_8 & x_7 & x_6 & x_5 & x_4 & x_3 \\ x_3 & x_2 & x_1 & x_8 & x_7 & x_6 & x_5 & x_4 \\ x_4 & x_3 & x_2 & x_1 & x_8 & x_7 & x_6 & x_5 \\ \hline x_5 & x_4 & x_3 & x_2 & x_1 & x_8 & x_7 & x_6 \\ x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_8 & x_7 \\ x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_8 \\ x_8 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 \end{array}\right] \left[\begin{array}{c} h_1 \\ h_2 \\ h_3 \\ h_4 \\ \hline h_5 \\ h_6 \\ h_7 \\ h_8 \end{array}\right] \qquad \text{III-11}$$

Podemos representar a la matriz circulante $\mathbf{X}$ y al vector $\mathbf{h}$ como:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X_1} & \mathbf{X_2} \\ \mathbf{X_2} & \mathbf{X_1} \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} \mathbf{h_1} \\ \mathbf{h_2} \end{bmatrix},$$
$$\mathbf{y} = \mathbf{Xh} \qquad \text{III-12}$$
$$\mathbf{y} = \begin{bmatrix} \mathbf{X_1 h_1} + \mathbf{X_2 h_2} \\ \mathbf{X_1 h_2} + \mathbf{X_2 h_1} \end{bmatrix}$$

Nuevamente el mismo algoritmo presentado en III-2 se usará para resolver la anterior multiplicación, es decir:

$$\mathbf{y}=\begin{bmatrix}\mathbf{y_1}\\ \mathbf{y_2}\end{bmatrix}=\begin{bmatrix}\mathbf{X_1} & \mathbf{X_2}\\ \mathbf{X_2} & \mathbf{X_1}\end{bmatrix}.\begin{bmatrix}\mathbf{h_1}\\ \mathbf{h_2}\end{bmatrix}=\begin{bmatrix}\mathbf{m_1}+\mathbf{m_2}\\ \mathbf{m_1}-\mathbf{m_2}\end{bmatrix},$$

donde:

$$\mathbf{m_1}=\frac{1}{2}(\mathbf{X_1}+\mathbf{X_2})(\mathbf{h_1}+\mathbf{h_2}),\ y \qquad \text{III-13}$$

$$\mathbf{m_2}=\frac{1}{2}(\mathbf{X_1}-\mathbf{X_2})(\mathbf{h_1}-\mathbf{h_2})$$

Ahora bien, los vectores $\mathbf{m_1}$ y $\mathbf{m_2}$ se encuentran realizando las siguientes computaciones:

Para $\mathbf{m_1}$ llamaremos

$$\begin{bmatrix}x'_1 & x'_4 & x'_3 & x'_2\\ x'_2 & x'_1 & x'_4 & x'_3\\ x'_3 & x'_2 & x'_1 & x'_4\\ x'_4 & x'_3 & x'_2 & x'_1\end{bmatrix}=\begin{bmatrix}x_1+x_5 & x_8+x_4 & x_7+x_3 & x_6+x_2\\ x_2+x_6 & x_1+x_5 & x_8+x_4 & x_7+x_3\\ x_3+x_7 & x_2+x_6 & x_1+x_5 & x_8+x_4\\ x_4+x_8 & x_3+x_7 & x_2+x_6 & x_1+x_5\end{bmatrix}$$

$$\begin{bmatrix}h'_1\\ h'_2\\ h'_3\\ h'_4\end{bmatrix}=\begin{bmatrix}h_1+h_5\\ h_2+h_6\\ h_3+h_7\\ h_4+h_8\end{bmatrix}, \text{ tenemos } 8 \text{ sumas.} \qquad \text{III-14}$$

Ahora para la multiplicación para obtener $\mathbf{m_1}$

$$\mathbf{m_1}=\frac{1}{2}\left[\begin{array}{cc|cc}x'_1 & x'_4 & x'_3 & x'_2\\ x'_2 & x'_1 & x'_4 & x'_3\\ \hline x'_3 & x'_2 & x'_1 & x'_4\\ x'_4 & x'_3 & x'_2 & x'_1\end{array}\right].\begin{bmatrix}h'_1\\ h'_2\\ h'_3\\ h'_4\end{bmatrix}$$

Tiene las mismas propiedades que III-3, por lo tanto puede ser realizado usando 4 multiplicaciones y 24 sumas.

Para $\mathbf{m_2}$ llamaremos

$$\begin{bmatrix} x'_1 & -x'_4 & -x'_3 & -x'_2 \\ x'_2 & x'_1 & -x'_4 & -x'_3 \\ x'_3 & x'_2 & x'_1 & -x'_4 \\ x'_4 & x'_3 & x'_2 & x'_1 \end{bmatrix} = \begin{bmatrix} x_1 - x_5 & x_8 - x_4 & x_7 - x_3 & x_6 - x_2 \\ x_2 - x_6 & x_1 - x_5 & x_8 - x_4 & x_7 - x_3 \\ x_3 - x_7 & x_2 - x_6 & x_1 - x_5 & x_8 - x_4 \\ x_4 - x_8 & x_3 - x_7 & x_2 - x_6 & x_1 - x_5 \end{bmatrix}$$

$$\begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \\ h'_4 \end{bmatrix} = \begin{bmatrix} h_1 - h_5 \\ h_2 - h_6 \\ h_3 - h_7 \\ h_4 - h_8 \end{bmatrix}, \text{tenemos } 8 \text{ sumas.} \qquad \text{III-15}$$

Ahora para la multiplicación para obtener $\mathbf{m_2}$

$$\mathbf{m_2} = \frac{1}{2} \left[\begin{array}{cc|cc} x'_1 & -x'_4 & -x'_3 & -x'_2 \\ x'_2 & x'_1 & -x'_4 & -x'_3 \\ \hline x'_3 & x'_2 & x'_1 & -x'_4 \\ x'_4 & x'_3 & x'_2 & x'_1 \end{array}\right] \cdot \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \\ h'_4 \end{bmatrix}$$

Al realizar la división propuesta para la matriz anterior, se puede observar una característica similar a III-8, por lo tanto, puede utilizarse el siguiente algoritmo para obtener su resultado:

$$\mathbf{m_2} = \frac{1}{2} \begin{bmatrix} \mathbf{x'_3} & -\mathbf{x'_4} \\ \mathbf{x'_4} & \mathbf{x'_3} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{h'_3} \\ \mathbf{h'_4} \end{bmatrix}$$

$$\mathbf{m_2} = \frac{1}{2} \begin{bmatrix} (r_1 + r_2) \\ (r_1 - r_2)/i \end{bmatrix}, \text{ d}onde \qquad \text{III-16}$$

$$r_1 = \frac{1}{2}(\mathbf{x'_3} + i\mathbf{x'_4})(\mathbf{h'_3} + i\mathbf{h'_4})$$

$$r_2 = \frac{1}{2}(\mathbf{x'_3} - i\mathbf{x'_4})(\mathbf{h'_3} - i\mathbf{h'_4})$$

Donde $i = \sqrt{-1}$ y $\mathbf{x'_3} = \begin{bmatrix} x'_1 & -x'_4 \\ x'_2 & x'_1 \end{bmatrix}$, $\mathbf{x'_4} = \begin{bmatrix} x'_3 & x'_2 \\ x'_4 & x'_3 \end{bmatrix}$, $\mathbf{h'_3} = \begin{bmatrix} h'_1 \\ h'_2 \end{bmatrix}$, $\mathbf{h'_4} = \begin{bmatrix} h'_3 \\ h'_4 \end{bmatrix}$

Por lo tanto, las sumas y restas para hallar r_1 y r_2 son:

$$
\begin{aligned}
(\mathbf{x}_3' + i\mathbf{x}_4') &= \begin{bmatrix} x''_1 & ix''_2 \\ x''_2 & x''_1 \end{bmatrix} = \begin{bmatrix} x'_1 - ix'_3 & -x'_4 - ix'_2 \\ x'_2 - ix'_4 & x'_1 - ix'_3 \end{bmatrix} \\
(\mathbf{x}_3' - i\mathbf{x}_4') &= \begin{bmatrix} x''_3 & -ix''_4 \\ x''_4 & x''_3 \end{bmatrix} = \begin{bmatrix} x'_1 + ix'_3 & -x'_4 + ix'_2 \\ x'_2 + ix'_4 & x'_1 + ix'_3 \end{bmatrix} \\
(\mathbf{h}_3' + \mathbf{i}\mathbf{h}_4') &= \begin{bmatrix} h''_1 \\ h''_2 \end{bmatrix} = \begin{bmatrix} h'_1 + ih'_3 \\ h'_2 + ih'_4 \end{bmatrix} \\
(\mathbf{h}_3' - \mathbf{i}\mathbf{h}_4') &= \begin{bmatrix} h''_3 \\ h''_4 \end{bmatrix} = \begin{bmatrix} h'_1 - ih'_3 \\ h'_2 - ih'_4 \end{bmatrix}
\end{aligned}
\qquad \text{III-17}
$$

