37

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

MEDICION Y CALCULO DE LOS INDICES ACUSTICOS EN RECINTOS

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

AREA ELECTRICA - ELECTRONICA

P R E S E N T A :

VANESA GARRIDO HERNANDEZ

DIRECTOR: DR. FELIPE ORDUÑA BUSTAMANTE.

MEXICO, D. F.

284672

[redacted]

2000

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Dedico esta tesis:

A mi familia, por su cariño y apoyo en todo momento.

Muy especialmente a mi abuela Esperanza Calderón, por ser un gran ejemplo de fuerza, coraje y superación. Gracias, donde quiera que estés.

Agradecimientos:

A mis profesores, amigos y todo ser viviente que durante todos estos años me han ayudado a ser una mejor persona.

A mi guitarra, por darme buenos ratos de inspiración.

Al programa de becas PROBETEL de Fundación UNAM.

Y finalmente al Dr. Felipe Orduña por su infinita paciencia, por todos sus consejos y sobre todo por compartir todos sus conocimientos, y aún más importante, por su amistad. MUCHAS GRACIAS, Felipe.

Para este trabajo se usó el procesador de textos LATEX

Índice General

Prefacio

Mucho de nuestro tiempo lo pasamos en lugares cerrados, ya sea en un salón de clases, en una oficina o simplemente en nuestra casa; y por su puesto, deseamos que nuestra estancia sea lo más placentera posible. De esta manera, surge la necesidad de evaluar las características acústicas de estos lugares.

Por tal motivo, resulta interesante desarrollar técnicas, métodos, sistemas, etc., que nos permitan evaluar la calidad acústica de los recintos. La evaluación de un recinto no sólo se aplica para salas de conciertos, sino también para salas de conferencias, iglesias, oficinas, cines, etc.

El trabajo realizado en esta tesis está enfocado a la evaluación de las características acústicas de un recinto por medio de ciertos índices, como son: la claridad, el tiempo de reverberación, la definición, etc. Estos índices se obtienen a partir de la respuesta al impulso del recinto. En consecuencia, el cálculo de los índices acústicos implica obtener la respuesta al impulso de la sala. De tal suerte, se requiere hallar un método sencillo y práctico para su obtención.

La forma más simple de medir la respuesta al impulso de un sistema lineal[1] es, precisamente, excitándolo con una señal impulsiva, v. gr. un disparo, el estallido de un globo, etc. Aunque estas señales presentan relaciones de señal a ruido muy bajas y problemas de repetibilidad.

Otra manera de determinar la respuesta al impulso es excitándo al sistema con ruido blanco y correlacionando la entrada (señal de excitación) y la salida (respuesta a tal excitación) del sistema (en este caso, el cuarto). A fin de reducir la cantidad de cómputo que se necesita para determinar la correlación y para tener una alta inmudad al ruido, el cuarto (sistema) es excitado con secuencias de máxima longitud (método propuesto por Schroeder [16]) y la correlación se calcula con la Transformada rápida de Hadamard [3, 5].

Este trabajo conjunta la técnica que usa las secuencias de máxima longitud como señal de excitación y el cálculo de la correlación con la Transformada rápida de Hadamard, para obtener la respuesta al impulso. Por medio de un sistema de procesamiento digital de

[1] Para este trabajo, el cuarto o recinto a analizar se identifica como el sistema lineal.

señales (basado en el procesador TMS320C30 de Texas Instruments) fue posible desarrollar un sistema de medición de la respuesta del recinto que genera las secuencias ML [2] y al mismo tiempo captura la respuesta del recinto. Una vez obtenida la respuesta del recinto, se calcula la correlación con un programa que realiza la Transformada rápida de Hadamard.

Los índices acústicos se determinan a partir de la respuesta al impulso. La mayor parte de estos índices se relacionan en forma más o menos simple con la energía de la respuesta al impulso del recinto en diferentes intervalos de tiempo, y fue factible desarrollar programas en `Matlab` para su cálculo. A manera de demostración del trabajo realizado, se evualarán algunos índices acústicos de respuestas al impulso simuladas o medidas de una sala.

Breve Historia de la Acústica Arquitectónica

El desarrollo de la acústica arquitectónica ha pasado por tres etapas en las que ha ido cambiando el enfoque para el análsis y diseño de un recinto [7]. Comenzando con los trabajos realizados por Wallace Clement Sabine e introduciendo en los primeros años del siglo diecinueve la tan conocida relación $T = kV/A$. Hubo un periodo de consolidación durante el cual todos los diseños acústicos se basaban solamente en el control del tiempo de reverberación. El descubrimiento de Sabine fue de tal trascendencia y avance, que en un principio parecía que todos los problemas de la acústica arquitectónica se podían resolver con ella. Las sutilezas de la percepción de las reflexiones como componentes del proceso de reberveración no aparecían como factores importantes en los próximos 50 años.

La segunda fase, que comenzó después de la Segunda Guerra Mundial, dirigió su atención a la importancia de la reflexiones tempranas que cuenta entre otras cosas, por el hecho observado de que el tiempo de reverberación no siempre representa una impresión subjetiva adecuada. Además, la noción de que la experiencia de escuchar satisfactoriamente en una sala de conciertos depende de los detalles temporales de las reflexiones tempranas se extendió significativamente con el descubrimiento de que las reflexiones laterales realizan una importante contribución para la sensación de espaciosidad. Esto es, ambas propiedades espaciales y temporales de las reflexiones tempranas deben ser controladas con el diseño de la sala además del tiempo de reverberación. La forma del cuarto, ignorada por la relación de Sabine, se debe tomar en cuenta en el diseño de la sala. El tiempo de integración y el umbral absoluto de percepción para las reflexiones tempranas y el umbral para el eco para el lenguaje y otras señales se volvió el centro de la investigación.

Aún de mayor importancia, fue la aplicación del análisis de factores (o análisis de componentes principales) para experimentos de preferencia rigurosamente diseñados, la demostración de que la preferencia es multidimensional y el aislamiento de los 4 principales

[2] Abrev. Maxima Longitud

factores subjetivos que la determinan. Se propusieron correlaciones objetivas para tales factores: Tiempo de Reverberación (RT) y el Tiempo de decaimiento temprano (EDT) para la reverberancia, el nivel total referido al nivel de sonido directo a 10 m de la fuente (G) para la fuerza, la porción de energía en 80 ms (C80) y la medición relacionadas como el Tiempo Central de Cremer (Tc) para la claridad; y cualquier fracción lateral (LF) de 80 ms o la correlación interauricular (IACC) para la impresión espacial.

En las últimas decadas, la tercera fase del desarrollo de la Acústica Arquitectónica ha centrado su atención en el refinamiento del criterio de preferencia del escucha.

Areas abiertas de investigación están cambiando el marco de preferencia y las formas de arte evolucionan. Uno sólo tiene que ver como ha cambiado el panorama en 50 años desde que Floyd Watson en 1941 supuso que los hábitos al escuchar desprenden desarrollos en nuevos equipos. Finalmente, las propias formas de arte cambian, nuevamente, seguida de los avances técnicos. Las últimas dos décadas del siglo XX han visto el extenso desarrollo de las computadoras digitales que abren nuevos caminos de posibilidades en lo referente a la música - y hasta lo inimaginable - incluyendo la interacción de lo visual y lo audible.

Contenido de la tesis.

En el Capítulo 1 se definirán algunos conceptos fundamentales de Acústica [3]. Rasgos muy generales de la propagación del sonido en el aire. Se determinarán los conceptos más importantes para el análisis de sistemas y señales y el análisis estadístico de señales aleatorias.

En el Capítulo 2, se presentarán dos diferentes aproximaciones para describir el campo sonoro en un cuarto. El primero involucra los modos normales de vibración de un recinto para describir la distribución de la presión sonora en el cuarto. Una descripción más ilustrativa se obtiene restringiendo la discusión a la propagación y distribución de la energía. Esto se lleva a cabo introduciendo rayos sonoros y fuentes imagen que producen un método sencillo del cálculo de la distribución de la energía sonora. Posteriormente, se definirán algunos parámetros con los que es posible evalúar las características acústicas de un recinto, y por último, se describirán las principales técnicas de medición de la respuesta al impulso.

En el Capítulo 3, se tratará lo concerniente a la técnica propuesta por Schroeder; se definirán lo que son las secuencias de máxima longitud, sus propiedades, la forma de generación, etc. Existe un algoritmo muy eficiente para el cálculo de la correlación: la Transformda rápida de Hadamard. Se describirá éste procedimiento de cálculo.

[3] No es intención del autor el hacer una descripción muy detallada de la Acústica y de la Acústica arquitectónica, pues el trabajo desarrollado en esta tesis no lo exige. Solamente se abordarán temas que a juicio del autor son importantes para la comprensión de algunos conceptos que se discutirán en capítulos ulteriores.

En el Capítulo 4, se implementará la técnica de obtención de la respuesta al impulso del recinto que tiene como señal de excitación, las secuencias de máxima longitud y para el cálculo de la correlación (respuesta al impulso), la transformada rápida de Hadamard. Además se calcularán los índices acústicos a partir de respuestas al impulso medidas o simuladas. Finalmente, se analizarán los resultados obtenidos.

En el Capítulo 5, se presentarán las conclusiones del trabajo.

Capítulo 1

Fundamentos de acústica y de análisis de señales

En el presente Capítulo se definirán algunos conceptos fundamentales de Acústica [10, 2]. Rasgos muy generales de la propagación del sonido en el aire que nos servirán para posteriorme hacer una descripción de la teoría ondulatoria de la acústica de recintos (veáse Capítulo 2). Se determinarán los conceptos más importantes para el análisis de sistemas y señales [20, 13] y el análisis estadístico de señales aleatorias.

1.1 Conceptos fundamentales de acústica

En esta sección definiremos la ecuación de onda, la solución general a esta ecuación en el caso de una dimensión, y por último determinaremos el concepto de nivel de presión sonora y sonoridad.

1.1.1 La ecuación de onda

En el siguiente apartado primero estableceremos las ecuaciones expresadas en la Segunda ley de Newton, la ley de los gases y las leyes de conservación de la masa. Finalmente, combinaremos estas ecuaciones para deducir la ecuación de onda.

La Segunda ley de Newton (ecuación de movimiento) es expresada por la siguiente relación

$$-\text{grad}p = \rho_0 \frac{\partial u}{\partial t} \tag{1.1}$$

donde p indica la presión sonora, u el vector velocidad de la partícula, t el tiempo y ρ_0 el valor estático de la densidad del gas.

Ley de los gases

Si asumimos que el gas es ideal, la ley de los gases está definida por la siguiente ecuación

$$PV = NRT \tag{1.2}$$

donde P es la presión, V es el volumen, N es el número de moles de gas, R es la constante del gas y T es la temperatura en grados Kelvin. Usando la ecuación anterior podemos hallar la relación entre la presión sonora y las variaciones en el volumen. Antes de establecer esta relación, sin embargo, debemos conocer como varía la temperatura T con los cambios de P y V, en particular cuando el proceso es adiabático.

Los cambios en la presión del gas, ocasionados por las ondas sonoras a frecuencias de audio, ocurren de manera que no hay transferencia de calor entre los elementos de volumen adyacentes. Como resultado, una onda sonora producirá variaciones adiabáticas de temperatura, y de esta manera, la temperatura también puede considerarse como una cantidad característica de una onda sonora. Para procesos adiabáticos, la relación entre la presión y el volumen es

$$PV^{\gamma} = constante \tag{1.3}$$

De la ecuación(1.3) obtenemos la relación entre la presión sonora y las variaciones del volumen [1]. Esto es,

$$\frac{p}{P_0} = -\gamma \frac{V}{V_0} \tag{1.4}$$

siendo $p = \Delta P = P - P_0$, $V = \Delta V = V - V_0$ y donde γ es la relación del calor específico del gas a presión constante y el calor específico a volumen constante. Para el aire, nitrógeno, hidrógeno y oxígeno $\gamma = 1.4$.

La derivada con respecto al tiempo de la ecuación (1.4) es

$$\frac{1}{P_0}\frac{\partial p}{\partial t} = -\frac{\gamma}{V_0}\frac{\partial V}{\partial t} \tag{1.5}$$

La ecuación de continuidad

La ecuación de continuidad es la expresión matemática del principio de conservación de la masa, que dice: la masa total de un gas en un medio deformable deberá permanecer constante. Por la ley de conservación de la masa, podemos escribir una sola relación entre el incremento de la velocidad y la superficie de la porción de volumen de gas.

Si la masa del gas permanece constante, el incremento del volumen V sólo depende de la divergencia del vector desplazamiento ξ, esto es

$$V = V_0 \mathrm{div}\xi \tag{1.6}$$

[1] la deducción de la relación anterior puede verse con mayor detalle en [2].Capítulo 2.

Derivando con respecto al tiempo,

$$\frac{\partial V}{\partial t} = V_0 \text{div} u \tag{1.7}$$

Donde u es la velocidad de la partícula.

La ecuación de onda.

La ecuación de onda se obtiene al combinar la ecuación de movimiento (1.1), la ley de los gases (1.4) y la ecuación de continuidad (1.7).

Combinando (1.5) y (1.7)

$$\frac{\partial p}{\partial t} = -\gamma P_0 \text{div} u \tag{1.8}$$

Derivando con respecto a t

$$\frac{\partial^2 p}{\partial t^2} = -\gamma P_0 \text{div} \frac{\partial u}{\partial t} \tag{1.9}$$

Tomando la divergencia de ambos lados de la ecuación (1.1)

$$-\nabla^2 p = \rho_0 \text{div} \frac{\partial u}{\partial t} \tag{1.10}$$

Si combinamos la ecuación (1.9) y (1.10) tenemos

$$-\nabla^2 p = \frac{\rho_0}{\gamma P_0} \frac{\partial^2 p}{\partial t^2} \tag{1.11}$$

Por definición

$$c^2 \equiv \frac{\gamma P_0}{\rho_0} \tag{1.12}$$

Obtenemos la ecuación de onda

$$\nabla^2 p = \frac{1}{c^2} \frac{\partial^2 p}{\partial t^2} \tag{1.13}$$

1.1.2 Solución a la ecuación de onda

Ahora, si asumimos que las cantidades acústicas dependen sólo del tiempo y en una sola dirección, podemos escoger la dirección del eje x del sistema de coordenadas cartesianas. Entonces, la ecuación de onda en una dimensión es

$$\frac{\partial^2 p}{\partial^2 x} = \frac{1}{c^2} \frac{\partial^2 p}{\partial t^2} \tag{1.14}$$

la ecuación anterior puede quedar en términos de la velocidad de la partícula

$$\frac{\partial^2 u}{\partial^2 x} = \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} \tag{1.15}$$

Solución general

La solución general a la ecuación (1.14) es la suma de dos términos,

$$p(x,t) = F(ct - x) + G(ct + x) \tag{1.16}$$

donde F y G son funciones arbitrarias, que sus segundas derivadas existen. El primer término de la ecuación representa a una onda de presión que viaja en la dirección positiva de x con una velocidad c. El segundo término describe a una onda de presión que se propaga en la dirección negativa de x. Donde c es la velocidad del sonido. El valor de c a una presión ambiente de 10^5 Pa y a una temperatura de 22° C es de 344.2 m/s.

Solución armónica.

Como es sabido de la teoría de las series de Fourier, una señal periódica se puede representar como una suma de funciones senoidales, de la forma

$$\Psi(t) = \Phi\cos(\omega t + \theta) \tag{1.17}$$

Por ejemplo, si Ψ es la presión sonora, escribimos

$$p(t) = \sum_i p_i(t) = \sum_i p_i \cos(\omega_i t + \theta_i) \tag{1.18}$$

Donde $\omega_i = 2\pi f_i$; f_i es la frecuencia del i-ésimo componente de la onda, θ_i es el ángulo de fase y Φ (o p_i) es el valor pico de la componente.