Para realizar entonces la multiplicación para encontrar r_1 tenemos:

$$
\begin{aligned}
r_1 &= \frac{1}{2}(\mathbf{x}_3' + i\mathbf{x}_4')(\mathbf{h}_3' + \mathbf{i}\mathbf{h}_4') \\
r_1 &= \frac{1}{2}\begin{bmatrix} x''_1 & ix''_2 \\ x''_2 & x''_1 \end{bmatrix} \cdot \begin{bmatrix} h''_1 \\ h''_2 \end{bmatrix}
\end{aligned}
\qquad \text{III-18}
$$

La cual deberá realizarse por medio de un nuevo algoritmo, de la siguiente forma:

$$
r_1 = \frac{1}{2}\begin{bmatrix} x''_1 & ix''_2 \\ x''_2 & x''_1 \end{bmatrix} \cdot \begin{bmatrix} h''_1 \\ h''_2 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} s_1 + s_2 \\ (s_1 - s_2)/\sqrt{i} \end{bmatrix}
$$

donde:

$$
\begin{aligned}
s_1 &= \frac{1}{2}(x''_1 + \sqrt{i}x''_2)(h''_1 + \sqrt{i}h''_2) \\
s_2 &= \frac{1}{2}(x''_1 - \sqrt{i}x''_2)(h''_1 - \sqrt{i}h''_2)
\end{aligned}
\qquad \text{III-19}
$$

Este algoritmo es nuevo, y los que corresponden a las subsiguientes raíces de la unidad, forman parte de los aportes fundamentales del presente trabajo.

Para realizar la multiplicación para encontrar r_2 tenemos:

$$
\begin{aligned}
r_2 &= \frac{1}{2}(\mathbf{x}_3' - i\mathbf{x}_4')(\mathbf{h}_3' - \mathbf{i}\mathbf{h}_4') \\
r_2 &= \frac{1}{2}\begin{bmatrix} x''_3 & -ix''_4 \\ x''_4 & x''_3 \end{bmatrix} \cdot \begin{bmatrix} h''_3 \\ h''_4 \end{bmatrix}
\end{aligned}
\qquad \text{III-20}
$$

La cual deberá realizarse por medio de otro algoritmo nuevo, de la siguiente forma:

$$r_2 = \frac{1}{2}\begin{bmatrix} x''_3 & -ix''_4 \\ x''_4 & x''_3 \end{bmatrix}\cdot\begin{bmatrix} h''_3 \\ h''_4 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} s_3 + s_4 \\ (s_3 - s_4)/\sqrt{-i} \end{bmatrix}$$

donde:

$$s_3 = \frac{1}{2}(x''_3 + \sqrt{-i}\,x''_4)(h''_3 + \sqrt{-i}\,h''_4) \qquad \text{III-21}$$

$$s_4 = \frac{1}{2}(x''_3 - \sqrt{-i}\,x'_4)(h'_3 - \sqrt{-i}\,h''_4)$$

Lo único que resta por hacer ahora es agrupar los resultados y realizar las sumas y restas correspondientes de los resultados parciales obtenidos para así hallar el resultado general.

Dejaremos como ejercicio algebraico para el lector estas operaciones, y nos concentraremos en los resultados obtenidos para la convolución de secuencias de orden *N=8:*

1) Para el caso de *N=8*, tenemos *(3)(S) =(3)(3)=9* etapas de multiplicación por las raíces y por la constante 1/8.

2) *(3)(N)(log2(N))* sumas. Para el caso de *N=8*, tenemos *(3)(8)(3)=72* sumas en total.

3) 1 etapa de multiplicación de orden *N=8* por los elementos de las secuencias que estan siendo procesadas.

Estos resultados pueden observarse con mejor claridad en la aplicación presentada en las figuras 19, 20 y 21 las cuales presentan además, un ejemplo práctico de cómo es efectuado el flujo de la data, para el proceso de convolución de las siguientes dos secuencias:

$$x = [1,0,10,1,0,0,1]'$$

$$h = [0,1,0,1,1,0,1,1]'$$

$$y = x * h = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 2 \\ 4 \\ 1 \\ 3 \\ 3 \\ 2 \end{bmatrix} \qquad \text{III-22}$$

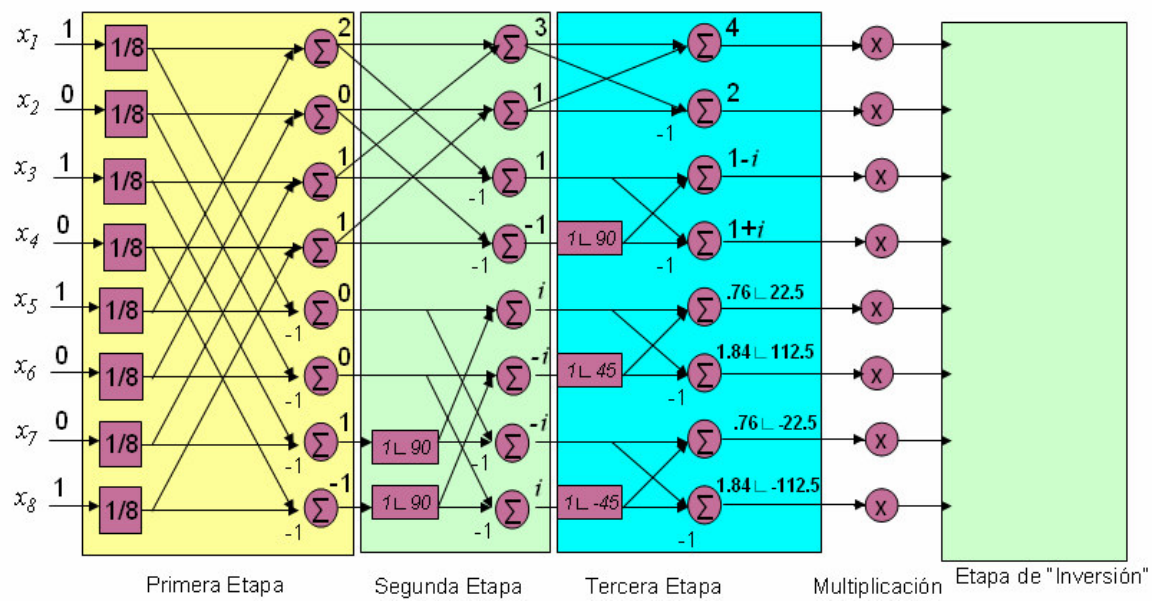


Figura 19. ***Aplicación para convolución cíclica para un orden N=8 (secuencia x).***

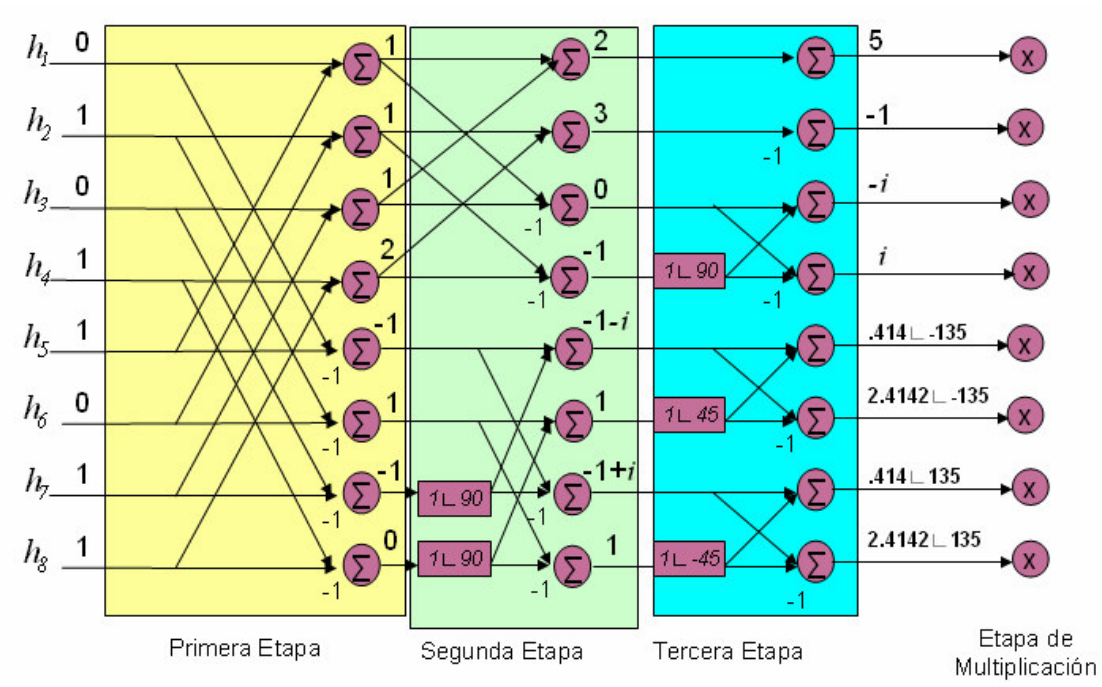


Figura 20. ***Aplicación para convolución cíclica para un orden N=8 (secuencia h).***

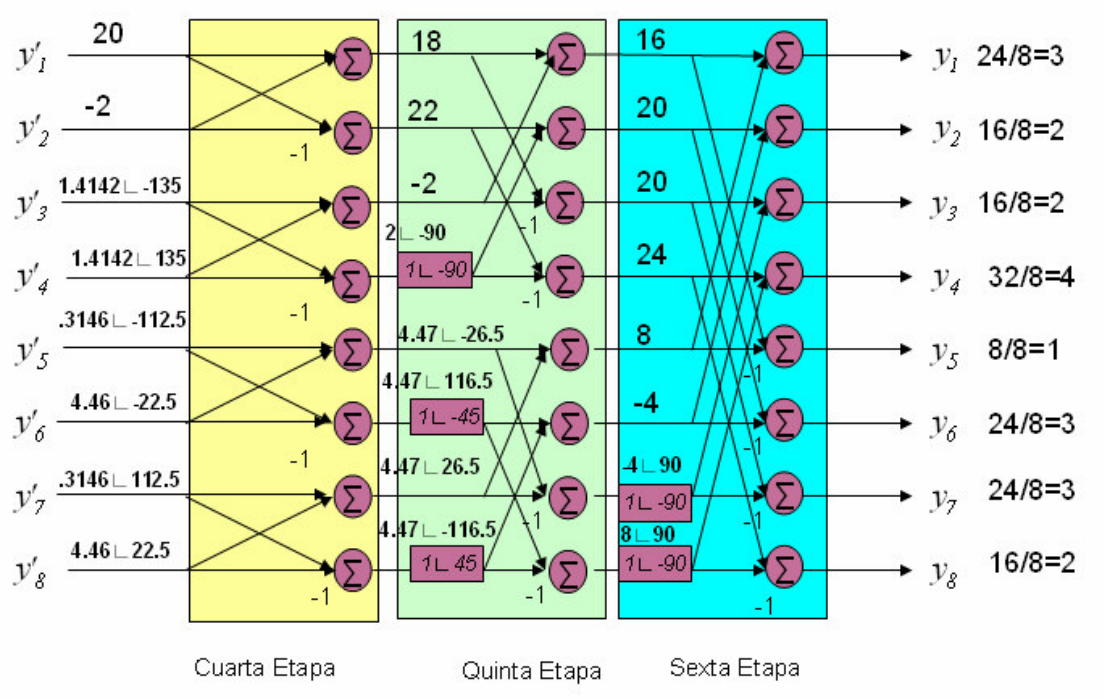


Figura 21. ***Aplicación para convolución cíclica para un orden N=8 (etapa de "inversión")***

En éste punto también podemos notar el procedimiento recursivo que viene ocurriendo, primero la data es multiplicada por las raíces *1* y *-1*, y luego pasan a una etapa en donde se multiplican por las raíces *i* y *–i,* y por último a una etapa de multiplicación por las constantes $\sqrt{i}$ y $\sqrt{-i}$, para luego realizar el proceso inverso, es decir, el proceso de división por las mismas constantes. Si la data a convolucionar fuera aún mayor, una estructura similar a la de la siguiente figura sería utilizada para la aplicación de las raíces:

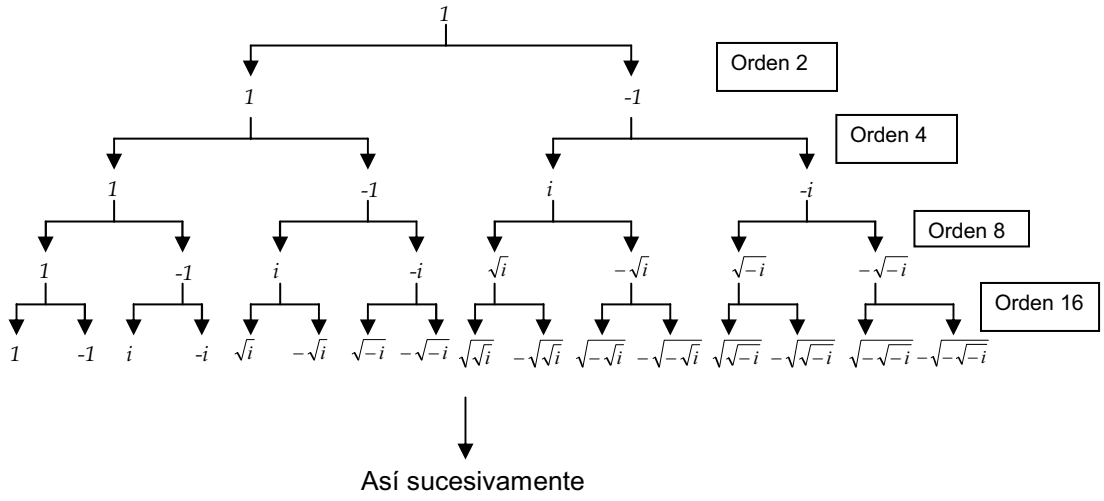


Figura 22. ***Aplicación de las raices para convolución cíclica hasta un orden N=16.***

Un proceso muy similar al descrito con anterioridad para *N=2, N=4* y *N=8* es empleado para encontrar las raíces del polinomio $z^N - 1$, para resolver las convoluciones de orden $N = 2^s$, por lo tanto no tiene sentido seguir con nuestra descripción, lo que haremos será representar estas operaciones mediante un algoritmo general para toda $N = 2^s$.

Para comprender de una mejor manera nuestro algoritmo y observar mejor sus fundamentos matemáticos, es útil observar el problema de computar la convolución $\mathbf{y} = \mathbf{x} * \mathbf{h}$ como un problema de multiplicación de dos polinomios.

Sean $R(u) = \sum_{i=0}^{m} x_i u^i$ y $S(u) = \sum_{j=0}^{n} h_j u^j$ dos polinomios en u, con los coeficientes $x_i's$ y $h_j's$ respectivamente. El polinomio $Q(u) = R(u) \cdot S(u)$ puede escribirse como $Q(u) = \sum_{k=0}^{m+n} y_k u^k$ donde los coeficientes de $Q(u)$ son las entradas de $\mathbf{y} = \mathbf{x} * \mathbf{h}$. Por ésta razón, el problema de la convolución puede ser visto como el de la computación de los coeficientes del polinomio $Q(u)$ de grado $(m+n)$, de los coeficientes del polinomio $R(u)$ de grado m y de los coeficientes del polinomio $S(u)$ de grado n.

El siguiente ejemplo presenta una comparación entre los procedimientos para obtener algoritmos mínimos para la convolución aperiódica usando el Teorema del Residuo Chino y nuestro nuevo procedimiento usando la convolución cíclica y las raíces del polinomio $u^N - 1$ como herramientas.