Podemos representar una onda senoidal de frecuencia ω como la parte real de una función exponencial compleja. Entonces, en un punto fijo en el espacio x, la presión sonora es

$$p(x,t) = Re[p(x)e^{\omega t}] \tag{1.19}$$

Generalmente omitimos escribir Re aunque siempre se debe de recordar que sólo se toma la parte real cuando se usa la expresión final para la presión sonora. En estado estacionario, entonces, podemos reemplazar F y G de la ec (1.16) por una suma de funciones con una frecuencia angular ω, de tal manera que

$$p(x,t) = \sum_i p_i(x,t) = \sum_i [\{p_+ e^{-j\omega_i x/c} + p_- e^{j\omega_i x/c}\} e^{j\omega_i t}] \tag{1.20}$$

Donde el subíndice + y − indica ondas viajando hacia adelante y atrás respectivamente.

El número de onda k se define,

$$k \equiv \frac{\omega}{c} = \frac{2\pi f}{c} = \frac{2\pi}{\lambda} \tag{1.21}$$

De tal manera que los términos de la solución a la ecuación de onda quedan de la forma

$$p(x,t) = p(x)e^{j\omega t} = pe_+e^{jk(ct-x)} + pe_-e^{jk(ct+x)} \tag{1.22}$$

De la misma forma la solución a la ecuación de onda en términos de la velocidad de la partícula es

$$u(x,t) = ue_+e^{jk(ct-x)} + ue_-e^{jk(ct+x)} \tag{1.23}$$

1.1.3 Nivel de presión sonora

En el rango de frecuencias en el que nuestro oído es mas sensible (2000 a 5000 Hz) el umbral de sensación y el umbral de dolor al escuchar se separan por alrededor de 13 ordenes de magnitud de intensidad. Por esta razón resulta poco práctico caracterizar al sonido en términos de su presión sonora o de su intensidad. En cambio, se emplea lo que se llama *nivel de presión sonora* que se usa generalmente para este propósito, definido por

$$L = 20\log_{10}(\frac{\tilde{p}}{\tilde{p_0}}) \text{ [dB]} \tag{1.24}$$

Donde los valores con una tilde son valores efectivos (rms) de presión y el valor p_0 de referencia es $2 \times 10^{-5}\text{N/m}^2$ que corresponde al umbral normal de audibilidad a 1000 Hz.

1.2 Análisis de Sistemas Dinámicos Lineales.

1.2.1 Señales Discretas y Sistemas.

Una señal se puede definir como una función del tiempo que transfiere información generalmente acerca del estado o comportamiento de un sistema físico. Existen señales de tiempo continuo, o de tiempo discreto dependiendo si la variable independiente, tiempo, se toma en forma continua o como una serie de valores discretos. Las señales de tiempo discreto son en estos términos secuencias de números reales o complejos. Frecuentemente estas secuencias de números se obtienen al tomar muestras uniformemente espaciadas de una señal de tiempo continuo.

Una señal discreta se define como una función de variable independiente entera. La notación $f[n]$ determinará una secuencia de números, reales o complejos, definidos para cada entero n.

Esta secuencia $f[n]$ es en si, una señal discreta o digital, y el índice n indica tiempo discreto.

Las siguientes funciones son casos especiales que se emplean de manera frecuente:

Secuencia Escalón

$$u[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases} \tag{1.25}$$

Véase Figura (1.1a)

Secuencia Delta

Una secuencia delta (Figura 1.1b) se define como

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases} \tag{1.26}$$

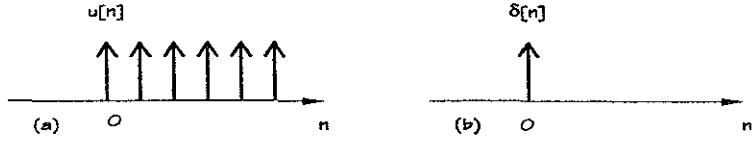


Figura 1.1: Representación gráfica de la secuencia escalón (a) y la secuencia delta (b).

Si una señal $f[n]$ se retrasa m unidades de tiempo entonces $y[n]$ se vuelve $y[n-m]$. Una forma útil de representar una secuencia arbitraria $f[n]$ es descomponerla en una suma de secuencias delta desplazadas, multiplicadas por los valores de la secuencia $f[n]$

$$\sum_{m=\infty}^{\infty} f[m]\delta[n-m] \tag{1.27}$$

Lo anterior se ilustra con la Figura 1.2

Sistema discreto.

Un sistema discreto se puede ver como cualquier proceso que produce una transformación de señales. Entonces, un sistema tiene una señal de entrada $x[n]$ y una señal de salida $y[n]$ la cual esta relacionada con la entrada a través de la transformación del sistema (Figura 1.3)

$$y[n] = L\{x[n]\} \tag{1.28}$$

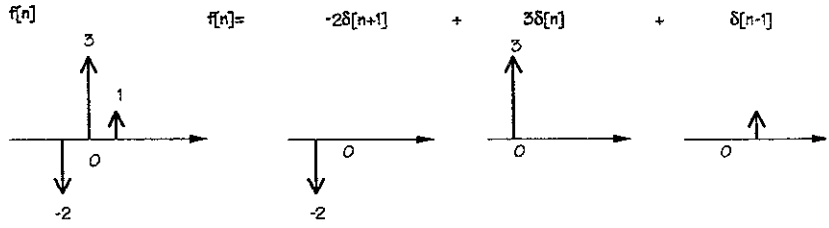


Figura 1.2: Representación de la señal f[n] como sumas de secuencias delta.

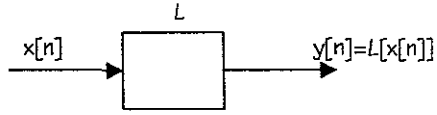


Figura 1.3: Sistema discreto.

En general, para determinar el valor de la salida $y[n]$ para una n específica, debemos conocer la entrada $x[n]$ para cada n pasada o futura.

Propiedades.

- Linealidad.

  Un sistema L es lineal si

  $$L\{a_1 f_1[n] + a_2 f_2[n]\} = a_1 L\{f_1[n]\} + a_2 L\{f_2[n]\} \tag{1.29}$$

  para cualquier a_1 , a_2, $f_1[n]$ y $f_2[n]$.

- Invariancia en el tiempo.

  Un sistema invariante con el tiempo es aquel que sus propiedades no varían con el tiempo. El único efecto de un corrimiento en el tiempo de la señal de entrada en el sistema es un corrimiento en el tiempo de la señal de salida.

1.2.2 Respuesta al Impulso.

La respuesta de un sistema discreto lineal e invariante con el tiempo a una secuencia delta $\delta[n]$ (o también llamada muestra unitaria) es denominada respuesta a la muestra unitaria,

o respuesta al impulso del sistema, indicada como $h[n]$.

$$L\{\delta[n]\} \equiv h[n] \tag{1.30}$$

1.2.3 Relación entrada - salida. Convolución.

Como hemos descrito anteriormente, una secuencia $x[n]$ podemos representarla como una suma de secuencias $\delta[n]$ multiplicada por los valores de la secuencia $x[n]$.

Tenemos por definición que la respuesta a $\delta[n]$ es $h[n]$.
Como es un sistema lineal, la respuesta a $c\delta[n]$ es $ch[n]$.
Además, al ser invariante con el tiempo, la respuesta a $c\delta[n-m]$ es $ch[n-m]$. Basándonos en lo anterior, para un sistema lineal e invariante con el tiempo tenemos

$$\begin{aligned} y[n] &= L\{x[n]\} \\ &= L\{\sum_{m=-\infty}^{\infty} x[m]\delta[n-m]\} \\ &= \sum_{m=-\infty}^{\infty} x[m]L\{\delta[n-m]\} \\ &= \sum_{m=-\infty}^{\infty} x[m]h[n-m] \end{aligned} \tag{1.31}$$

La ecuación anterior nos da la respuesta $y[n]$ de un sistema LTI[2] como función de la señal de entrada $x[n]$ y la respuesta al impulso $h[n]$ es denominada sumatoria de convolución.

La operación de convolución se indica con $*$

$$y[n] = x[n] * h[n] \tag{1.32}$$

La respuesta al impulso de un sistema a una señal de entrada $x[n]$ está dada por la convolución en tiempo discreto.

1.3 Análisis estadístico de señales aleatorias.

En muchas situaciones, los procesos para generar señales son tan complejos que realizar una descripción precisa de una señal resulta extremadamente difícil o indeseable, si no imposible. En tales casos, modelar la señal en términos de probabilidad es de gran utilidad.

Se emplean 4 tipos de funciones estadísticas para describir señales aleatorias [12]:

[2]Linear Time Invariant System (Lineal Invariante con el tiempo)

1. Valores medios cuadráticos y la varianza - Nos dan información sobre la amplitud de la señal.

2. Distribuciones de probabilidad - Nos dan información sobre las propiedades estadísticas de la señal en el dominio de la amplitud.

3. Funciones de correlación - Nos dan información sobre las propiedades estadísticas de la señal en el dominio del tiempo.

4. Funciones de densidad espectral - Nos dan información sobre las propiedades estadísticas en el dominio de la frecuencia.

1.3.1 Correlación.

Secuencias de Correlación y autocorrelación.

Supongamos que tenemos dos secuencias reales $x[n]$ y $f[n]$ de longitud finita. La correlación de $x[n]$ y $f[n]$ es la secuencia, que se define como

$$\Phi_{xf}[n] = \sum_{m=-\infty}^{\infty} x(m)f(m+n) \tag{1.33}$$

O de forma equivalente como

$$\Phi_{xf}[n] = \sum_{m=-\infty}^{\infty} x(m-n)f(m) \tag{1.34}$$

El índice n es el parámetro de corrimiento y el subíndice xf en la secuencia de correlación $\Phi_{xf}[n]$ indica las secuencias que son correlacionadas.

Las similitudes entre el cálculo de la correlación y la convolución de dos secuencias es aparente. En el cálculo de la convolución, una de las secuencias se "dobla" ($x[-n]$), luego se corre, para después multiplicarla por la otra secuencia y finalmente los valores del producto se suman. Excepto por la operación de doblamiento, el cálculo de la secuencia de correlación involucra las mismas operaciones: corrimiento de una de las secuencias, multiplicación de las dos secuencias, y sumar todos los valores de las secuencias producto. Entonces, la convolución de $x[-n]$ con $f[n]$ produce la correlación $\phi_{xf}[n]$, que es,

$$x_{xf}[n] = x[-n] * f[n] \tag{1.35}$$

Para el caso en que $x[n] = f[n]$, tenemos la autocorrelación de $x[n]$, que se define como la secuencia

$$\Phi_{xx}[n] = \sum_{m=-\infty}^{\infty} x(m)x(m+n) \tag{1.36}$$

O de forma equivalente

$$\Phi_{xx}[n] = \sum_{m=-\infty}^{\infty} x(m-n)x(m) \tag{1.37}$$

Propiedades.

1. La autocorrelación es una secuencia par, i. e. ,

$$\Phi_{xx}[n] = \Phi_{xx}[-n] \tag{1.38}$$

2. La correlación satisface

$$\Phi_{xf}[n] = \Phi_{fx}[-n] \tag{1.39}$$

3. La correlación se puede realizar por medio de la convolución (representada por $*$) de la siguiente manera

$$\Phi_{xf}[n] = x[-n] * f[n] \tag{1.40}$$

o bien

$$\Phi_{fx}[n] = f[n] * x[-n] \tag{1.41}$$

Secuencias de correlación Entrada - Salida.

Asumamos que una señal $x[n]$ con una determinada autocorrelación $\Phi_{xx}[n]$ se aplica a un sistema LTI con una respuesta al impulso $h[n]$, produciendo la señal de salida

$$y[n] = h[n] * x[n] = \sum_{m=-\infty}^{\infty} h(n)x(n-m) \tag{1.42}$$

La correlación entre la salida y la entrada es

$$\Phi_{yx}[n] = y[-n] * x[n] = h[-n] * [x[-n] * x[n]] \tag{1.43}$$

O

$$\Phi_{yx}[n] = h[-n] * \Phi_{xx}[n] \tag{1.44}$$

Si sustituimos n por $-n$, obtenemos

$$\Phi_{xy}[n] = h[n] * \Phi_{xx}[-n] \tag{1.45}$$

Un caso particular de la ecuación anterior se presentaría cuando la autocorrelación de la señal de entrada fuera igual a una señal delta. Entonces, la respuesta al impulso sería equivalente a la correlación de la entrada con la salida. Esto es,

$$\Phi_{xx}[n] = \delta[n] \tag{1.46}$$

entonces,

$$\Phi_{xy}[n] = h[n] \tag{1.47}$$

El resultado anterior es la parte medular del método de medición usado en esta tesis como se verá en los próximos capítulos.

Capítulo 2

Fundamentos de Acústica de Recintos.

En un recinto, el sonido viaja distancias cortas antes de encontrarse con paredes y otros obstáculos. Estos obstáculos reflejan y absorben sonido de manera que determinan considerablemente las propiedades acústicas del cuarto.

En este capítulo se presentarán dos diferentes aproximaciones para describir el campo sonoro en un cuarto [2, 8]. El primero involucra los modos normales de vibración de un recinto para describir la distribución de la presión sonora en el cuarto (Sección 2.1). La descripción modal es útil a bajas frecuencias ya que la densidad modal (número de modos de vibración en un determinado rango de frecuencias) aumenta rápidamente para altas frecuencias. Una descripción distinta se obtiene restringiendo la discusión a la propagación y distribución de la energía. Esto se lleva a cabo introduciendo rayos sonoros y fuentes imagen que producen un método sencillo del cálculo de la distribución de la energía sonora (Sección 2.2). Esta descripción geométrica en términos de rayos de sonido es una aproximación válida a altas frecuencias. Por lo tanto, la descripción modal y la geométrica se complementan una con la otra en la descripción de la acústica de un recinto.

Posteriormente se definirán algunos parámetros que sirven para evaluar las características acústicas de un recinto, y por último se describirán las principales técnicas de medición de la respuesta al impulso [18].

2.1 Descripción Modal.

2.1.1 Campos Sonoros en recintos rectangulares.

Cuando una fuente de sonido a cierta frecuencia se coloca en un recinto de forma rectangular, excitará uno o más de la infinidad de condiciones de resonancia, llamados modos normales de vibración. La frecuencia f_n de un modo normal de vibración se denomina

frecuencia natural. Las frecuencias naturales de un recinto rectangular se obtienen de la siguiente ecuación,

$$f_n = \frac{c}{2}\sqrt{\left(\frac{n_x}{l_x}\right)^2 + \left(\frac{n_y}{l_y}\right)^2 + \left(\frac{n_z}{l_z}\right)^2} \tag{2.1}$$

Donde f_n es la n-ésima frecuencia natural en Hz; n_x, n_y, n_z son enteros que se pueden escoger de manera independiente y que pueden tomar valores enteros entre 0 e ∞; lx, ly, lz son dimensiones del cuarto en m; c es la velocidad del sonido en m/s.

La distribución de presión sonora en un recinto rectangular para cada modo normal de vibración con una frecuencia natural ω_n es igual al producto de 3 cosenos,

$$p_{n_x,n_y,n_z} \propto \cos\frac{\pi n_x x}{l_x}\cos\frac{\pi n_y y}{l_y}\cos\frac{\pi n_z z}{l_z} \tag{2.2}$$

Donde el origen de las coordenadas se encuentra en la esquina del cuarto. Se asume que las paredes tienen muy poca absorción.