Consideremos la multiplicación $y = x \cdot h$ de los siguientes dos polinomios:

$x(u) = x_0 + x_1 u$

$h(u) = h_0 + h_1 u + h_2 u^2$

Los elementos deseados del polinomio resultante y son:

$y(u) = x_0 h_0 + (x_0 h_1 + x_1 h_0)u + (x_0 h_2 + x_1 h_1)u^2 + (x_1 h_2)u^3$

Como podemos observar, la multiplicación directa es realizada por medio de 6 multiplicaciones y 2 sumas. Ahora, seguiremos el teorema de Winograd que nos dice que el mínimo algoritmo para ésta multiplicación de polinomios es obtenido usando el teorema del residuo Chino, mediante la división por los polinomios $(u-1),(u+1),(u),(u-\infty)$:

Realizando las divisiones correspondientes a los dos polinomios entre $u-1$ tenemos:

$$\begin{array}{r|l} x_0 + x_1 u & u - 1 \\ x_1 - x_1 u & x_1 \\ \hline x_0 + x_1 \text{ residuo} & \end{array} \qquad \begin{array}{r|l} h_0 + h_1 u + h_2 u^2 & u - 1 \\ h_2 u - h_2 u^2 & h_2 u + (h_1 + h_2) \\ \hline h_0 + (h_1 + h_2)u & \\ h_1 + h_2 - (h_1 + h_2)u & \\ \hline h_0 + h_1 + h_2 \text{ residuo} & \end{array}$$

Siguiendo un procedimiento similar para los demás polinomios $(u+1),(u),(u-\infty)$, tenemos los siguientes residuos:

$(x_0 - x_1)$ y $(h_0 - h_1 + h_2)$ para $(u+1)$

(x_0) y (h_0) para (u)

(x_1) y (h_2) para $(u-\infty)$

Ahora, multiplicaremos los residuos de las divisiones por cada polinomio para obtener las siguientes relaciones:

$m_1 = (x_0)(h_0)$

$m_2 = (x_0 + x_1)(h_0 + h_1 + h_2)/2$

$m_3 = (x_0 - x_1)(h_0 - h_1 + h_2)/2$

$m_4 = (x_1)(h_2)$

Mediante una combinación lineal de los valores obtenidos podemos expresar los resultados de nuestra multiplicación de polinomios como:

$$y_0 = m_1 = x_0 h_0$$
$$y_1 = m_2 - m_3 - m_4 = (x_0 h_1 + x_1 h_0)$$
$$y_2 = -m_1 + m_2 + m_3 = (x_0 h_2 + x_1 h_1)$$
$$y_3 = m_4 = (x_1 h_2)$$

Podemos observar que solo necesitamos 4 multiplicaciones en vez de las seis obtenidas por la multiplicación directa.

Estas mismas multiplicaciones pueden ser obtenidas mediante el uso de la convolución cíclica de la siguiente manera:
Hagamos $Q = N+M-1 = 2+3-1 = 4$, el tamaño de nuestras nuevas sucesiones:

$$x(u) = x_o + x_1 u + 0u^2 + 0u^3$$
$$h(u) = h_o + h_1 u + h_2 u^2 + 0u^3$$

Demostraremos mediante éste ejemplo la clave presentada en 2.6, es decir, que los algoritmos con mínima complejidad multiplicativa para la convolución cíclica son fuertemente dependiente del polinomio mónico $u^N - 1$, en éste caso $u^4 - 1$:

Realizando las correspondientes divisiones de nuestros dos polinomios $x(u), h(u)$ entre las raíces $(u-1),(u+1),(u-i),(u+i)$, obtenemos los siguientes residuos:

$(x_0 + x_1)$ y $(h_0 + h_1 + h_2)$ para $(u-1)$
$(x_0 - x_1)$ y $(h_0 - h_1 + h_2)$ para $(u+1)$
$(x_0 + ix_1)$ y $(h_0 + ih_1 - h_2)$ para $(u-i)$
$(x_0 - ix_1)$ y $(h_0 - ih_1 + h_2)$ para $(u+i)$

Ahora, multiplicaremos los residuos de las divisiones por cada polinomio para obtener las siguientes relaciones:

$$m_1 = (x_0 + x_1)(h_0 + h_1 + h_2)/2$$
$$m_2 = (x_0 - x_1)(h_0 - h_1 + h_2)/2$$
$$m_3 = (x_0 + ix_1)(h_0 + ih_1 - h_2)/2$$
$$m_4 = (x_0 - ix_1)(h_0 - ih_1 + h_2)/2$$

Mediante una combinación lineal de los valores obtenidos podemos expresar los resultados de nuestra multiplicación de polinomios como:

$$y_0 = (m_1 + m_2 + m_3 + m_4)/2 = x_0 h_0$$
$$y_1 = ((m_1 - m_2) + (m_3 - m_4)/i)/2 = (x_0 h_1 + x_1 h_0)$$
$$y_2 = ((m_1 + m_2) - (m_3 + m_4))/2 = (x_0 h_2 + x_1 h_1)$$
$$y_3 = ((m_1 - m_2) - (m_3 - m_4)/i)/2 = (x_1 h_2)$$

Nuevamente podemos observar que solo necesitamos 4 multiplicaciones para la multiplicación de los polinomios usando la convolución cíclica como herramienta.

Es necesario hacer notar que éste último enfoque es muy similar al que usamos para la expresión matricial de la convolución cíclica. Las raíces utilizadas son las mismas, y en esencia las relaciones y combinaciones son idénticas. La ventaja de la derivación presentada con la representación matricial de la convolución radica en que no se hace necesario la obtención previa de los residuos de los polinomios, ya que las constantes a manejar se obtienen del polinomio mónico $u^N - 1$, y por lo tanto, la complejidad del algoritmo se reduce.

3.1 Algoritmo General.

En ésta sección describiremos la manera como va a actuar nuestro algoritmo general para la convolución cíclica, para lo cual, seguiremos un planteamiento muy similar al expuesto en la sección 2.2 para la transformada de Fourier base-2.

Expresar el algoritmo propuesto es relativamente simple. Supongamos que queremos realizar la convolución de dos secuencias digamos $x[n]$ y $h[n]$, ambas de longitud $N = 2^s$. En aras de la originalidad vamos a hallar en principio cual es el conjunto de nuestro campo de constantes, sabemos que éste campo depende únicamente del orden (N) de las secuencias, además, no necesitamos todos los valores de dichas constantes, puesto que la mitad son positivas y las otras el valor negativo de las anteriores, entonces una vez halladas las constantes, nos concentraremos en guardar solo las positivas.

Un algoritmo creado en Matlab® para realizar dicha operación, el cual usa como base la derivación presentada en la figura 22, es el siguiente:

```
N=max(size(x)); %Longitud de las secuencias
stg=log2(N); %Etapas del algoritmo
s=1;
c=[1];                                                    III-23
k=1;
while max(size(c))<N  %Creacion de campo de constantes de acuerdo
   cn=[];             % a la longitud de las secuencias
```

```
   k=1;
   for i= 1:2:2^s
      cn(i)=sqrt(c(k));
      cn(i+1)=-sqrt(c(k));
      if i==2^s-1
      c=cn;
      s=s+1;
      else
      k=k+1;
      end
   end
end
c=[];
for i=1:N/2  % Almacenamiento de las constantes positivas unicamente
   c(i)=cn(2*i-1);
end
```

El vector de constantes ésta organizado desde la constante 1 en adelante, una vez tenido este vector, que tendrá un tamaño igual a la mitad de la longitud de las secuencias, procedemos a realizar las etapas de "transformación" que consisten en la realización de las siguientes instrucciones:

```
s=1; %primera etapa
k=1;
inc=0;
j=1;
ti=0;
while s<=stg
   for i=1: N/2^s; %
      xn(i+inc)=x(i+inc)+c(j)*x(i+inc+N/2^s); % Se observa el computo
de x y h en paralelo
      xn(i+N/2^s+inc)=  x(i+inc)-c(j)*x(i+inc+N/2^s);  %Multiplicacion
por las constantes respectivas
      hn(i+inc)=h(i+inc)+c(j)*h(i+inc+N/2^s);
      hn(i+N/2^s+inc)= h(i+inc)-c(j)*h(i+inc+N/2^s);
      temp=inc;
    end
    if (i+temp+N/2^s)==N
      s=s+1;
      x=xn;
      h=hn;
      j=1;
      inc=0;
    else
```

```
      inc = inc+N/2^(s-1);
      j=j+1;
   end
end
```

Lo que gráficamente corresponde a las primeras etapas, antes del proceso de multiplicación de las figuras 17, 18, 19 y 20. Nótese que tanto para $x[n]$ como para $h[n]$ se realizan los mismos cómputos. En éste algoritmo general no hemos tomado en cuenta aún la multiplicación por la constante $\frac{1}{N}$ (pudo haber sido aplicada a $x[n]$ ó a $h[n]$), la cual preferiremos utilizar al final del proceso.

En éste punto el algoritmo creado se basa en lo que ocurre en las figuras 17, 18, 19 y 20 aunque con el objeto de guiarnos para la formulación del algoritmo, también se realizaron diagramas de aplicación para $N = 16$, los cuales omitimos en el presente trabajo por razones de espacio.

Al realizar las aplicaciones correspondientes a los diagramas de flujo de señal, observamos que se construía una estructura muy regular y relativamente sencilla de implementar en forma de algoritmo, en particular observamos para cada etapa cuales eran las raíces que se aplicaban, cuantas de ellas se aplicaban en cada subrutina y sobre todo, los lugares en que se aplicaban. De dicho estudio, llegamos a la formulación de la siguiente subrutina que nos deja el proceso justo antes de la etapa de multiplicación de las dos secuencias que están siendo convolucionadas (etapa de multiplicación en las figuras 17, 18, 19 y 20).

```
s=1; %primera etapa
k=1;
inc=0;
j=1;
ti=0;
while s<=stg
   for i=1: N/2^s; %
      xn(i+inc)=x(i+inc)+c(j)*x(i+inc+N/2^s); % Se observa el computo de
x y h en paralelo                                                    III-25
      xn(i+N/2^s+inc)= x(i+inc)-c(j)*x(i+inc+N/2^s); %Multiplicación por
las constantes respectivas
      hn(i+inc)=h(i+inc)+c(j)*h(i+inc+N/2^s);
      hn(i+N/2^s+inc)= h(i+inc)-c(j)*h(i+inc+N/2^s);
      temp=inc;
   end
   if (i+temp+N/2^s)==N
      s=s+1;
```

```
      x=xn;
      h=hn;
      j=1;
      inc=0;
   else
      inc = inc+N/2^(s-1);
      j=j+1;
   end
end
```

El siguiente proceso consiste en la etapa de multiplicación punto a punto de las dos secuencias (vectores resultantes) y en Matlab® es posible realizarlo con una sola instrucción de la siguiente manera:

```
yn=x.*h;% Etapa de multiplicación de las dos secuencias “transformadas"
```
III-26

Ahora solo nos queda seguir la ruta inversa para llegar a la solución final de nuestro algoritmo, la cual es muy parecida a III-25, con la diferencia que para III-25, primero se aplicaban las raíces y luego se sumaba, ahora primero se suma y luego se dividirá por las raíce, pero nuevamente con un alto grado de regularidad:

```
y=yn;
temp=0;
inc=0;
s=s-1;
k=0;
while k~=N
   for i=1: N/2^s; % Primera etapa se suman y restan los elementos sucesivos
de la secuencia.
      yn(i+inc)=y(i+inc)+y(i+inc+N/2^s);
      yn(i+N/2^s+inc)= y(i+inc)-y(i+inc+N/2^s);
      temp=inc;
   end
   if (i+temp+N/2^s)==N
      y=yn;
      inc=0;
      k=i+temp+N/2^s;
   else
      inc = inc+N/2^(s-1);
   end
end
y=yn;
```
III-27

```
s=s-1;
r=1;
while s>0 %En las subsiguientes etapas se necesitan dividir los resultados
obtenidos por las constantes del campo
   nra=2^(s); % Numero de raíces a aplicar en la etapa correspondiente
   nacr=2^(r-1); % Numero de veces que cada raíz es aplicada en la etapa
   lapr=2^(stg-s)-nacr+1; %lugar de aplicación de la primera raíz a aplicar
   inc=2^(stg-s); %Incremento en los lugares de aplicación de las raíces
   temp=lapr;
   for i=1:nra;
         k=temp;
         if k<=N
            for j=1:nacr;
               yn(k+j-1)=yn(k+j-1)*c(i)^-1; % División por las constantes
respectivas
            end
         end
         temp=temp+inc;
    end
   y=yn;
   r=r+1;
   inc2=0;
   p=0;
   while p~=N
      for i=1: N/2^s; %
          yn(i+inc2)=y(i+inc2)+y(i+inc2+N/2^s); % sumas y restas
correspondientes
          yn(i+N/2^s+inc2)= y(i+inc2)-y(i+inc2+N/2^s);
          temp=inc2;
      end
      if (i+temp+N/2^s)==N
         s=s-1;
         y=yn;
         inc=0;
      else
         inc2 = inc2+N/2^(s-1);
      end
      p=i+temp+N/2^(s+1);
   end
 end
yf=1/N*yn; %resultado final de la convolución
end
```

Como parte final solo nos queda dividir el resultado obtenido entre N con lo que obtenemos la convolución deseada:

yn=yn/N; % se divide entre N el vector para obtener el valor de la convolucion III-28

En el apéndice A.2 se presentará una copia completa del algoritmo en Matlab®.

La tabla siguiente presenta un paralelo de la complejidad multiplicativa del producto directo y la metodología propuesta en éste trabajo:

Orden $N = 2^s$	**Usual Multipl.** N^2	**Usual Sum.** $N(N-1)$	**New Multipl.** N	**New Sum.** $3N * log_2(N)$
2	4	2	2	(3*2*1) =6
4	16	12	4	(3*4*2) =24
8	64	56	8	(3*8*3)=72
16	**256**	**240**	**16**	**(3*16*4)=192**
32	1024	992	32	480
64	4096	4032	64	1152
128	16384	16256	128	2688
256	65536	65280	256	6144

Tabla 2: Paralelo entre la complejidad aritmética del producto directo y la nueva metodología.

1- La fila con sombreado en la tabla indica el orden en que las sumas del nuevo algoritmo comienzan a ser menores que en la multiplicación directa.

Una comparación entre las tablas 1 y 2 muestra como los resultados del mínimo número de multiplicaciones son en apariencia mejorados con nuestro algoritmo con respecto a la FFT-2 y la FFT-4, sin embargo, al analizar los diagramas de flujo de señal de la figura 19 (nuestro algoritmo) y el de la figura 23 (FFT-2 orden *N=8),* observamos que las operaciones por las raíces de la unidad son esencialmente las mismas, y la reducción es solo debida a que nosotros no consideramos dichas multiplicaciones por encontrarse dentro de nuestro campo de constantes, sin embargo, podemos decir a favor de nuestros algoritmos, que los mismos no necesitan la etapa de "bit reversal" o cambio de orden de la secuencia para ser ejecutados organizadamente, lo cual puede redundar en un ahorro considerable de memoria y tiempo de proceso.

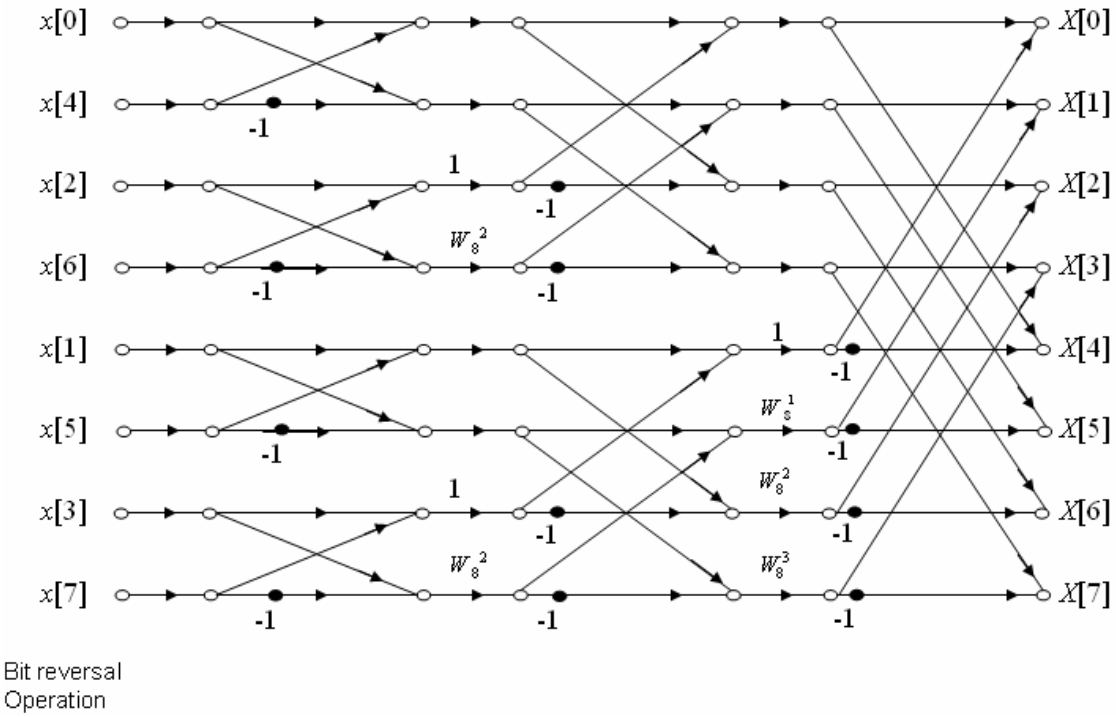


Figura 23. ***Diagrama de flujo de señal de FFT-2 para un orden de secuencia N=16.***

3.2 Complejidad en notación Big-O

En la presente sección realizaremos un análisis teórico del número de operaciones para la ejecución de nuestros algoritmos, tomando como herramienta para este problema, la notación *Big-O,* la cual se define como una medida teórica del tiempo de ejecución de un algoritmo, usualmente el tiempo o memoria necesitada, dado un problema de tamaño *N*, lo cual es usualmente el número de ítems.