Si inspeccionamos la ecuación anterior a detalle, observamos que n_x, n_y, n_z indican el número de planos de presión cero que ocurren a lo largo de x, y, y z respectivamente. Tal distribución del nivel de presión sonora se puede representar como una superposición de ondas planas.

2.1.2 Presión sonora transitoria y en estado estable.

Cuando una fuente de sonido se enciende en un recinto, excitará de manera predominante a uno o más modos normales de vibración. Asumiendo que la amplitud de la fuente es constante en función de la frecuencia, y que la frecuencia empleada varía alrededor de la frecuencia natural de un cierto modo, entonces la contribución de este modo a la presión sonora total será la que corresponde a la curva de resonancia estándar mostrada en la Figura (2.1). El ancho de la curva de resonancia en puntos a mitad de la potencia (3 dB abajo) es igual a

$$f'' - f' = \frac{k_n}{\pi} \tag{2.3}$$

La magnitud de la presión sonora p_n (contribuida por el modo n) está dada por

$$|p_n| = \frac{2K\omega}{\sqrt{4\omega_n^2 k_n^2 + (\omega^2 - \omega_n^2)}} \tag{2.4}$$

Donde ω es la frecuencia angular manejada y ω_n es la frecuencia angular natural dada aproximadamente por la ecuación(2.1).

Si la frecuencia usada cae entre dos frecuencias naturales o si k_n es grande de manera que la curva de resonancia sea ancha, más de un modo de vibración se excitará.

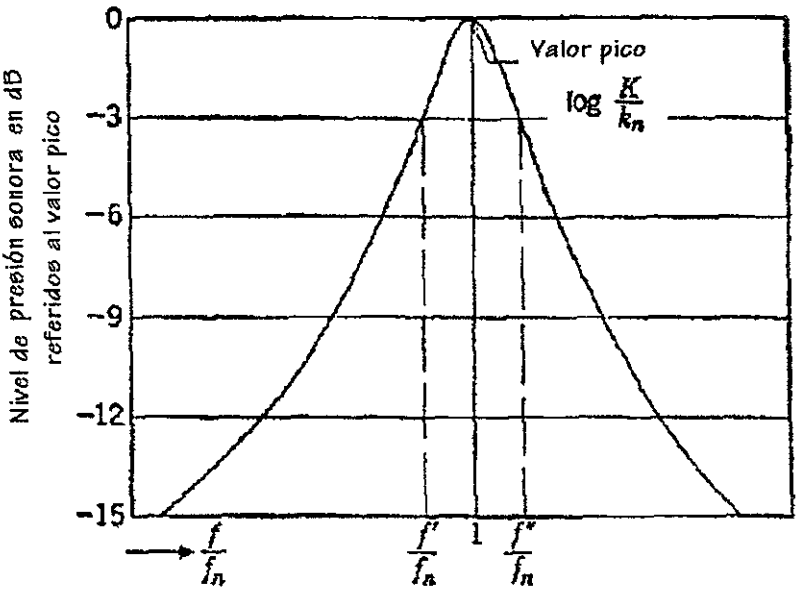


Figura 2.1: Curva de resonancia para un modo normal de vibración. Nivel de presión sonora vs. la relación de frecuencia a f_n

Cuando la fuente es de una sola frecuencia y esta frecuencia coincide con una de las frecuencias naturales del recinto, la presión sonora para tal modo de vibración tomará un valor rms en estado estacionario igual a

$$|p_n| = \frac{K}{k_n} \tag{2.5}$$

Donde K depende de la fuerza y de la ubicación de la fuente y del volumen del cuarto. Por otra parte, k_n es una constante de amortiguamiento determinada principalmente por la cantidad de absorción en el cuarto y por el volumen del mismo. Entre más material absorbente sea introducido al cuarto el valor k_n será más grande.

Cuando la fuente de sonido es apagada, cada modo normal de vibración se comporta como un circuito eléctrico resonante paralelo en el que la energía se ha almacenado inicialmente. La presión para tal modo normal de vibración decaerá exponencialmente en su propia frecuencia natural como se muestra en la Figura (2.2).

Si sólo un modo de vibración es excitado, el decaimiento será como se muestra en la

Figura (2.2a) y estará expresado por la siguiente ecuación

$$p_n = \frac{K}{k_n} e^{-k_n t} \cos \omega_n t \tag{2.6}$$

En una escala del *log* p_n vs tiempo, la magnitud del nivel de presión sonora efectivo decaerá linealmente con el tiempo.

Si dos o más modos de vibración están decayendo simultáneamente, existirían ciertas pulsaciones debido a que cada una tiene su propia frecuencia natural (Figura 2.2*b* y *c*).

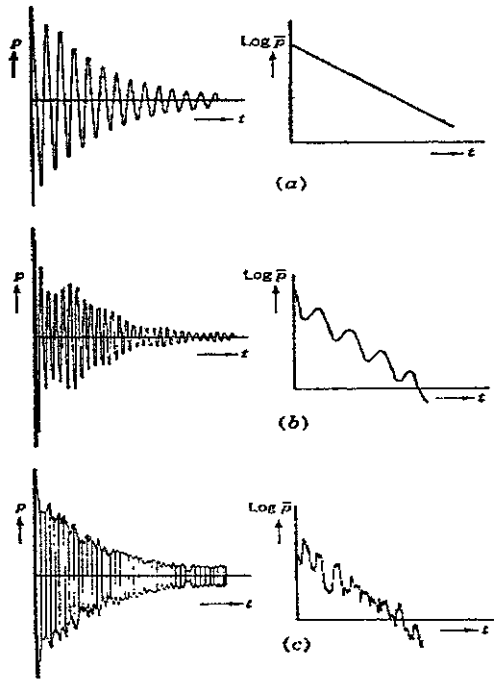


Figura 2.2: (a) Curva de decaimiento de la presión sonora para un sólo modo de vibración. (b) Curva de decaimiento para dos modos de vibración con la misma constante de decaimiento. (c) Curva de decaimiento para un número de modos de vibración con la misma constante de decaimiento. Las gráficas de la izquierda muestran el curso de la presión sonora instantánea, y las de la derecha muestran la envolvente de las curvas de la izquierda, graficadas en $\log \bar{p}$ vs t.

Además, es posible que cada una tenga su propia constante de decaimiento, depen-

diente de la posición de los materiales absorbentes en el cuarto. En tal caso, la magnitud de las curvas de nivel de presión sonora decaerá con dos o más pendientes como se muestra en la Figura (2.3).

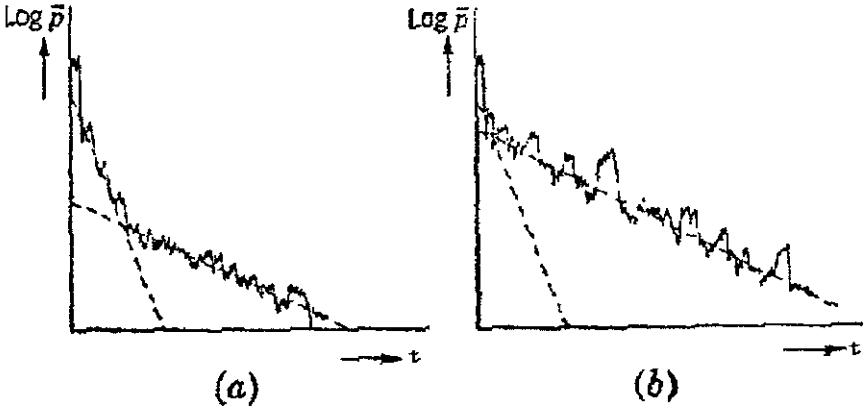


Figura 2.3: Curvas de decaimiento con pendientes dobles producidas por modos normales de vibración con diferentes constantes de decaimiento.

2.2 Descripción Geométrica.

Cuando se tienen recintos de forma irregular o muy grandes, el análisis del campo sonoro a través de sus modos normales de vibración resulta muy complicado. El estudio del comportamiento de las ondas sonoras en un recinto se puede simplificar si sustituimos las ondas de compresión y rarefacción por rayos sonoros imaginarios, perpendiculares al frente de onda progresivo, viajando en línea recta en cierta dirección del espacio, de manera similar a los haces de luz en la óptica. Esta aproximación en la acústica arquitectónica, que compara el comportamiento de las ondas sonoras en rayos, es llamada acústica geométrica. Cuando una onda sonora choca con las paredes del cuarto, parte de su energía se reflejará, y parte de ella será absorbida.

2.2.1 Reflexión del sonido.

En superficies rígidas y lisas[1], como el concreto, el vidrio, el yeso, etc., se reflejará casi toda la energía del sonido incidente que choque con la superficie. Los rayos sonoros incidentes

[1]En donde el calificativo de "liso" depende del rango de longitudes de onda (o de frecuencias) que se considere.

y reflejados están situados en el mismo plano y el ángulo de incidencia de la onda sonora es igual al ángulo de reflexión (Ley de la reflexión). En la Figura (2.4) el rayo sonoro 1 y 3 ilustran el fenómeno de la reflexión sonora. La ley de la reflexión sonora es válida sólo si la longitud de onda λ de las ondas sonoras es pequeña comparada con las dimensiones de las superficies reflejantes. Las superficies reflejantes cóncavas tenderán a concentrar mientras que las superficies convexas dispersarán las ondas sonoras reflejadas en los cuartos.

2.2.2 Difracción.

Este fenómeno acústico ocasiona que las ondas sonoras se encorven y sean esparcidas alrededor de los obstáculos (esquinas, columnas, paredes, pilas, etc.), de manera que estos elementos no producen una sombra acústica completa como se muestra en el área 9 de la Figura (2.4), pero se producirán una especie de "flecos" alrededor de los obstáculos, como se muestra en el área 10 de la misma Figura. La difracción alrededor de los obstáculos será más pronunciada para bajas frecuencias que para altas frecuencias. Esto prueba que las leyes de la acústica geométrica no son adecuadas para determinar de forma precisa el comportamiento del sonido en espacios cerrados porque los obstáculos que normalmente se encuentran en un recinto son demasiado pequeños comparados con las longitudes de onda del rango de frecuencias audibles. La acústica geométrica, es una aproximación útil para problemas relacionados con sonido a altas frecuencias y se aplica difícilmente para frecuencias abajo de los 250 Hz.

2.2.3 Absorción de sonido. Coeficiente de absorción.

Es bien sabido que los materiales suaves y porosos, los muebles y la gente absorben una porción considerable de la energía de las ondas sonoras que chocan en ellos, en otras palabras, son absorbentes de sonido. Por definición, la absorción del sonido es el cambio de la energía del sonido en alguna otra forma de energía, frecuentemente en calor. La cantidad de calor producido por la conversión de la energía del sonido en energía calorífica es extremadamente pequeña. En los diferentes tipos de auditorios, los siguientes elementos contribuyen a la absorción del sonido en el recinto: (a) el tratamiento de las superficies del recinto, como las paredes, el piso, el techo (véase el área 6 de la Figura (2.4).); (b) el contenido del cuarto, es decir, la audiencia, los asientos, las cortinas, las alfombras, etc. La eficiencia de absorción de sonido de un material a cierta frecuencia es determinada por el coeficiente de absorción. Por definición, el coeficiente de absorción de una superficie es la fracción de energía sonora absorbida o no reflejada por la superficie. Se denota por la letra griega alfa α. El valor α de diferentes materiales puede variar entre 0 y 1. Estos valores se encuentran en tablas que se pueden hallar en la literatura o son datos proporcionados por el fabricante.

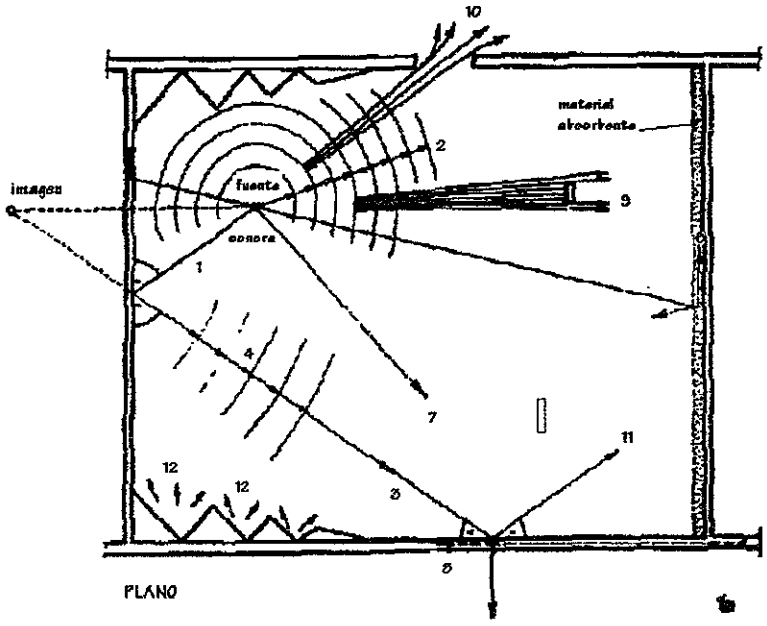


Figura 2.4: Comportamiento del sonido en un espacio cerrado.1 sonido incidente. 2 frente de onda directo. 3 sonido reflejado. 4 frente de onda reflejado. 5 sonido transmitido a través del recinto. 6 sonido absorbido por la pared. 7 sonido absorbido por el aire. 8 energá sonora disìpada dentro de la estructura. 9 sombra acústica. 10 difracción del sonido. 11 multiples reflexiones que contribuyen a la reverberación. 12 sonido difuso debido a irregularidades en la superficie.

2.2.4 Crecimiento y decaimiento del sonido en un cuarto.

Cuando una onda estacionaria es generada en un cuarto la presión sonora irá aumentando gradualmente y tomara algún tiempo, en la mayoría de los cuartos en alrededor de 1 segundo, hasta que alcance su valor de estado estacionario. Si el campo sonoro es difuso en el cuarto, esto es, que la energía sonora esta esparcida uniformemente por todo el cuarto, y las ondas sonoras viajan en todas direcciones, entonces, el nivel de presión sonora en estado estacionario será directamente proporcional a la potencia acústica de salida de las fuentes de sonido e inversamente proporcional a la absorción total del cuarto. De manera similar, cuando la fuente de sonido ha cesado, un tiempo digno de atención habrá pasado antes de que el sonido muera (decaiga) y no se escuche. Esta propagación del sonido como

resultado de una sucesión de reflexiones en un espacio cerrado después de que la fuente de sonido es apagada, se llama reverberación. Las condiciones desfavorables que prevalecen en recintos altamente reverberantes (principalmente en las iglesias) son bien conocidas por todos. La inteligibilidad del lenguaje se reduce en estos lugares. La importancia del control de la reverberación en el diseño acústico ha requerido la introducción de una medida estándar relevante: el tiempo de reverberación. Es el tiempo para el que el nivel de presión sonora en un cuarto decrece 60 dB después de que la fuente se ha apagado.

2.2.5 Tiempo de reverberación

W. C. Sabine fue el primero que estableció cuantitativamente la relación entre el tiempo de reverberación, el volumen del cuarto y la cantidad de absorción total aplicada en las paredes del cuarto. La formula de Sabine para el cálculo del tiempo de reverberación es:

$$R.T. = 0.16\frac{V}{A} \tag{2.7}$$

Donde R.T. es el tiempo de reverberación en segundos, 0.16 es una constante con unidades [s/m], V es el volumen en m^3 y A es la absorción total en m^2

La absorción de la superficie se obtiene multiplicando su área por su coeficiente de absorción α, y la absorción total A se calcula sumando estos productos.

Como el coeficiente de absorción de varios materiales y acabados usados en el diseño de un auditorio varía con la frecuencia, naturalmente el tiempo de reverberación varía con la frecuencia.

La Figura (2.5) indica los tiempos de reverberación considerados deseables para auditorios de varios tamaños y funciones.

Aunque después de la definición del tiempo de reverberación se han ido descubriendo otros parámetros que describen la calidad acústica de un recinto, el tiempo de reverberación sigue siendo el parámetro más importante para la evaluación de cualquier sala.

2.3 Evaluación de Recintos.

Hace varios años el único criterio que se tomaba en cuenta para definir la acústica en un recinto era el tiempo de reverberación. Posteriormente, varios estudios revelaron que diferentes salas a pesar tener el mismo tiempo de reverberación se escuchaban de diferente manera. Por lo que existen otros factores que definen las características acústicas de una sala.

Desde la primera definición del tiempo de reverberación por Sabine, ciertos parámetros se han desarrollado para describir el comportamiento de una sala.

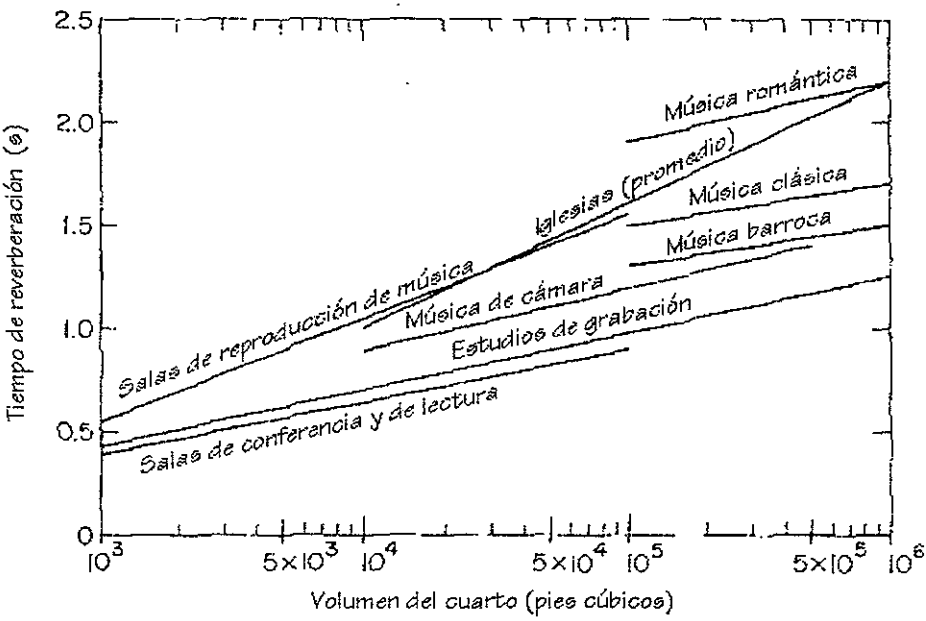


Figura 2.5: Tiempo de reverberación para diferentes salas.

2.3.1 Sonido directo, temprano y reverberante.

Para comprender como se propaga el sonido dentro de un cuarto, describiremos la experiencia sonora total resultante de tocar una sola nota musical. Considérese primero un sonido percusivo extremadamente corto. La trayectoria más corta posible de la fuente al escucha es una línea recta, y el **sonido directo** viajando por esa trayectoria llega primero. Seguida inmediatamente por varias reflexiones del sonido de las paredes o del techo; la distancia recorrida por cada una de las trayectorias de reflexión determina que tan atrasadas llegarán esas reflexiones. Los sonidos que llegan aproximadamente de 50 a 100 ms después del sonido directo se califican como **reflexiones tempranas**. Posteriormente, se continua recibiendo sonido de más y más diferentes trayectorias de reflexión. Cada componente individual es más débil y débil en tanto el número de reflexiones crece, y todas se unen en un **sonido reverberante** continuamente decayendo, que es percibido como una prolongación y un decaimiento gradual del evento original. En la Figura (2.6) se muestran estas reflexiones llegando con varios tiempos de retraso t_1, t_2 y t_3, etc.

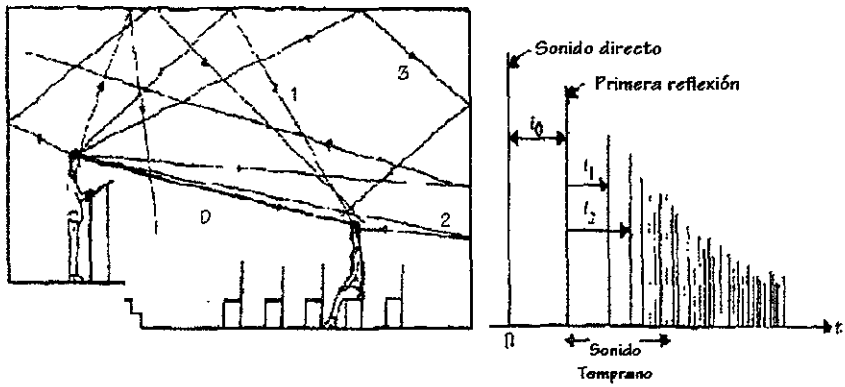


Figura 2.6: Trayectorias del sonido directo y reflejado de la fuente (D representa el sonido directo y 1, 2 y 3 son la reflexiones tempranas) al escucha con sus correspondientes retrasos en el tiempo.

2.3.2 Parámetros subjetivos que definen la calidad de un recinto.

Existen diferentes términos para definir la acústica en un recinto, Beranek [1] encontró 18 atributos subjetivos de calidad de acústica musical que pueden relacionarse a la acústica de recintos. Entre los más importantes están los siguientes:

Claridad. La claridad se obtiene cuando el nivel de sonido temprano más el sonido directo es más grande que el nivel de sonido reverberante en todo lugar. Cada nota deberá llegar limpia, vigorosa y sin oscuridad. Esta característica es de especial importancia si el cuarto es usado para conferencias así como para música, porque la inteligibilidad de las palabras depende directamente en la claridad de la articulación.

Uniformidad. Los espectadores en todas las partes de la sala deberán escuchar tan cerca el mismo sonido como sea posible, no deberá haber sitios muertos.

Envolvimiento. El escucha no deberá sentirse separado de la fuente de sonido mas bien bañado de sonido por todos lados; al mismo tiempo el sonido deberá ser identificable en el escenario para empatar el sentido del sonido con lo que se ve.

Ausencia de eco. Aunque hay repetidas reflexiones del sonido en las paredes, ninguna de estas deberá percibirse como un eco separado; todas las reflexiones deberán mezclarse.

Reverberación. Esta continuación, o caída del sonido en un cuarto, deberá tener una sonoridad relativa apropiada al sonido original.

Satisfacción del ejecutante El escenario también deberá estar libre de ecos distractores y que los ejecutantes en un grupo puedan sentir que están en una buena comunicación uno con otro.

Libre de ruido. Los pasajes suaves en la música no deberán ser perturbados por ruido de tráfico o por el ruido del sistema de ventilación del auditorio.

Intimidad. Un recinto tiene intimidad acústica cuando la música suena como si fuera tocada en un cuarto pequeño. El tiempo de retraso entre el sonido directo y el reflejado debe ser menor a 20 ms para que la sala sea íntima.

Vida. Se relaciona principalmente con el tiempo de reverberación para frecuencias medias y altas. El tiempo de reverberación depende del tamaño y la función. Una sala con tiempo de reverberación corto se determina como "seca".

Calidez. La calidez se obtiene cuando el tiempo de reverberación para bajas frecuencias es mayor que el tiempo de reverberación para altas frecuencias. Si el tiempo de reverberación es muy grande para bajas frecuencias, el sonido se puede volver "turbio" y carecerá de claridad.

Conjunción. Deberán ser superficies de reflexión amplias a los lados y arriba de la orquesta para que los músicos puedan escucharse unos a otros.

Impresión espacial. Dependen de las contribuciones del sonido temprano y las reflexiones laterales.

Tiempo de decaimiento temprano. La relación inicial del decaimiento del sonido reverberante parece ser más importante que el tiempo de reverberación total. Un decaimiento inicial rápido es aparentemente interpretado por el oído como si el tiempo de reverberación fuera corto.

Nivel de sonido reverberante. El nivel de sonido reverberante, que será el mismo a través del recinto, depende de la potencia de la fuente y el tiempo de reverberación.

2.3.3 Índices Acústicos.

Existen ciertos índices acústicos definidos por relaciones matemáticas, con los cuales nos es posible calificar de manera objetiva la calidad acústica de un recinto; varios índices acústicos están intimamente relacionados con algunos parámetros subjetivos de evaluación. A continuación describiremos varios de estos índices acústicos.

Tiempo de Reverberación (RT).

El tiempo de reverberación es aún una medición fundamental en el diseño de una sala de conciertos aunque en años recientes lo concerniente al diseño se ha extendido no sólo al tiempo de reverberación sino a otros atributos de la sala.

El tiempo de reverberación óptimo es entonces, un compromiso entre la claridad (que requiere un tiempo de reverberación corto), intensidad de sonido (que necesita un nivel reverberante alto), y vida (requiriendo tiempos de reverberación altos). El tiempo de reverberación óptimo dependerá del tamaño del auditorio y el uso para el que esté diseñado. Un recinto para conferencias deberá tener un tiempo de reverberación más corto que uno para música.

Una manera de calcular el tiempo de reverberación es ajustando una porción de la curva de decaimiento [15, 16] de entre -5 y -35 dB (o -25 dB cuando el rango dinámico es insuficiente) a una recta y extrapolando para un decaimiento de -60 dB.

Tiempo de Decaimiento Temprano. (EDT)

El tiempo de decaimiento temprano (EDT) es uno de los criterios más importantes en la evaluación de un recinto, seguido de la Eficiencia lateral y la Claridad. El EDT no deberá diferir del tiempo de reverberación de Sabine más del $\pm 10\%$, para salas de conciertos deberán buscarse valores entre 1.8 y 2.3 s. El tiempo de decaimiento se mide con los primeros 10 dB de energía disminuida, equivalentes a la pendiente de la curva de energía medida dentro de los primeros 500 ms. Como la EDT es sensible a la geometría del cuarto, en particular para reflexiones tempranas fuertes para reforzar el sonido en los primeros 100 ms, la EDT variará con la localización alrededor de la sala.

El EDT propuesto por Jordan se calcula a partir de las curvas de decaimiento pero se toma un intervalo comprendido entre los 0 a -10 dB.

Definición D50.

Este parámetro fue presentado por Thiele, y es la relación de la energía temprana con un tiempo límite de 50 ms y la energía total, comúnmente se expresa como porcentaje

$$D50 = \frac{\int_0^{50ms} h^2(t)dt}{\int_0^{\infty} h^2(t)dt} \tag{2.8}$$

Relación Señal a Ruido (S/N).

La relación señal a ruido descrita por Lochner y Burger expresada en dB, es otra relación entre energía temprana y la energía tardía:

$$S/N = 10 \cdot \log_{10}\left[\frac{\int_0^{95ms} a(t)h^2(t)dt}{\int_{95ms}^{\infty} h^2(t)dt}\right] \tag{2.9}$$

Con un factor de peso $a(t)$ descrita por la siguiente función:

$$a(t) = \begin{cases} 1 & 0 \leq t \leq 35ms \\ -\frac{1}{60}(t-95) & 35 \leq t \leq 95ms \\ 0 & t \geq 95ms \end{cases} \tag{2.10}$$

Claridad C80.

Indice propuesto por Reichardt es una medida de relación,expresada en dB, de energía temprana (antes de 80 ms) y la energía tardía (posterior a 80 ms):

$$C80 = 10 \cdot \log_{10}\left[\frac{\int_0^{50ms} h^2(t)dt}{\int_0^{\infty} h^2(t)dt}\right] \tag{2.11}$$

Tiempo Central Tc.

Propuesto por Cremer, tiene la ventaja de no asumir cualquier limite arbitrario entre las reflexiones tempranas y tardías como en C80 y D50:

$$Tc = \frac{\int_0^{\infty} t \cdot h^2(t)dt}{\int_0^{\infty} h^2(t)dt} \tag{2.12}$$

Indica el momento de primer orden del cuadrado de la respuesta al impulso en nivel de presión, expresada en milisegundos.

Intervalo de Tiempo Inicial (ITDG)

Definido por Beranek, es el retraso de la primera reflexión de la onda directa, expresada en milisegundos. Se calcula directamente de la repuesta al impulso.

Los siguientes índices involucran relaciones no sólo de los niveles de energía en determinados intervalos de tiempo. El trabajo desarrollado únicamente se encargará de los índices descritos en los párrafos anteriores y los demás sólo serán mencionados.

Correlación Interauricular (IACC).

Sugerida por Ando, es el coeficiente de correlación normalizado entre los primeros 50 ms de la repuesta al impulso medida en los dos oídos de un micrófono binaural.

Fuerza G.

Es la diferencia entre el nivel de presión sonora medido, y el producido por la misma fuente omnidireccional en el espacio libre a 10 m de distancia de su centro, y se expresa en decibeles. Y es definida por la siguiente ecuación

$$G = 10 \cdot \log_{10} \left[\frac{\int_0^\infty h^2(t)dt}{\int_0^\infty h_{10}^2(t)dt} \right] \tag{2.13}$$

Speech Transmission Index STI. (Índice de transmisión del lenguaje.)

Se usa como parámetro para la inteligibilidad del lenguaje, y tiene sentido para teatros, o salas de opera, donde la inteligibilidad de los actores y los cantantes es un factor importante.

Eficiencia lateral LE.

Es un parámetro que se mide con micrófonos bidireccionales. Jordan definió la eficiencia lateral como la relación de la energía que llega entre el 5 ms y 80 ms medida por un micrófono de "figura de ocho" y la energía temprana (que llega antes de los 80 ms), medida por un micrófono omnidireccional:

$$LE = \frac{\int_5^{80ms} h^2(t)dt}{\int_0^{80ms} h_8^2(t)dt} \tag{2.14}$$

Como la Definición (D50), LE comúnmente se expresa como un porcentaje.

Función de levantamiento de energía ERF(τ)

La historia del tiempo de la energía sonora integrada definida por

$$ERF(\tau) = \int_0^\tau h^2[t]dt \tag{2.15}$$

$L_{\tau x}$

El nivel de la función de levantamiento de energía sonora a τx ms definido por

$$L_{\tau x} = 10 \log \left(\frac{ERF(\tau x)}{4 \times 10^{-10}} \right) \text{ dB} \tag{2.16}$$

Respuesta del cuarto RR.

$$RR \sim 10 \log \frac{energía\ lateral(25-80)\ ms\ +\ energía\ total(80-160)\ ms}{energía\ total(0-80)\ ms} \tag{2.17}$$

2.3.4 Técnicas de Medición de la Respuesta al Impulso en Recintos.

Si consideramos una trayectoria de transmisión en un cuarto, extendiéndose desde una fuente sonora a un punto receptor, como un sistema lineal, observamos que todas las técnicas de medición pueden reducirse a la medición de su respuesta al impulso. En la acústica de recintos, la medición más importante, tal vez después del tiempo de reverberación es la de la respuesta al impulso de una sala, ya sea para conciertos, conferencias, etc. De la respuesta al impulso podemos evaluar parámetros acústicos importantes tales como, el tiempo de reverberación o la definición. Estas evaluaciones se realizan de manera efectiva con la computadora. De hecho, el procesamiento digital de las señales medidas se ha aplicado a la acústica de recintos en los últimos años y sigue desarrollándose.

Los métodos de obtención de la respuesta al impulso se basan en el mismo principio:

- Excitación de la sala con una fuente de sonido;
- Grabación de la respuesta del recinto;
- Cálculo de la respuesta al impulso;

Es posible clasificar las técnicas de medición de la respuesta al impulso, de acuerdo a la señal de excitación empleada, en dos grupos:

1. Aquellas que emplean señales impulsivas tales como, disparos de una pistola, pulsos de corta duración, estallido de globos, chispas eléctricas, etc. Estas señales tienen una relación señal a ruido baja y presentan problemas de reproducción.
2. Uso de ruido aleatorio: ruido blanco, secuencias de máxima longitud (MLS) y barrido senoidal. La propiedad que poseen las dos primeras señales es que su autocorrelación es casi un impulso.