Informalmente, alguna ecuación $f(n) = O(g(n))$, significa que las operaciones necesarias para realizar el algoritmo descrito por $f(n)$, son menores que alguna constante múltiplo de $g(n)$ *(NIST: Nacional Institute of Standar and Technology).*

Analizaremos primero el número de multiplicaciones necesarias para ejecutar nuestro algoritmo, por tal motivo, incluiremos la multiplicación por nuestras constantes, ya que las mismas hacen parte de la solución final requerida.

Un análisis de las gráficas de flujo de señal de las figuras 19, 20 y 21, muestra que el número de multiplicaciones por las constantes y los productos generales de la etapa de multiplicación de cada gráfica puede ser caracterizado por la expresión general:

$$mult(n) = n + 3\sum_{i=1}^{s-1}\left(\frac{n}{2^{s-i}} - 1\right)\cdot\left(\frac{n}{2^{i+1}}\right) \qquad \text{III-29}$$

Donde $n = 2^S$, es el tamaño de las secuencias. La ecuación puede expresarse de la siguiente manera:

$$mult(s) = 2^s + 3\sum_{i=1}^{s-1}\left(\frac{2^s}{2^{s-i}} - 1\right)\cdot\left(\frac{2^s}{2^{i+1}}\right) = 2^s + 3\sum_{i=1}^{s-1}(2^i - 1)\cdot(2^{s-1-i})$$
$$mult(s) = 2^s + 2^{s-1}\cdot 3\sum_{i=1}^{s-1}(1 - 2^{-i}) = n\left(1 + \frac{3}{2}\sum_{i=1}^{s-1}(1 - 2^{-i})\right) \qquad \text{III-30}$$

De la ecuación final, podemos asumir que un caso hipotético máximo (que más multiplicaciones conllevaría), podría ocurrir cuando $i = s - 1$, en todas las $s - 1$ sumas, por lo tanto:

$$mult(n) \le n + \frac{3n}{2}(s-1)\frac{(2^{s-1} - 1)}{2^{s-1}} = n + \frac{3n}{2}(s-1)\left(\frac{\frac{n}{2} - 1}{\frac{n}{2}}\right) = n + 3(s-1)\left(\frac{n}{2} - 1\right) \qquad \text{III-31}$$

Ahora, expresando $s = log_2(N)$, podemos obtener que en la notación Big-O, nuestro número de multiplicaciones será:

$$mult(n) = O(n \cdot log(n)) \qquad \text{III-32}$$

Con lo que se tiene entonces un algoritmo bastante eficiente en cuanto a número de multiplicaciones.

Analizaremos ahora el número de sumas, este es un caso más sencillo, puesto que en el diagrama de flujo de señal de las figuras 19, 20 y 21, se observa claramente que el número total de ellas está dado por la expresión:

$$sum(n) = 3 \cdot n \cdot log_2 n \qquad \text{III-33}$$

Entonces, en la notación Big-O, tenemos que nuestro número de sumas es:

$$sum(n) = O(n \cdot log\, n) \qquad \text{III-34}$$

Nuevamente, bastante eficiente en cuanto a número de sumas.

Conclusiones

En el trabajo presentado se estudiaron las características más importantes de las matrices circulantes, se realizó un énfasis especial en el proceso de convolución cíclica que hace uso de éstas matrices y se estudiaron sus relaciones con los polinomios y la multiplicación de los mismos, con el fin de obtener un nuevo algoritmo optimo en el sentido de alcanzar la mínima complejidad multiplicativa expresada en el teorema de Winograd, debido a que nuestro algoritmo general para realizar $CF(Z_N)$,[4] necesita solo N multiplicaciones y $N \log_2 N$ sumas:

Se ha establecido un vínculo novedoso entre la factorización de las matrices circulantes y el proceso de decimación en tiempo de la FFT base-2. Esta factorización obtenida se ha demostrado que conlleva a un algoritmo de convolución en paralelo, que aunque tiene los mismos puntos de computación que la FFT base 2, no necesita realizar los procesos de Bit reversal lo cual constituye un ahorro de computación.

El algoritmo obtenido refleja íntimamente la relación de dependencia de los polinomios mónicos con el campo F de las constantes. Observamos que las raíces del polinomio $(z^N - 1)$ juegan un papel preponderante en el algoritmo propuesto para la convolución cíclica y para los productos de polinomios.

Como pasos lógicos para futuros desarrollos, proponemos analizar el tipo de estructura presentada aquí para resolver el problema de la convolución cíclica en múltiples dimensiones, de acuerdo a la formulación en columna mayor presentada en 1.19, la cual resulta en una multiplicación de una matriz circulante por bloques, con bloques circulantes, por un vector. Si el orden de las secuencias multidimensionales son potencias de 2, un enfoque similar con algunas variantes puede ser aplicado para obtener algoritmos eficientes para la convolución cíclica multidimensional.

Pueden estudiarse también las herramientas del producto Tensor o Kronecker presentada en 1.15 para aplicarlas de acuerdo a los diagramas de flujo de señal obtenidos en la implementación de los algoritmos, para constatar si existe algún otro tipo de formulación que pueda ser implementada con mejor desempeño computacional a la presentada en éste trabajo.

[4] $CF(Z_N)$ es campo complejo orden N

Referencias

[Ammar] G. Ammar and P. Gader, *New decompositions of the inverse of a Toeplitz matriz,* pp 421-428, Signal Processing, Scattering and Operador Theory, and Numerical Methods, Proc. Int. Symp MTNS-89 vol III, Birkhauser, Boston 1990.

[Catalan] Catalan E. *Récherches sur les Déterminants*, pp 534-555, Bulletin de l 'Academie Royale de Belgique, 1846

[Davis] *Philip J. Davis Circulants Matrices.* pp 16-107, John Wiley & Sons New York, New York, 1979

[Gohberg1] I. Gohberg and V. Olshevsky, *Complexity of Multiplication with Vectors for Structured Matrices,* pp163-192 Linear Algebra Appl. 202, 1994

[Gohberg2] I. Gohberg and V. Olshevsky X. *Circulants, Displacements and Decomposition of Matrices,* pp 853-863, Integral Equations and Operator Theory Vol. 15, 1992

[Heideman] M. Heideman, *"Multiplicative Complexity, Convolution, and the DFT".* Springer Verlag, New York 1988.

[Lara] T. Lara, *Matrices Circulantes,* pp 85-102, Divulgaciones Matemáticas vol II, Departamento de Física y Matemáticas NURR-ULA, Trujillo, Venezuela 2001.

[Muir1] Muir T.,.*The Theory of Circulants in the Historical Order of Development up to 1860*, pp 390-398, Proceedings of the Royal Society of Edinburgh, 1911

[Muir2] Muir T.,.*The Theory of Circulants from 1861 to 1880*, pp 136-149 Proceedings of the Royal society of Edinburgh, 1906

[Muir3] Muir T.,.*The Theory of Circulants from 1880 to 1900*, pp 151-179, Proceedings of the Royal society of Edinburgh, 1911

[Muir4] Muir T.,.*The Theory of Circulants from 1900 to 1920*, pp 218-241, Proceedings of the Royal society of Edinburgh, 1923

[Myers] D.G. MYERS, *"Efficient Convolution and Fourier Transform Techniques".* Prentice Hall of Australia 1990.

[Rodriguez1] Rodriguez, D*.; Tensor product algebra as a tool for VLSI implementation of the discrete Fourier transform,* pp 1025 -1028 vol.2, Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on , 14-17 Apr 1991.

[Rodriguez2] M. Teixeira and D. Rodriguez, "*A new method mathematically links fast Fourier transform algorithms with fast cyclic convolution algorithms,"* Proc. 37th Midwest Symposium on Circuit and Systems, Lafayette, Louisiana, August 1994

[Rodriguez3] M. Teixeira and D. Rodriguez*, "A Class of Fast Cyclic Convolution Algorithms Based on Block Pseudocirculants".* IEEE Signal Processing Letters, 5 (2), p 92. May 1995.