Al tener como entrada señales impulsivas, por definición, se tiene directamente la respuesta la repuesta al impulso. Aunque tienen la desventaja de presentar relaciones de señal a ruido bajas y problemas de repetibilidad.

En el caso de tener ruido aleatorio, se requiere procesar la señal de salida $y[n]$, es decir, la respuesta del recinto. Sabiendo que la correlación $\Phi_{xy}[n]$ de la señal de entrada $x[n]$ y la salida $y[n]$, para un sistema lineal e invariante con el tiempo, se obtiene al convolucionar la autocorrelación de la señal de entrada $\Phi_{xx}[n]$ con la respuesta al impulso $h[n]$ (Sección 1.3.1). Esto es,

$$\Phi_{xy}[n] = \Phi_{xx}[n] * h[n] \tag{2.18}$$

Y basándose en el hecho de que la autocorrelación de señal de ruido es casi un impulso; para el caso de tener ruido blanco y secuencias de máxima longitud, como señal de excitación

$$\Phi_{xx}[n] \cong \delta[n] \tag{2.19}$$

Por lo tanto, la respuesta al impulso es igual a la correlación de la señal de entrada con la señal de salida.

$$\Phi_{xy}[n] \cong h[n] \tag{2.20}$$

Para las secuencias de máxima longitud, el cálculo de la correlación se efectúa por medio de la Transformada rápida de Hadamard. Este método resulta bastante eficiente en lo que se refiere al tiempo de procesamiento. La ventaja de estas secuencias no sólo radica en el hecho de tener un algoritmo muy rápido para el cálculo de la correlación, sino también varios autores [16, 3, 5, 14] han mostrado que las mediciones con secuencias de máxima longitud son prácticas, puesto que se generan fácilmente y generalmente proveen de una alta inmudad al ruido (una relación de señal a ruido igual a $10 \log_{10} \textit{longitud de la secuencia}$). Por estas razones y por ser la técnica más moderna desarrollada hasta estos días, se decidió implementarla. En el siguiente Capítulo se describirán con más detalle lo referente a estas secuencias y a la Transformada de Hadamard.

Es importante mencionar que calcular la correlación no es la única forma para obtener la respuesta al impulso. Otra forma sería trasladar la señal de entrada y la señal de salida al dominio de la frecuencia, dividir la señal de salida con la señal de entrada, y regresar al dominio del tiempo para tener la respuesta al impulso. Cosa que resulta un poco laboriosa.

Observaciones en la instrumentación y en los procedimientos de medición.

A fin de obtener respuestas a impulso con una relación de señal a ruido grande y un espectro rico, ciertas condiciones referentes a la señal de la fuente se deben cumplir:

- La señal generada debe contener suficiente energía de manera que se tengan altas relaciones de señal a ruido en todas las componentes de frecuencia de interés.
- Además, deberá abarcar todas las frecuencias de interés para tener un espectro rico.
- La señal debe ser repetible a lo largo del tiempo y el espacio, i. e. ,

1. La señal de tiempo generada debe ser repetible.
2. Para la mayoría de los propósitos de medición, la fuente deberá radiar el sonido de manera uniforme en todas direcciones. Para mediciones de la reverberación los requerimientos referentes a la omnidireccionalidad de la radiación del sonido no son tan estrictos, ya que en cualquier caso los distintos componentes del sonido se mezclaran en el transcurso del tiempo.

En cuanto al detector; generalmente se emplean micrófonos sensibles a la presión con características omnidireccionales como receptores de sonido. También el detector deberá tener una respuesta en frecuencia que abarque el ancho de banda de interés.

Para evitar la influencia de ruido en la medición se deberán tener factores pico [4] (relación entre el valor eficaz y el valor pico de la señal) bajos para la excitación. Y repetir varias veces la medición. Es importante mencionar que si esta presente el ruido de la línea éste no se podrá eliminar promediando, se tendrá que usar un dispositivo externo que lo elimine.

En el caso de trabajar con señales digitales, la duración de la respuesta al impulso es definida por el tiempo de reverberación. Por ejemplo, el tiempo de reverberación para salas de conciertos va del orden de 1 a 2 segundos. A fin de trabajar con el rango audible de frecuencias (20 Hz a 20 kHz) y por el teorema de muestreo, debemos tener tasas de muestreo de al menos dos veces la frecuencia máxima en la que se va a trabajar. Es decir, frecuencias de muestreo de al menos 40 kHz.

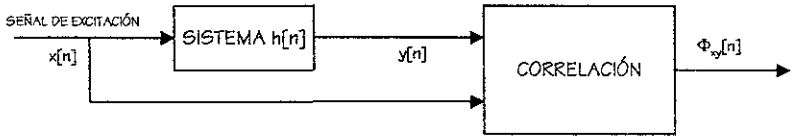
Capítulo 3

Medición de la Respuesta al Impulso usando secuencias ML

Uno de los métodos modernos para la obtención de la respuesta al impulso es el propuesto por Schroeder [16]. En él se emplean secuencias de máxima longitud como señal de excitación; su característica principal es que la autocorrelación de estas señales es aproximadamente una señal delta. Si tomamos en cuenta esta propiedad y de acuerdo con la teoría de los sistemas dinámicos lineales, la correlación de entrada-salida $\Phi_{xy}[n]$ de un sistema lineal invariante con el tiempo se define como

$$\Phi_{xy}[n] = \Phi_{xx} * h[n] \tag{3.1}$$

Donde $\Phi_{xx}[n]$ corresponde a la autocorrelación de la señal de entrada $x[n]$, $h[n]$ es la respuesta al impulso y $*$ denota la operación de convolución. Ahora bien, si $\Phi_{xx}[n]$ es igual a una señal delta tenemos que la respuesta al impulso es proporcional a la función de correlación entre la entrada y la salida $\Phi_{xy}[n]$ Figura (3.1)



$$\phi_{xy}[n] = \phi_{xx}[n] * h[n] \cong \sum_{k=0}^{N-1} x(k-n)y(k)$$

Figura 3.1: Método para medir la respuesta al impulso a partir de la función de correlación entrada-salida

La técnica propuesta por Schroeder involucra trabajar con secuencias ML [1] como señal de excitación. Posteriormente, procesar la señal de salida $y[n]$ medida calculando la correlación de entrada-salida por medio de la transformada de Hadamard [3, 5]; método adaptado de la espectroscopia de Hadamard, que resulta ser muy rápido.

De manera que surgen tres incógnitas:

1. ¿Cómo se producen las secuencias ML?
2. ¿Cuál es el procedimiento de cálculo para la Transformada de Hadamard? y
3. ¿Cómo se reduce la correlación de las secuencias ML con la señal de salida a una Transformada rápida de Hadamard?

En las siguientes secciones se tratarán de resolver estas preguntas.

3.1 Secuencias de Máxima longitud (MLS).

Las secuencias de máxima longitud (Maximal Length Sequence) que también son llamadas secuencias de pseudoruido (PN) o secuencias m, son secuencias periódicas binarias [11] de longitud $2^m - 1$ con una relación señal a ruido igual a $10\log_{10}(2^m - 1)$.

Antes que nada, es necesario hacer un breve paréntesis para explicar algunos conceptos de la aritmética modular que se utilizan para la generación de las secuencias ML.

3.1.1 Aritmética Modular.

Aritmética modular entera.

La aritmética modular consiste, grosso modo, en calcular el residuo del cociente entre dos números [17].

Cuando c se divide por m y deja el residuo b se escribe

$$c \equiv b \pmod{m}; \tag{3.2}$$

c es congruente a $b \pmod{m}$.

De forma más general, la congruencia anterior se define como sigue

$$(c - b)/m, \tag{3.3}$$

m divide a c menos b (sin residuo); o bien

$$c = qm + b, \tag{3.4}$$

[1] Abrev. ML. Máxima Longitud.

con q entero.

Otra forma de calcular esta equivalencia es restar m a c hasta que el resultado b sea menor a m. Lo cual es otra manera de interpretar un cociente. Por ejemplo,

$$16 \equiv ? \pmod 3, \tag{3.5}$$

16 dividido por 3 deja como residuo 1. Esto es porque

$$\begin{aligned} 16 - 3 &= 13 \\ 13 - 3 &= 10 \\ 10 - 3 &= 7 \\ 4 - 3 &= 1 \end{aligned}$$

y entonces $16 \equiv 1 \pmod 3$.

Tal vez sea más laborioso realizar una serie de sustracciones, pero para el caso de la aritmética modular entre polinomios es más ventajoso trabajar con restas entre polinomios que divisiones.

Una regla muy útil es la siguiente:

$$mc \equiv mb \pmod n \tag{3.6}$$

Ejemplo:

$$28 \equiv 4 \pmod 6$$

pero

$$28 = 7 \times 4$$

$$7 \equiv 1 \pmod 6$$

por lo tanto

$$7 \times 4 \equiv 1 \times 4 \pmod 6$$

Factorización de polinomios.

Un polinomio $p(x)$ de grado n sobre un campo F se define como

$$p(x) = \sum_{k=0}^{n} a_k x^k; \tag{3.7}$$

donde los coeficientes a_k son elementos del campo de números F. Por ejemplo, el campo de los números complejos C, o el campo de los números racionales Q. También se puede tratar de algún campo finito como $GF(2)$, el campo de Galois de orden 2 que consiste típicamente de los elementos 0 y 1; las operaciones en este tipo de campo se llevan a cabo con aritmética módulo el orden del campo, en este caso $\pmod 2$.

Un polinomio irreducible no puede ser representado por polinomios de grado menor. Por ejemplo, el polinomio $x^2 + 1$ es irreducible en Q pero en el campo de los números complejos se reduce a la siguiente expresión

$$x^2 + 1 = (x + i)(x - i).$$

Este polinomio también es reducible sobre el campo finito $GF(2)$

$$x^2 + 1 = (x + 1)^2 = x^2 + x + x + 1,$$

porque $1 + 1 = 2 \pmod 2 = 0$ en $GF(2)$.

De tal manera, el campo elegido determinará la posible factorización del polinomio.

Aritmética modular de polinomios.

Dos polinomios $p_1(z)$ y $p_2(z)$ se dice que son congruentes módulo $d(z)$ si su diferencia es divisible entre $d(z)$. Se escribe

$$p_1(z) \equiv p_2(z) \pmod{d(z)}. \tag{3.8}$$

De forma equivalente

$$\frac{p_1(z) - p_2(z)}{d(z)}, \tag{3.9}$$

o bien

$$p_1(z) = q(z)d(z) + p_2(z) \tag{3.10}$$

Ejemplo: Considere la congruencia módulo $(x^2 - 2)$ en Q.

Podemos decir que

$$a(x) \equiv b(x) \pmod{x^2 - 2}$$

si $x^2 - 2$ divide a $a(x) - b(x)$, o si

$$a(x) = b(x) + q(x)(x^2 - 2)$$

para algún polinomio $q(x)$ en Q. Entonces

$$x^2 \equiv? \pmod{x^2 - 2}$$

restamos $x^2 - 2$ a x^2

$$x^2 - (x^2 - 2) = 2$$

por lo tanto

$$x^2 \equiv 2 \pmod{x^2 - 2}$$

3.1.2 Generación de secuencias ML

Para construir secuencias ML de longitud $n = 2^m - 1$ se requieren ciertas ecuaciones de recurrencia que se obtienen a partir de una serie de multiplicaciones entre polinomios de grado menor a m, módulo un polinomio primitivo $h(x)$ de grado m. Los coeficientes de estos polinomios pertenecen a $GF(2)$. En la Tabla (3.1) se proporcionan los polinomios primitivos de grado 1 al 40.

Grado m	h(x)	*Grado m*	h(x)
1	$x+1$	21	$x^{21}+x^2+1$
2	x^2+x+1	22	$x^{22}+x+1$
3	x^3+x+1	23	$x^{23}+x^5+x+1$
4	x^4+x+1	24	$x^{24}+x^4+x^3+x+1$
5	x^5+x^2+1	25	$x^{25}+x^3+1$
6	x^6+x+1	26	$x^{26}+x^8+x^7+x+1$
7	x^7+x+1	27	$x^{27}+x^8+x^7+x+1$
8	$x^8+x^6+x^5+x+1$	28	$x^{28}+x^3+1$
9	x^9+x^4+1	29	$x^{29}+x^2+1$
10	$x^{10}+x^3+1$	30	$x^{30}+x^{16}+x^{15}+x+1$
11	$x^{11}+x^2+1$	31	$x^{31}+x^3+1$
12	$x^{12}+x^7+x^4+x^3+1$	32	$x^{32}+x^{28}+x^{27}+x+1$
13	$x^{13}+x^4+x^3+x+1$	33	$x^{33}+x^{13}+1$
14	$x^{14}+x^{12}+x^{11}+x+1$	34	$x^{34}+x^{15}+x^{14}+x+1$
15	$x^{15}+x+1$	35	$x^{35}+x^2+1$
16	$x^{16}+x^5+x^3+x^2+1$	36	$x^{36}+x^11+1$
17	$x^{17}+x^3+1$	37	$x^{37}+x^{12}+x^{10}+x^2+1$
18	$x^{18}+x^7+1$	38	$x^{38}+x^6+x^5+x+1$
19	$x^{19}+x^6+x^5+x+1$	39	$x^{39}+x^4+1$
20	$x^{20}+x^3+1$	40	$x^{40}+x^{21}+x^{19}+x^2+1$

Tabla 3.1: Polinomios primitivos de orden 1 al 40

Por ejemplo, para m=3

$$h(x) = x^3 + x + 1,$$

y de forma general

$$a + bx + cx^2 \tag{3.11}$$

Determinemos la siguiente congruencia

$$x^3 \equiv? \pmod{x^3 + x + 1}).$$

Sumamos a x^3 el polinomio irreducible para obtener la congruencia

$$x^3 + (x^3 + x + 1),$$

pero como $1+1=2 \pmod 2 = 0$ entonces,

$$x^3 \equiv x+1 \pmod{x^3+x+1}.$$

Si multiplicamos a la ecuación (3.11) por x tenemos

$$ax + bx^2 + cx^3,$$

y como $x^3 \equiv x+1$, entonces

$$c + (a+c)x + bx^2. \tag{3.12}$$

Volviendo a multiplicar por x

$$cx + (a+c)x^2 + b(x+1) = b + (b+c)x + (a+c)x^2; \tag{3.13}$$

y si multiplicamos por x una vez más

$$bx + (b+c)x^2 + (a+c)(x+1) = (a+c) + (b+a+c)x + (b+c)x^2. \tag{3.14}$$

Renombrando los coeficientes de los polinomios de las ecs. (3.11),(3.12),(3.13) y (3.14) se producen ciertas relaciones entre los coeficientes del mismo orden que son bastante interesantes. Los resultados se proporcionan en la Tabla (3.2)

a	bx	cx^2
$a = a_0$	$b = b_0$	$c = c_0$
$c = a_1$	$a + c = b_1$	$b = c_1$
$b = a_2$	$b + c = b_2$	$a + c = b_2$
$a + c = a_3 = a_0 + a_1$	$b + (a + c) = a_3 = b_0 + b_1$	$b + c = c_3 = c_0 + c_1$

Tabla 3.2:

Si proseguimos multiplicando el polinomio resultante por x nos daremos cuenta que se cumple la siguiente ecuacion de recurrencia para los coeficientes del polinomio de la forma $a + bx + cx^2$

$$a_{i+3} = a_i + a_{i+1}, \tag{3.15}$$

donde + indica adición $\pmod 2$.