[Winograd] S. Winograd, *"Arithmetic Complexity of computations".* Society for Industrial and Applied Mathematics, Philadelphia 1980

Apéndice A

A.1 Formulación de algoritmos de FFT Usando el Producto Kronecker

En el Capítulo 1 se presentaron las siguientes formulas:

$$\tilde{y}[\ell] = \sum_{k=0}^{N-1} \tilde{x}[k] e^{\frac{-j2\pi\ell k}{N}} \;;\; k = 0,1.....N-1, \;\; and\; j = \sqrt{-1} \qquad [A.12]$$

Esta función de DFT se puede expresar:

$$\tilde{y} = (F_S \otimes I_R) T_{N,S} (I_S \otimes F_R) P_{N,S} \tilde{x} \qquad [A.13]$$

La matriz $T_{N,S}$ es una matriz especial llamada *"twiddle factor",* y está definida como:

$$T_{N,S} = diag[I_R, D_{R,N}, ...(D_{R,N})^{S-1}] \qquad [A.14]$$

Donde *diag,* significa diagonal, I_R es la matriz identidad de orden R y $D_{R,N}$ esta definida por:

$$D_{R,N} = diag[1, w_N, ...(w_N)^{R-1}] \qquad [A.15]$$

Para explicar de una mejor forma como llegar al resultado final dado en [A.13] [Tolimieri], usando el producto Kronecker vamos a iniciar con un ejemplo para un vector de longitud $N=8$, para el cual, $R=2$, y $S=4$.

Nuestro vector de entrada es:

$$\tilde{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \qquad [A.16]$$

Vamos a asociar a este vector de entrada un arreglo llamado x (nótese sin el súper índice ~) de la siguiente manera:

$$x = \begin{bmatrix} x_0 & x_4 \\ x_1 & x_5 \\ x_2 & x_6 \\ x_3 & x_7 \end{bmatrix} \qquad [A.17]$$

Y éste último a una matriz:

$$x_1 = x^t = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \end{bmatrix}, \qquad [A.18]$$

Ahora, el vector $\tilde{x_1}$ correspondiente ha x_1, puede ser obtenido por:

$$\tilde{x_1} = P(8,4)\,\tilde{x} \qquad [A.19]$$

Siendo éste;

$$\tilde{x}_1 = \begin{bmatrix} x_0 \\ x_4 \\ x_1 \\ x_5 \\ x_2 \\ x_6 \\ x_3 \\ x_7 \end{bmatrix} \qquad [A.19.1]$$

Debemos tener en cuenta que las permutaciones obtenidas mediante la función $P(8,4)$ corresponden a los subíndices (0,4), (1,5), (2,6), y (3,7).

Ahora, vamos a asociar al vector de salida $\tilde{y}[\ell]$, su correspondiente arreglo de 2x4, como se hizo con $\tilde{x}$, de la siguiente manera:

$$y = \begin{bmatrix} y_0 & y_2 & y_4 & y_6 \\ y_1 & y_3 & y_5 & y_7 \end{bmatrix} \qquad [A.20]$$

Podemos entonces escribir nuestra ecuación [A.12], en términos de los arreglos x_1 e y, e introduciendo un cambio de variables para tener la nueva matriz en términos de variables relacionadas a *R* y *S*:

$$x_1[k_0, k_1] = \tilde{x}[k_1 + 4k_0]; \quad 0 \le k_0 < 2; \quad 0 \le k_1 < 4, \qquad [A.21]$$

Y similarmente:

$$y[\ell_0, \ell_1] = \tilde{y}[\ell_0 + 2\ell_1]; \quad 0 \le \ell_0 < 2; \quad 0 \le \ell_1 < 4 \qquad [A.22]$$

El siguiente paso consiste en expresar $e^{\frac{-j2\pi\ell k}{N}}$, como $w_N^{\ell k}$, éste cambio de variables cambiará la sumatoria sencilla mostrada en [A.12], en una sumatoria doble como se muestra a continuación:

$$y[\ell_0,\ell_1]=\sum_{k1=0}^{3}\sum_{k0=0}^{1}x_1[k_0,k_1]w^{(k_1+4k_0)(\ell_0+2\ell_1)} \qquad [A.23]$$

O en una forma general como:

$$y(\ell_0,\ell_1)=\sum_{k1=0}^{S-1}\sum_{k0=0}^{R-1}x_1(k_0,k_1)w_N^{(k_1+Sk_0)(\ell_0+R\ell_1)} \qquad [A.24]$$

Bien, para nuestro caso particular de *N=8, R=2, S=4,* vamos a definir las variables $v_N = w_N^k = w_8^2 = -i$ y $u_N = w_N^\ell = w_8^4 = -1$, que se obtienen de $e^{\frac{-j2\pi(2)}{8}}$ y $e^{\frac{-j2\pi(4)}{8}}$, respectivamente. Por ésta razón, nuestro término exponencial $w^{(k_1+Sk_0)(\ell_0+R\ell_1)}$, en éste caso particular, puede ser expresado como $w_8^{k_1\ell_0}.w_8^{2k_1\ell_1}.w_8^{4k_0\ell_0}.w_8^{8k_0\ell_1}$ donde $w_8^{8}=1$, trasladando éste último resultado a [A.24] tenemos:

$$y[\ell_0,\ell_1]=\sum_{k1=0}^{3}\left[\sum_{k0=0}^{1}\left(x_1(k_0,k_1)u_8^{(k_0\ell_0)}\right)\right]w_8^{k_1\ell_0}v_8^{k_1\ell_1} \qquad [A.25]$$

Enfocaremos ahora nuestra atención a la expresión dentro de los paréntesis y llamaremos a ésa parte de la ecuación y_1, la cual será considerada en función de $[\ell_0,k_1]$, por lo tanto tenemos:

$$y_1[\ell_0,k_1]=\sum_{k0=0}^{1}\left(x_1(k_0,k_1)u_8^{(k_0\ell_0)}\right) \qquad [A.26]$$

Vamos a expandir los términos de la suma para encontrar los valores de $y_1[\ell_0,k_1]$ de la siguiente manera:

$$y_1[\ell_0,k_1]=x_1(0,k_1)u_8^{(0)}+x_1(1,k_1)u_8^{(\ell_0)} \qquad [A.27]$$

Ahora, para los valores de $0\le\ell_0<2,$ y $0\le k_1<4$, tenemos:

$$\begin{aligned} y_1[0,0]&=x_1[0,0]+x_1[1,0]\\ y_1[0,1]&=x_1[0,1]+x_1[1,1]\\ y_1[0,2]&=x_1[0,2]+x_1[1,2]\\ y_1[0,3]&=x_1[0,3]+x_1[1,3] \end{aligned} \qquad [A.28]$$

$$y_1[1,0] = x_1[0,0] - x_1[1,0]$$
$$y_1[1,1] = x_1[0,1] - x_1[1,1]$$
$$y_1[1,2] = x_1[0,2] - x_1[1,2]$$
$$y_1[1,3] = x_1[0,3] - x_1[1,3]$$

Que en forma matricial puede representarse como:

$$y_1[\ell_0, k_1] = \begin{bmatrix} x_1[0,0]+x_1[1,0] & x_1[0,1]+x_1[1,1] & x_1[0,2]+x_1[1,2] & x_1[0,3]+x_1[1,3] \\ x_1[0,0]-x_1[1,0] & x_1[0,1]-x_1[1,1] & x_1[0,2]-x_1[1,2] & x_1[0,3]-x_1[1,3] \end{bmatrix} \quad [A.29]$$

Recordando el resultado obtenido para el vector $\tilde{x}_1$ (vector de permutaciones de $\tilde{x}$), [A.19.1], podemos obtener los términos de la anterior matriz [A.29] por medio del siguiente producto Kronecker:

$$\tilde{y}_1 = (I_4 \otimes F_2) \times \tilde{x}_1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_4 \\ x_1 \\ x_5 \\ x_2 \\ x_6 \\ x_3 \\ x_7 \end{bmatrix} = \begin{bmatrix} x_0 + x_1 \\ x_0 - x_1 \\ x_2 + x_3 \\ x_2 - x_3 \\ x_4 + x_5 \\ x_4 - x_5 \\ x_6 + x_7 \\ x_6 - x_7 \end{bmatrix} \quad [A.30]$$