La recursión anterior nos muestra que los elementos próximos se calculan con los valores pasados. Por lo que es necesario almacenar los $m-1$ valores anteriores.

Una vez que son determinados los $m-1$ elementos iniciales, la secuencia se determina únicamente por la recursión. Una forma de implementar físicamente un generador de secuencias sería por medio de un arreglo de registros de corrimiento, como se muestra en la Figura (3.2).

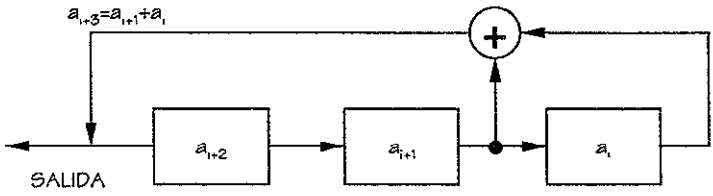


Figura 3.2: registro de corrimiento

En general, estos registros de corrimiento consisten de elementos de memoria o flip-flops, almacenando un 0 o 1. Cada unidad de tiempo el contenido de los flip-flops se corren un lugar hacia la derecha, y según lo determine la ecuacion de recurrencia, en este caso, $a_{i+3} = a_{i+1} + a_i$, los términos anteriores se suman y el resultado es almacenado. La suma se calcula con aritmética (mod 2), o de manera equivalente, empleando la compuerta lógica de la *or* exclusiva cuya tabla de verdad es

E1	E2	S
0	0	0
0	1	1
1	0	1
1	1	0

El registro de corrimiento requiere de condiciones iniciales, por lo tanto se deben especificar los valores iniciales a_0, a_1,a_2,... a_{m-1}. Como cada uno de los m elementos de memoria contiene un 0 o un 1, hay 2^m posibles estados del registro de corrimiento. Así, la secuencia $a_0\ a_1\ a_2\ldots$ debe ser periódica. Pero el estado 000...0 no puede ocurrir a menos que la secuencia sea de ceros. Por lo que el periodo máximo posible es $2^m - 1$.

3.1.3 Propiedades.

En el siguiente apartado se mencionarán algunas propiedades que presentan las secuencias ML que nos serán de utilidad para el desarrollo de esta tesis.

Si $h(x)$ es un polinomio primitivo de grado m y δ_m la serie de secuencias obtenidas de $h(x)$. Estas secuencias pseudoaleatorias son los $2^m - 1$ segmentos

$$a_i a_{i+1} \ldots a_{i+2^m-2}, i = 0, 1, \ldots 2^m - 2 \textit{ de longitud } 2^m - 1.$$

Para $m = 3$ y $x^3 + x + 1$ δ_m sería:

$$\delta_m = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Propiedad de Recurrencia.

Supóngase que

$$\sum_{i=0}^{m} h_i x^i \tag{3.16}$$

Con $h_0 = h_m = 1$, $h_i = 0$ o 1 para $0 < i < m$. Cualquier secuencia $b \in \delta_m$ satisface la recurrencia

$$h_m b_{i+m} = h_{m-1} b_{i+m-1} + h_{m-2} b_{i+m-2} + ... + h_1 b_{i+1} + h_0 b_i \tag{3.17}$$

para i=0,1,...

De modo recíproco, cualquier solución de (3.17) esta en δ_m. Empleando los $2^m - 1$ valores iniciales $b_0 b_1 ... b_{m-1}$ en (3.17) se obtienen las $2^m - 1$ secuencias m.

Función de Autocorrelación

La función de autocorrelación de una secuencia binaria $a_0\ a_1 \ldots a_{n-1}$ es igual a la función de autocorrelación de la secuencia real

$$(-1)^{a_0}, (-1)^{a_1}, ..., (-1)^{a_{n-1}},$$

obtenida reemplazando 1's por -1's y 0's por 1's. Entonces,

$$\rho(i) = \frac{1}{n} \sum_{j=0}^{n-1} (-1)^{a_j + a_{i+j}} \tag{3.18}$$

La función de autocorrelación de una secuencia ML de longitud $n = 2^m - 1$ esta dada por

$$\begin{aligned} &\rho(0) = 1 \\ &\rho(i) = -\tfrac{1}{n} \quad \text{para } 1 \leq i \leq 2^m - 2 \end{aligned} \tag{3.19}$$

Construcción de la matriz de Hadamard

Las secuencias m tienen una estrecha relación con la construcción de un tipo específico de matrices, llamadas matrices de Hadamard. Como veremos a continuación a partir de la matriz δ_m es posible generar este tipo de matrices.

Matriz de Hadamard. La matriz de Hadamard es una matriz real H_n de $n \times n$ de 1's y -1's que satisface la siguiente igualdad

$$H_n H_n^T = n\mathbf{I}_n, \tag{3.20}$$

donde la T determina la transposición de la matriz e I la matriz identidad.

Del arreglo δ_m se cambian los 0's por 1's y los 1's por -1's y se agrega una columna y un renglón de 1's. El arreglo de tamaño $2^m \times 2^m$ resultante es una matriz de Hadamard. La Figura (3.3) es una matriz de Hadamard obtenida a partir de la secuencia 0010111.

Existe un caso particular de este tipo de matrices denominadas de tipo Sylvester, que determinan la Transformada de Hadamard, como veremos en la Sección 3.3.1.

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

Figura 3.3: Matriz de Hadamard de 8 × 8

Polinomio recíproco.

El polinomio recíproco se obtiene a partir de la siguiente expresión:

$$\tilde{h}(x) = \sum_{k=0}^{m} h_k x^{m-k} \tag{3.21}$$

La característica principal de $\tilde{h}(x)$ es que genera secuencias inversas a $h(x)$. Es decir, si con $h(x) = x^3 + x + 1$ tenemos la secuencia 0010111. Entonces con $\tilde{h}(x) = x^3 + x^2 + 1$ obtenemos 1110100. La ventaja que presenta el trabajar con secuencias obtenidas de polinomios recíprocos, es la facilidad para la construcción de la matriz de corrimientos circulares a la derecha (matriz de correlación). Por esta razón, las secuencias usadas en este trabajo serán obtenidas a partir de polinomios recíprocos.

Resumen

Recapitulando lo anterior, una secuencia ML es una secuencia periódica binaria que tiene un espectro casi plano y su función de autocorrelación es aproximadamente un impulso.

Es generada por un arreglo de registros de corrimiento. La longitud de su periodo N es igual a $2^m - 1$, donde m es el número de estados del arreglo de registros de corrimiento.

Por ejemplo, tomando el polinomio recíproco de grado $m = 3$

$$x^3 + x^2 + 1.$$

Gracias a la propiedad de la recurrencia el polinomio se puede expresar como:

$$a_{i+3} = a_{i+2} + a_i$$

Con un registro de corrimiento como se muestra en la Figura(3.4)

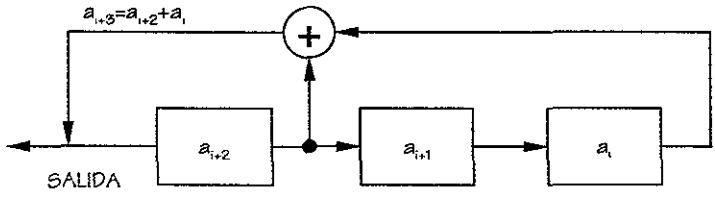


Figura 3.4: Registro de corrimiento.

Teniendo como condiciones iniciales para los registros $a_0 = a_1 = 0$, y $a_2 = 1$, se generan los siguientes estados dados en la Tabla(3.3):

a_{i+2}	a_{i+1}	a_i
1	0	0
1	1	0
1	1	1
0	1	1
1	0	1
0	1	0
0	0	1

Tabla 3.3: Estados del registro de corrimiento

La secuencia m es:

1110100

Para el procesamiento de la señal, los estados binarios de 0 y 1 de la secuencia tienen que ser mapeados dentro de los valores de ± 1 con la siguiente regla: se sustituirán 1 por -1 y 0 por $+1$. La secuencia mapeada será la que se aplique al sistema.

Una vez definida la generación de las secuencias ML, otro punto importante en esta técnica, es la función de correlación.

3.2 Función de Correlación.

La correlación de la señal de entrada $x[n]$ y la señal de salida $y[n]$ $\Phi_{xy}[n]$ en tiempo discreto para un sistema LTI se puede expresar como la autocorrelación de la entrada $\Phi_{xx}[n]$ convolución $*$ la respuesta al impulso $h[n]$:

$$\Phi_{xy}[n] = \Phi_{xx}[n] * h[n] \tag{3.22}$$

Cuando la autocorrelación de la entrada $\Phi_{xx}[n]$ es igual a $\delta[n]$, una función delta, el resultado de convolucionar $\Phi_{xx}[n]$ con cualquier función es la función misma, en este caso, la respuesta al impulso. De esta manera, la respuesta al impulso se puede recuperar correlacionando la entrada $x[n]$ con la salida $y[n]$. La operación de correlación se define como:

$$\Phi_{xy}[n] = \sum_{m=\infty}^{\infty} x[m]h[m+n]; \tag{3.23}$$

o de forma equivalente

$$\Phi_{xy}[n] = \sum_{m=\infty}^{\infty} x[m-n]h[m] \tag{3.24}$$

Como la señal de entrada $x[n]$ es una señal periódica, por lo tanto, la señal de salida $y[n]$ es periódica [17] entonces

$$x_n = x_{n+N},$$

donde N es el periodo.

De forma general, se puede escribir

$$\left.\begin{array}{l} x_n = x_m \\ y_n = y_m \end{array}\right\} para\ n \equiv m \pmod{n} \tag{3.25}$$

La función de correlación para una señal periódica o circular es

$$\Phi_{xy}[n] = \sum_{m=0}^{N-1} x\langle m-n\rangle_N y[m], \tag{3.26}$$

donde los corchetes $\langle\rangle_N$ indican $n-m$ módulo N.

La ecuación anterior puede describirse en términos de una multiplicación de matrices:

$$\Phi_{xy} = \mathbf{M}_N \mathbf{Y}, \tag{3.27}$$

donde Φ_{xy} y $\mathbf{Y}$ son vectores y la matriz $\mathbf{M}_N$ $(x\langle m-n\rangle)$ contiene las versiones de la secuencia con corrimientos hacia la derecha circulares.

Para el polinomio recíproco de grado 3 x^3+x^2+x+1 la matriz $\mathbf{M}_7$ es la siguiente:

$$\mathbf{M}_7 = \begin{bmatrix} -1 & -1 & -1 & 1 & -1 & 1 & 1 \\ 1 & -1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & 1 & -1 \\ -1 & 1 & 1 & -1 & -1 & -1 & 1 \\ 1 & -1 & 1 & 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

El cálculo de la correlación esta basado en el uso de la transformada rápida de Hadamard, la cual se puede calcular mediante un algoritmo bastante eficiente.

3.3 Procedimiento eficiente para el cálculo de la correlación.

Un método rápido para calcular la correlación entre dos señales es por medio de la Transformada rápida de Hadamard. Aunque para ello haya que realizar una serie de permutaciones que hagan equivalente la matriz de Hadamard a la matriz de corrimientos (correlación) de la secuencia.

3.3.1 Transformada Rápida de Hadamard

La transformada de Hadamard puede describirse en términos de una multiplicación de matrices. La matriz que transforma al vector se conoce como matriz de Hadamard $\mathbf{H}_n$, donde n es el número de columnas y renglones (matriz cuadrada). Los elementos de la matriz de Hadamard son ± 1, y la matriz debe satisfacer la relación:

$$\mathbf{H}_n\mathbf{H}_n^T = n\mathbf{I}_n$$

Este algoritmo sólo utiliza una clase específica de matrices de Hadamard conocidas como tipo Sylvester. Las matrices de Hadamard de tipo Sylvester son de orden 2^k donde k es un entero positivo. Tienen la forma (3.29) y se construyen a partir de matrices de orden menor $2^{k-1}(H_i)$. En general, son matrices que son definidas recursivamente. La matriz de menor orden es H_1 (ecuación 3.28).

$$\mathbf{H}_1 = [1] \tag{3.28}$$

En forma general, se tiene que

$$\mathbf{H}_{2i} = \begin{bmatrix} \mathbf{H}_i & \mathbf{H}_i \\ \mathbf{H}_i & -\mathbf{H}_i \end{bmatrix} \tag{3.29}$$

Si se quiere obtener la matriz $\mathbf{H}_2$ entonces $i = 1$

$$\mathbf{H}_2 = \begin{bmatrix} \mathbf{H}_1 & \mathbf{H}_1 \\ \mathbf{H}_1 & -\mathbf{H}_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Para $\mathbf{H}_4$, $i = 2$ de (3.29) tenemos

$$H_4 = \begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix} = \left[\begin{array}{cc:cc} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ \hdashline 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{array}\right]$$

Finalmente, para $\mathbf{H}_8$

$$H_8 = \begin{bmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{bmatrix} = \left[\begin{array}{cccc:cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 \\ \hdashline 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{array}\right]$$

Las líneas punteadas enfatizan la estructura que se desarrolla de una función recursiva.

La transformada de Hadamard se puede calcular si se realiza la evaluación directa de la multiplicación de la matriz de Hadamard por el vector que se desee transformar. Para el caso de un vector de 8 renglones sería

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} a & b & c & d & e & f & g & h \\ a & -b & c & -d & e & -f & g & -h \\ a & b & -c & -d & e & f & -g & -h \\ a & -b & -c & d & e & -f & -g & h \\ a & b & c & d & -e & -f & -g & -h \\ a & -b & c & -d & -e & f & -g & h \\ a & b & -1 & -d & -e & -f & g & h \\ a & -b & -1 & d & -e & f & g & -h \end{bmatrix}$$

Esta forma se llevaría 56 operaciones.

Si se empleara la transformada rápida de Hadamard lo haría en m pasos, donde m es el orden del polinomio primitivo. En este caso, m es igual a 3. La Figura (3.5) muestra una representación gráfica del algoritmo de la Transformada rápida de Hadamard. El número de operaciones que se realiza es sólo 24. La transformada de Hadamard requiere de $L \log_2 L$ en vez de $L(L-1)$ operaciones.

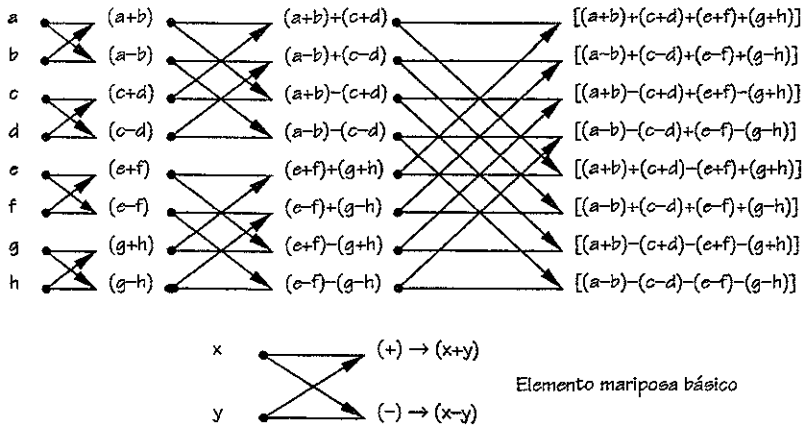


Figura 3.5: Diagrama de flujo para una Transformada rápida de Hadamard de orden 8

Como se mencionó anteriormente, para calcular la correlación se hará uso de la Transformada rápida de Hadamard que se obtiene mediante un algoritmo muy eficiente.

Para aplicar directamente la transformada rápida de Hadamard es necesario realizar una serie de permutaciones que permitan encontrar una equivalencia entre la matriz de corrimientos de la secuencia M y la matriz de Hadamard de tipo Sylvester de tal forma que

$$\mathbf{M} = \mathbf{PHQ}$$

donde $\mathbf{P}$ y $\mathbf{Q}$ son matrices de permutación que nos permiten aplicar la FHT [2] para obtener $\mathbf{M}_7\mathbf{Y}$ (la correlación de entrada-salida).

Cohn y Lempel [6] desarrollaron un método en el que no se construyen físicamente estas matrices, sino que a partir de ciertos índices obtenidos de la factorizacion de la matriz M se

[2] Fast Hadamard Transform. Trad. Transformada rápida de Hadamard

realizan los intercambios de los elementos del vector a transformar y es equivalente a haber realizado directamente el producto de estas matrices. Como se describirá a continuación.

Permutaciones

Excepto por la columna y renglón extra, la matriz $\mathbf{H}_8$ es similar a la matriz $\mathbf{M}_7$. La peculiaridad de esta técnica es que agregando una columna y renglón de $+1$ a la matriz $\mathbf{M}_7$, se pueden reordenar los renglones y columnas de la matriz aumentada para transformarla en $\mathbf{H}_8$. Conh y Lempel [6] han mostrado que cualquier matriz corrimientos de la secuencia ML $\mathbf{M}_m$ $(n \times n)$ puede ser factorizada en dos matrices $\mathbf{R}$ y $\mathbf{C}$. Una matriz $\mathbf{R}$ $(n \times m)$ y una matriz $\mathbf{C}$ $(m \times n)$. La matriz $\mathbf{C}$ se construye con los primeros m renglones de la matriz $\mathbf{M}_7$. La matriz $\mathbf{R}$ se forma a partir de las columnas de $\mathbf{M}_7$ de tal forma que los primeros m renglones constituyan una matriz identidad. La Figura (3.6) muestra las matrices $\mathbf{R}$ y $\mathbf{C}$ para una matriz de secuencias $\mathbf{M}_7$.

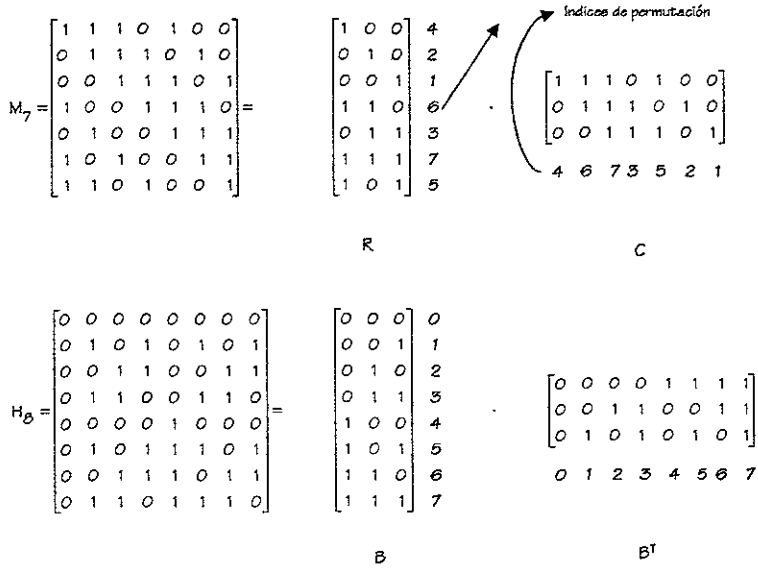


Figura 3.6: Factorización de la matriz de corrimientos y la matriz de tipo Sylvester. A lado de cada matriz factorizada están los índices de permutación correspondientes a cada renglón o columna

De forma similar, la matriz de Hadamard de tipo Sylvester $\mathbf{H}$ de $L \times L$ puede factorizarse en dos matrices: una matriz $\mathbf{B}$ de $L \times m$ y una matriz $\mathbf{B}^T$ de $m \times L$, donde $\mathbf{B}^T$ es la matriz transpuesta de $\mathbf{B}$. Los renglones $\mathbf{B}$ y las columnas de $\mathbf{B}^T$ son representaciones binarias de enteros. Un ejemplo de la matriz $\mathbf{H}_8$ también se muestra en la Figura (3.6). Si denotamos a los renglones de $\mathbf{R}$ y las columnas de $\mathbf{C}$ por medio de índices de acuerdo con su representación en decimal como se muestra en la Figura (3.6), es evidente que $\mathbf{M}_7$ y $\mathbf{H}_8$ contienen los mismos elementos pero en diferente orden. De manera que, usando los índices en las matrices $\mathbf{R}$ y $\mathbf{C}$ se puede transformar la matriz $\mathbf{M}_7$ en $\mathbf{H}_8$ en tres pasos:

1. Los índices de la matriz C reordenan las columnas de $\mathbf{M}_7$ (id est, mover la séptima columna de M, cuyo valor en decimal es 1, a la primera columna, la sexta columna a la segunda posición, etc.) para formar una matriz permutada $\mathbf{M}_7'$.

2. Los índices de la matriz R reordenan los renglones de la matriz $\mathbf{M}_7'$ (esto es, mover el tercer renglón a la primera posición, el quinto renglón a la tercera posición etc.).

3. Agregar una columna y un renglón de $+1$ a esta matriz para transformarla en $\mathbf{H}_8$. Los índices de permutación se generan con registros de corrimiento. Como describiremos a continuación.

Obtención de los índices de permutación.

Con la matriz $\mathbf{C}$, se obtienen los índices para reordenar las columnas (**permutación 1**). Ahora bien, si se transpone la matriz $\mathbf{C}$, se puede apreciar que los elementos de $\mathbf{C}$ corresponden a los diferentes estados por los que pasa el arreglo de registros de corrimiento para generar la secuencia ML (véase Figura (3.7)) que cumple con la siguiente recurrencia $a_{i+3} = a_{i+2} + a_i$. Entonces, estos índices se obtienen directamente de los estados del registro de corrimiento que estan generando la secuencia ML.

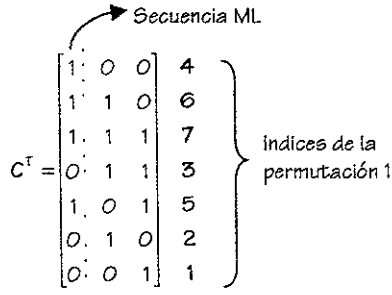


Figura 3.7: Matriz C transpuesta

Los índices que reordenan los renglones (**permutación 2**), que se extraen de la matriz **R**, se obtienen "invirtiendo" la forma del registro de corrimiento que genera la secuencia ML. Esto es, si tenemos el registro de corrimiento que produce la secuencia ML como se muestra en la Figura (3.8)

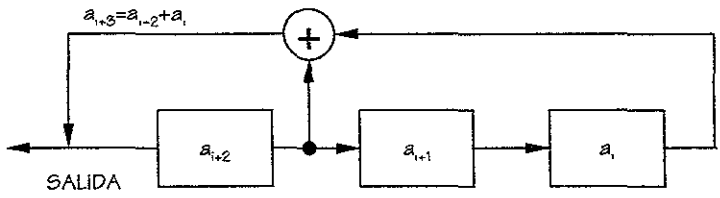


Figura 3.8: Registro de corrimiento para general secuencias de longitud

La inversión del registro de corrimiento quedaría (Figura 3.9)

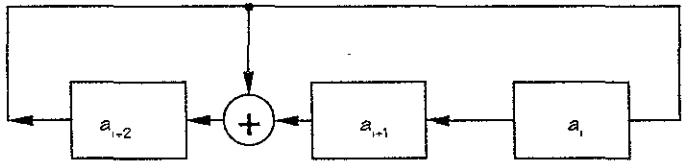


Figura 3.9: Registro de corrimiento invertido que calcula los índices de permutación 2

Para este caso, las condiciones iniciales del registro de corrimiento serán: $a_0 = 1$, $a_1 = 0$, y $a_2 = 0$. Como podemos observar son valores "contrarios" los de las condiciones iniciales para la **permutación 1**.

En los párrafos anteriores se describió la forma de transformar la matriz **M** en **H**. Con el único fin de ilustrar o ejemplificar la forma de transformación y poder a formular el procedimiento para calcular $[h]$ (respuesta al impulso) usando $\mathbf{H}_8$ (Transformada Rápida de Hadamard) en vez de $\mathbf{M}_7$.

Algoritmo para convertir H en M

1. Generar la matriz **M** .
2. Factorizar **M** en **R** y **C** .

3. Obtener los índices de los renglones de **R** y los índices de las columnas de **C** de acuerdo a la equivalencia entera de sus dígitos binarios de m bits.

4. Reordenar los elementos de **Y** usando los índices de las columnas de **C** y agregar un cero a la matriz permutada **Y'** para formar **Y''** (Figura 3.10).

Dado un vector Y

$$\mathbf{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \end{bmatrix} \quad \mathbf{Y'} = \begin{bmatrix} Y_7 \\ Y_6 \\ Y_4 \\ Y_1 \\ Y_5 \\ Y_2 \\ Y_3 \end{bmatrix} \quad \mathbf{Y''} \begin{bmatrix} 0 \\ Y_7 \\ Y_6 \\ Y_4 \\ Y_1 \\ Y_5 \\ Y_2 \\ Y_3 \end{bmatrix}$$

Figura 3.10: Vector Y permutado

5. Aplicar la Transformada Rápida de Hadamard a **Y''** para obtener **H''** (Figura 3.11).

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}}_{\mathbf{H_8}} \underbrace{\begin{bmatrix} 0 \\ Y_7 \\ Y_6 \\ Y_4 \\ Y_1 \\ Y_5 \\ Y_2 \\ Y_1 \end{bmatrix}}_{\mathbf{Y''}} = \underbrace{\begin{bmatrix} Y_7 & Y_6 & Y_4 & Y_1 & Y_5 & Y_2 & Y_3 \\ -Y_7 & Y_6 & -Y_4 & Y_1 & -Y_5 & Y_2 & -Y_3 \\ Y_7 & -Y_6 & -Y_4 & Y_1 & Y_5 & -Y_2 & -Y_3 \\ -Y_7 & -Y_6 & Y_4 & Y_1 & -Y_5 & -Y_2 & Y_3 \\ Y_7 & Y_6 & Y_4 & -Y_1 & -Y_5 & -Y_2 & -Y_3 \\ -Y_7 & Y_6 & -Y_4 & -Y_1 & Y_5 & -Y_2 & Y_3 \\ Y_7 & -Y_6 & -Y_4 & -Y_1 & -Y_5 & Y_2 & Y_3 \\ -Y_7 & -Y_6 & Y_4 & -Y_1 & Y_5 & Y_2 & -Y_3 \end{bmatrix}}_{h''}$$

Figura 3.11: *Producto de la matriz de Hadamard y el vector Y*

6. Omitir el primer elemento de **h''** y reordenar los elementos de la matriz resultante **h'**, usando los índices de los renglones de **R** para obtener **h**. Sin embargo, el proceso de reordenamiento será al revés. Los índices nos dicen que elemento debe de estar en la posición correspondiente al indice. Esto es, el cuarto elemento debe estar en la primera posición el segundo en la segunda posición el primer elemento en la tercera posición, etc.

En la practica, los pasos 1 y 2 del algoritmo anterior se omiten, ya que como se mencionó anteriormente, los índices de **permutación 1** se obtienen de los estados que se producen en el registro de corrimiento al generar la secuencia y los índices de **permutación 2** de los estados que genera el registro de corrimiento "invertido".

Capítulo 4

Implementación del método de obtención de la respuesta al impulso y cálculo de los índices acústicos

Una vez definida la técnica propuesta para la medición de la respuesta al impulso del recinto, daremos paso a la implementación de la misma. El trabajo desarrollado en este Capítulo constará de tres partes:

1. Medición de la respuesta del recinto . Un sistema de procesamiento digital de señales (LSI TMS320C30) se encargará de generar las secuencias ML (señal de excitación) y al mismo tiempo capturar la señal del micrófono (respuesta del recinto), como se verá en la Sección 4.1.

2. Obtención la respuesta al impulso del cuarto, es decir, correlacionando la señal de entrada con la señal de salida a través de un programa que calcula la Transformada rápida de Hadamard (Sección 4.2); y

3. Por ultimo, a partir de la respuesta al impulso calcular los índices acústicos con funciones desarrolladas en `Matlab` (Sección 4.3).

Así mismo, iremos ilustrando con un ejemplo práctico todas las fases del procedimiento, que van desde la medición de la respuesta del recinto hasta el cálculo de los índices acústicos.

Las mediciones se realizaron en un salón de clases ubicado en la Sección de Acústica del Centro de Instrumentos. Las condiciones de la medición fueron con el salón vacío y lleno.

4.1 Medición de la respuesta del recinto.

Para realizar cualquier medición en un cuarto es necesario, antes que nada, excitar al cuarto con una fuente sonora apropiada. En este caso, la señal de excitación serán las secuencias ML y para captar la respuesta del cuarto usaremos un micrófono. La señal medida por el micrófono será almacenada para posteriormente procesarla.

La generación de secuencias y la captura de la señal medida por el micrófono se llevará a cabo por medio de un sistema de tarjeta que contiene el procesador digital de señales **TMS320C30** de Texas Instruments.

En la siguiente Sección se describirán algunas características del hardware y software [19, 21] de este sistema de procesamiento.

4.1.1 Sistema de procesamiento LSI TMS320C30.

El procesador digital de señales (DSP) TMS320C30 (mejor conocido como C30) de Texas Instruments contiene unidades aritméticas enteras y de punto flotante, 2048x32 bit palabras en RAM, 4096x32 bit palabras en ROM, unidad de control e interfaces en serie y paralelo. Operando con reloj a 33.3 MHz, realizando 16.7 millones de instrucciones por segundo.

Mapa de memoria.

La memoria se divide en dos áreas denominadas A y B. El área A se divide en 3 secciones. Cada sección es de 64k palabras. Mientras que el área B cuenta con 16k palabras; de la dirección 30000h en adelante. La sección B es una memoria de tipo DUAL que se usa para transferir datos entre la PC y el DSP.

Interfaces analógicas.

La tarjeta del C30 tiene dos canales de interfaz A y B. Cada canal tiene una entrada y una salida de señales analógicas con sus respectivos convertidores analógicos-digitales y digitales-analógicos.

La conversión D/A o A/D se inicia, ya sea por una señal de disparo externa, o por un contador/temporizador interno.

El DSP cuenta con dos temporizadores internos. El temporizador 1 se utiliza para controlar el ritmo de muestreo. Consta de un contador progresivo de 32 bits y un registro de 32 bits que almacena el valor del periodo de muestreo. Cada 120ns, aproximadamente, el contador se va incrementando. Cuando el contador es igual al registro del periodo, genera un pulso que inicia la conversión D/A o A/D, y se reinicia el contador.

El manual de la tarjeta [19] sugiere manejar las conversiones D/A y A/D por medio de una rutina de interrupción.

Herramientas de programación del C30.

El software de la tarjeta del TMS320C30 tiene un compilador de lenguaje C, -lo que hace más sencilla la programación-, un ensamblador, un enlazador (linker), etc.

El compilador de C del C30 es un compilador que traduce programas en lenguaje ANSI-C a lenguaje ensamblador propio del DSP C30. Produce seis secciones de código relocalizables. Además, crea dos clases básicas de secciones: inicializadas y no inicializadas. La siguiente tabla resume estas secciones.