De acuerdo a éste último resultado, podemos asociar a la matriz $y_1[\ell_0, k_1]$ de [A.28] al vector $\tilde{y}_1 = (I_4 \otimes F_2)\tilde{x}_1$ de [A.28]. Continuando ahora con la multiplicación de ésta matriz por los valores de $w_8^{k_1 \ell_0}$ en [A.25], definiremos una nueva variable $y_2[\ell_0, k_1]$ de la siguiente manera:

$$y_2[\ell_0, k_1] = y_1[\ell_0, k_1] w_8^{k_1 \ell_0} \quad [A.31]$$

La cual, evaluada, entrega los siguientes resultados:

$$y_2[0,0] = y_1[0,0] \quad [A.32]$$

$$
\begin{aligned}
y_2[0,1] &= y_1[0,1] \\
y_2[0,2] &= y_1[0,2] \\
y_2[0,3] &= y_1[0,3] \\
y_2[1,0] &= y_1[1,0] \\
y_2[1,1] &= y_1[1,1]\, w_8 \\
y_2[1,2] &= y_1[1,2]\, w_8^2 \\
y_2[1,3] &= y_1[1,3] w_8^3
\end{aligned}
$$

Que en forma matricial pueden representarse como:

$$
y_2[\ell_0, k_1] = \begin{bmatrix} y_1[0,0] & y_1[0,1] & y_1[0,2] & y_1[0,3] \\ y_1[1,0] & y_1[1,1] & y_1[1,2] & y_1[1,3] \end{bmatrix} \qquad \text{[A.33]}
$$

Se puede verificar que el arreglo $y_2[\ell_0, k_1]$ está asociado al vector $\tilde{y}_2$, donde:

$$
\tilde{y}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & w_8^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_8^3 \end{bmatrix} \times \begin{bmatrix} y_1[0,0] \\ y_1[1,0] \\ y_1[0,1] \\ y_1[1,1] \\ y_1[2,0] \\ y_1[1,2] \\ y_1[0,3] \\ y_1[1,3] \end{bmatrix} \qquad \text{[A.34]}
$$

Por lo tanto, $\tilde{y}_2$ puede a su vez ser expresado como una matriz diagonal multiplicada por el vector de permutación de $\tilde{y}$, $(\tilde{y}_1)$. Ésta, matriz es de un tipo especial llamada *"twiddle factor",* y es definida como:

$$
T_{N,S} = diag[I_R, D_{R,N}, \ldots\ldots\ldots (D_{R,N})^{S-1}] \qquad \text{[A.35]}
$$

Donde el término *diag,* significa matriz diagonal, en nuestro caso:

$$I_R = I_2\,;$$

y

$$D_{R,N} = diag[1,\ w_N,\ .....(w_N)^{R-1}],\quad D_{2,8} = diag[1, w_8] \qquad \text{[A.36]}$$

Por lo tanto:

$$T_{8,4} = diag[I_2, D_{2,8}, ...(D_{2,8})^{4-1}] \qquad \text{[A.37]}$$

En [A.36], se observa que los términos están elevados a las potencias correspondientes de *S*, debe observarse, que cada término de la matriz, es elevado individualmente. En nuestro ejemplo específico tenemos:

$$T_{8,4} = diag[1, 1, 1, w_8, 1, w_8^2, 1, w_8^3] \qquad \text{[A.38]}$$

Por lo tanto,

$$\tilde{y}_2 = T_{8,4} * \tilde{y}_1 \qquad \text{[A.39]}$$

Por último, nos concentraremos en la sumatoria exterior de [A.25], la cual podemos expresarla ahora como:

$$y[\ell_0, \ell_1] = \sum_{k_1=0}^{3} y_2[\ell_1, k_1] v^{k_1 \ell_1}\ , \quad \text{for } 0 \le \ell_0 < 2, \text{ and } 0 \le \ell_1 < 4\,, \qquad \text{[A.40]}$$

Siguiendo un procedimiento similar al realizado para obtener $y_1[\ell_0, k_1]$, podemos encontrar un arreglo, que esté relacionado con el vector de salida $\tilde{y}$, de la siguiente forma:

$$\tilde{y} = (F_4 \otimes I_2)\,\tilde{y}_2 \qquad \text{[A.41]}$$

Combinado los resultados obtenidos anteriormente, tenemos:

$$\tilde{y} = (F_4 \otimes I_2) T_{8,4} (I_4 \otimes F_2) P_{8,4}\, \tilde{x} \qquad \text{[A.42]}$$

O en términos generales:

$$\tilde{y} = (F_S \otimes I_R) T_{N,S} (I_S \otimes F_R) P_{N,S}\, \tilde{x} \qquad \text{[A.43]}$$

A.2 Formulación del Algoritmo en Matlab®.

A continuación se presenta el algoritmo realizado en Matlab®, éste algoritmo presupone que las secuencias son de igual tamaño y de longitud igual a una potencia de 2, y que las mismas se encuentran guardadas en dos directorios llamados D:\data\x.txt; y D:\data\h.txt.

```
% Algoritmo para la convolución cíclica
clear all
clc
load D:\data\x.txt;
load D:\data\h.txt;
xd=x;
hd=h;
N=max(size(x)); %Longitud de las secuencias
stg=log2(N); %Etapas del algoritmo
s=1;
c=[1];
k=1;
while max(size(c))<N  %Creación de campo de constantes de acuerdo
   cn=[];                  % a la longitud de las secuencias
   k=1;
   for i= 1:2:2^s
      cn(i)=sqrt(c(k));
      cn(i+1)=-sqrt(c(k));
      if i==2^s-1
      c=cn;
      s=s+1;
      else
      k=k+1;
      end
   end
end
c=[];
for i=1:N/2  % Almacenamiento de las constantes positivas únicamente
   c(i)=cn(2*i-1);
   kn(i)=cn(2*i);
end
s=1; % Primera etapa
k=1;
inc=0;
j=1;
ti=0;
while s<=stg
```

```
   for i=1: N/2^s; %
       xn(i+inc)=x(i+inc)+c(j)*x(i+inc+N/2^s); % Se observa el computo de x y h en
paralelo
       xn(i+N/2^s+inc)= x(i+inc)-c(j)*x(i+inc+N/2^s); % Multiplicación por las
                                                        % constantes respectivas
       hn(i+inc)=h(i+inc)+c(j)*h(i+inc+N/2^s);
       hn(i+N/2^s+inc)= h(i+inc)-c(j)*h(i+inc+N/2^s);
       temp=inc;
    end
    if (i+temp+N/2^s)==N
       s=s+1;
       x=xn;
       h=hn;
       j=1;
       inc=0;
    else
       inc = inc+N/2^(s-1);
       j=j+1;
    end
 end
yn=x.*h;% Etapa de multiplicación de las dos secuencias "transformadas"
y=yn;
temp=0;
inc=0;
s=s-1;
k=0;
while k~=N
   for i=1: N/2^s; % Primera etapa; se suman y restan los elementos sucesivos de la
                   % secuencia.
      yn(i+inc)=y(i+inc)+y(i+inc+N/2^s);
      yn(i+N/2^s+inc)= y(i+inc)-y(i+inc+N/2^s);
      temp=inc;
   end
   if (i+temp+N/2^s)==N
      y=yn;
      inc=0;
      k=i+temp+N/2^s;
   else
      inc = inc+N/2^(s-1);
   end
end
y=yn;
s=s-1;
r=1;
```

```
while s>0 %En las subsiguientes etapas se necesitan dividir los resultados obtenidos
            %por  las constantes del campo
   nra=2^(s); % Número de raíces a aplicar en la etapa correspondiente
   nacr=2^(r-1); % Número de veces que cada raíz es aplicada en la etapa
   lapr=2^(stg-s)-nacr+1; %lugar de aplicación de la primera raíz a aplicar
   inc=2^(stg-s); %Incremento en los lugares de aplicación de las raíces
   temp=lapr;
   for i=1:nra;
         k=temp;
         if k<=N
            for j=1:nacr;
               yn(k+j-1)=yn(k+j-1)*c(i)^-1; % División por las constantes respectivas
            end
         end
         temp=temp+inc;
    end
   y=yn;
   r=r+1;
   inc2=0;
   p=0;
   while p~=N
      for i=1: N/2^s; %
          yn(i+inc2)=y(i+inc2)+y(i+inc2+N/2^s); % Sumas y restas correspondientes
          yn(i+N/2^s+inc2)= y(i+inc2)-y(i+inc2+N/2^s);
          temp=inc2;
      end
      if (i+temp+N/2^s)==N
         s=s-1;
         y=yn;
         inc=0;
      else
         inc2 = inc2+N/2^(s-1);
      end
      p=i+temp+N/2^(s+1);
   end
 end
yf=1/N*yn; % Resultado final de la convolución
end
```