NOMBRE	TIPO	CONTENIDO
.text	Inicializada	Código ejecutable y cadena de literales
.cinit	Inicializada	Tabla de variables preinicializadas
.const	Inicializada	Cadena de constantes y tablas de switch
.bss	No Inicializada	Variables globales y estáticas
.stack	No Inicializada	Pila del sistema
.sysmem	No Inicializada	Localidades dinámicas usando malloc, calloc o realloc

Tabla 4.1: Secciones generadas por el compilador

Cuando se liga un programa, se debe especificar donde se van a localizar las secciones en la memoria. En general, las secciones inicializadas pueden ligarse dentro de memoria ROM o RAM. Las secciones no iniciallizadas deberán ser ligadas en memoria RAM.
El linker define el mapa de memoria y asigna código y datos dentro de la memoria.

4.1.2 Programa de medición de la respuesta del recinto.

Este programa estará encargado de generar las secuencias ML y simultáneamente almacenar los valores medidos por el micrófono. Estos valores serán almacenados en la memoria del C30.

El usuario podrá modificar los siguientes parámetros: la frecuencia de muestro, el número de muestras, el número de repeticiones de la secuencia, la ganancia de la señal ML, el canal de entrada (señal del micrófono) y el canal de salida (señal de secuencias ML). Como se ha mencionado anteriormente, este programa será ejecutado por el C30.

Para ejecutar un programa en el C30 se requieren dos programas:

- Un programa que lleva las instrucciones que realizará el C30
- Un programa para comunicar a la PC con el C30

En esta Sección se explicarán de manera muy general estos programas. Para mayor detalle, refiérase al apéndice A.2 y A.1

Programa de instrucciones para la PC.

En el programa de la PC, la comunicación con el C30 se realiza por medio de una biblioteca de funciones de comunicación. Las funciones de esta biblioteca sirven para bajar el código que debe ejecutar el C30, para correr el programa del C30 (o pararlo si es que esta corriendo), para transferir o recibir datos entre el programa que se esta corriendo en el C30 y la PC, y también para verificar que no haya errores al cargar el programa.

En este caso, necesitamos transferir ciertos parámetros que el usuario definirá para la medición: el ritmo de muestreo, la longitud de la secuencia, el número de repeticiones, etc. Una vez que son enviadas las condiciones iniciales, el programa del C30 comienza a correr, generando las secuencias ML y almacenando la señal medida por el micrófono. Una vez que el C30 finaliza su tarea, se leen los valores almacenados en la memoria del DSP y se guardan en un archivo.

Para facilitar la ejecución de este programa, se implementó una interfaz de usuario, que nos permite desde la línea de comandos de MS-DOS modificar los parámetros de medición. En la Sección (4.1.3) hablaremos con más detalle de esta interfaz.

Programa de instrucciones para el C30.

En este programa se ocupará un canal de salida de la tarjeta para generar la secuencia (conversión D/A) y un canal de entrada para capturar la señal registrada por el micrófono (conversión A/D). El programa esta constituido por un programa principal y una rutina de interrupción.

En el programa principal se leen las condiciones iniciales (enviadas por el programa de comunicación) para la generación de la secuencia: el ritmo de muestreo, el número de muestras, etc. Después se habilita la rutina de interrupción. La rutina de interrupción realiza la generación de la secuencia, la captura y almacenamiento de la señal de micrófono.

La tarea del programa principal consiste en recibir los valores iniciales y permanecer "ocioso" mientras ocurren las interrupciones. Cada vez que se ejecuta la rutina de interrupción se genera una muestra de la secuencia ML y se captura una muestra de la señal del micrófono. El programa principal controla el número de veces que se repite la secuencia. Una vez que se llega al número de repeticiones solicitado se deshabilitan las interrupciones y se entra en un ciclo durante el cual la PC lee la señal del micrófono.

Rutina de interrupción.

La rutina de interrupción se encarga de la generación de las muestras de las secuencias ML y del almacenamiento de la señal medida por el micrófono.

El código que genera la secuencia ML elige la ecuación de recurrencia correspondiente a la longitud de la secuencia solicitada. Este programa esta diseñado para producir secuencias de longitudes desde 1 ($2^1 - 1$) hasta 1,048,575 ($2^{20} - 1$).

Como se describió en la Sección 3.1.2, las secuencias ML se generan por medio de registros de corrimiento. Para obtener el siguiente elemento de la secuencia se necesitan de los $m - 1$ elementos anteriores, por lo que no es necesario guardar todos los valores de la secuencia. Además, como los elementos del registro de corrimeintos son sólo unos o ceros (bits), es posible almacenarlos en una variable entera de 32 bits. Con este tipo de variable es posible generar secuencias de longitud hasta 2^{32}. Y se tiene la ventaja de que se obtienen directamente los índices de la permutación 1.

Por ejemplo, si queremos generar una secuencia de longitud 7 ($m = 3$) con la siguiente ecuación de recurrencia $a_{i+3} = a_{i+2} + a_i$ se utiliza el siguiente código:

```
1  size_t shift_reg;
2  shift_reg=4;
3  a[0] = (shift_reg & 1);
4  a[2] = (shift_reg[0] & 4);
5  a[2] = (a[0] << 2) ^ a[2];
6  shift_reg = (shift_reg >> 1) | a[2];
```

En la línea 1 se define un arreglo donde se almacenarán los valores de la secuencia. En la línea 2 se asignan las condiciones iniciales del registro de corrimiento. Para $m = 3$ queda `shift_reg=4=(0100)`$_2$, que equivale a $a_2 = 1$, $a_1 = 0$, $a_0 = 0$. En la línea 3 se obtiene el elemento a_0 por medio de una máscara de bits.En la línea 4 se obtiene el elemento a_2 de la misma manera. En la línea 5 se actualiza el elemento a_2. Finalmente, en la línea 6 se actualiza el registro de corrimiento (se corren los bits a la derecha y se inserta el nuevo elemento).

Es evidente que este conjunto de instrucciones deberá repetirse un número de veces que corresponda a la longitud de la secuencia.

Para almacenar la señal medida por el micrófono fue necesario relocalizar todas las secciones que genera el compilador a un sólo banco de memoria, en este caso al Banco 0, con el fin de reservar el espacio de memoria del Banco 3 (memoria DUAL) para almacenar los valores medidos por el micrófono. Como las secciones producidas por el compilador están contenidas en una sola sección de la memoria (Banco 0), para el compilador no

existe el Banco 3. Por lo que sólo se tendrá acceso a esta porción de memoria por medio de apuntadores a la dirección del Banco 3 (de la dirección 0x30000 a la 0x33FFF).

La memoria de tipo DUAL tiene la característica de poder ser leída tanto por el C30 como por la PC.

La capacidad de almacenamiento del programa es de hasta 16,383 muestras. Se usa una localidad de memoria para indicar por medio de una bandera que el C30 terminó su tarea y dar paso a la lectura de los valores medidos por el micrófono que se encuentran almacenados en la memoria del C30.

4.1.3 Programa de interfaz de usuario en MS-DOS.

Este programa es ejecutado desde la línea de comandos del sistema operativo `MS-DOS` con el nombre de `c30mls`. Tiene la característica de poder modificar el ritmo de muestreo, la longitud de la secuencia y su amplitud, el número de repeticiones, el canal de entrada y salida. Este programa cumple con la norma POSIX[1] que cumplen algunos sistemas operativos, como es el caso de `UNIX`. En particular, esta norma indica que los argumentos opcionales de un programa se indican en la línea de comando anteponiendo un guión – a ciertas letras clave que cada programa especifica para indicar sus argumentos opcionales. Por ejemplo, en el caso del programa `c30mls`, si se desea que la longitud de la secuencia sea 1023, entonces se teclearía `c30mls -L 1023`.

Este tipo de comandos es posible implementarlos con las funciones contenidas en el archivo `getopt.c` (Apéndice A.3), las cuales facilitan la programación de la interfaz con el usuario.

Si sólo se escribe el nombre del programa (`c30mls`) el programa es ejecutado con los valores determinados por default. Para la frecuencia de muestreo: 48,000 Hz, la longitud de la secuencia de 1023, 1 Volt de amplitud, 100 repeticiones, canal 1 de salida y canal 1 de entrada. Tiene un comando de ayuda, si se teclea `c30mls -H` aparece en la pantalla la forma de usar el programa. Como se muestra a continuación:

```
c30mls -H
Usage:
   c30mls [-L length -N averages -R rate]  [-H]
   -L length of MLS sequence    <default: 1024>
   -N number of averages        <default: 100>
   -R sampling rate             <default: 48000.0>
   -I input channel             <default: 1>
   -O ouput channel             <default: 1>
```

[1] IEEE Std 1003.2-1992, *IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities — Volume 1* (1993, IEEE), ISBN 1-55937-255-9.

En Figura (4.1) se muestra el proceso completo para la obtención de la respuesta al impulso. El C30 genera las secuencias ML y almacena la señal medida por el micrófono. Después, esta señal es correlacionada con la señal de excitación (secuencias ML); la correlación se calcula con la transformada rápida de Hadamard. Como resultado, obtenemos la respuesta al impulso del cuarto.

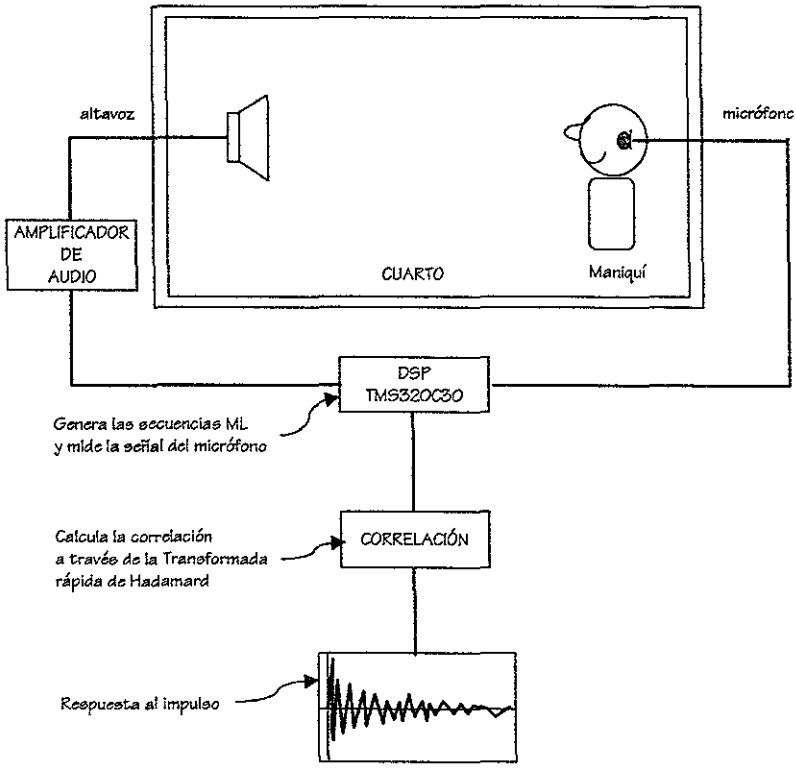


Figura 4.1: Implementación del equipo para la medición de la respuesta al impulso

4.1.5 Respuestas del recinto medidas.

En la gráfica (4.2) se muestran unos fragmentos de las las señales captadas por el micrófono de un salón de clases vacío y lleno. En las que podemos observar las diferencias de presión medidas por el micrófono.

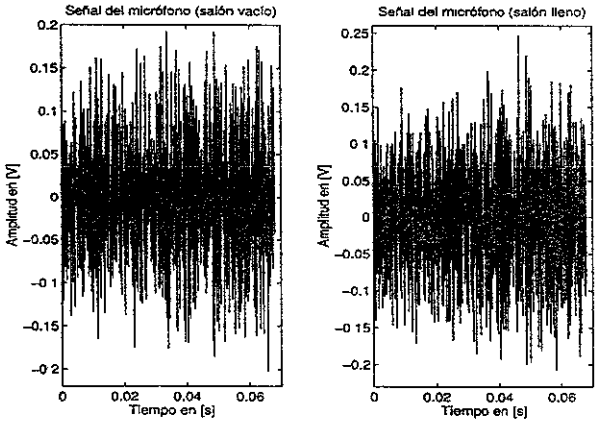


Figura 4.2: Señales medidas por el micrófono a una frecuencia de muestreo de 44.1 kHz

4.2 Obtención de la respuesta al impulso.

4.2.1 Programa que calcula la correlación.

El procedimiento de cálculo para obtener la correlación será la transformada rápida de Hadamard. Para ocupar este algoritmo directamente, es necesario realizar una serie de permutaciones antes y después de su cálculo. Entonces, el programa que calcula la correlación constará de 3 funciones: Una función que realiza la permutación 1 (previa al cálculo de la FHT), una función que calcula la transformada rápida de Hadamard; y por ultimo, una función que realiza la permutación 2 (posterior al cálculo de la FHT). Para mayor información, véase Apendice A.4.

Las funciones fueron programadas en lenguaje C. Se utilizó un compilador de C a 32 bits denominado `dgjpp`, ya que el compilador de `Borland` y de `Microsoft` no permiten reservar porciones de memoria de más de 64 kbytes.

El programa que calcula la correlación (denominado `mlscorr`) puede leer los valores de un archivo o se pueden ingresar directamente los valores desde la línea de comandos. Para el caso de tener nuestros datos en un archivo necesitamos direccionar el archivo (`mlscorr < archivo`) para que el programa sea capaz de leer los datos. Los resultados serán desplegados en la pantalla. Si se desea pueden ser guardados en un archivo (`mlscorr > archivo`).

Una vez obtenida respuesta del recinto calculamos la correlación de entrada-salida con

el programa `mlscorr`

```
mlscorr < mic.dat > ir.dat
```

Además, podemos encadenar (con el comando |) el programa `c30mls` y el programa `mlscorr`. Esto es, para que se obtenga directamente la respuesta al impulso. Esto se realiza de la siguiente manera:

```
c30mls -L ... >mic.dat | mlscorr <mic.dat >ir.dat
```

4.2.2 Respuestas al impulso del cuarto.

En la Figura(4.3) se muestran las respuestas al impulso del salón de clases lleno y vacío obtenidas con el programa `mlscorr`.

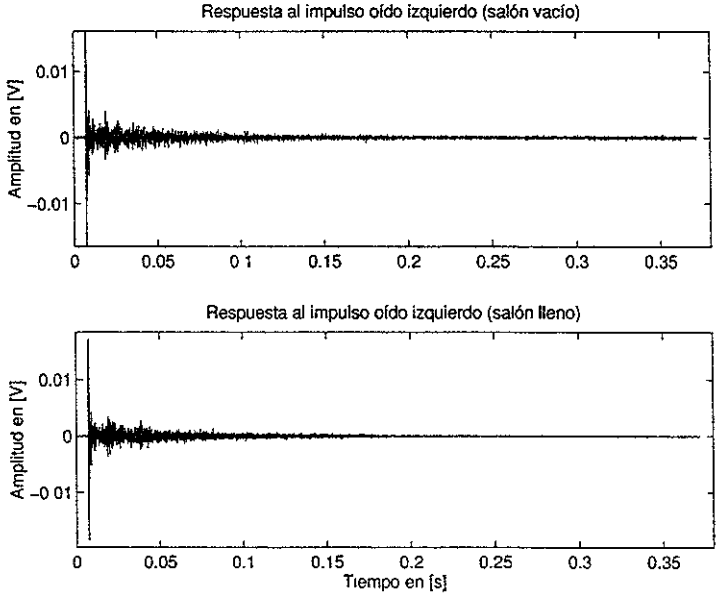


Figura 4.3: Respuestas al impulso correspondientes al salón de clases lleno y vacío a una frecuencia de muestreo de 44.1 kHz

4.3 Cálculo de los índices acusticos a partir de la respuesta al impulso.

En la presente Sección, se analizarán las características acústicas de un salón de clases. El salón de clases se localiza en la Sección de Acústica del Centro de Instrumentos. Los parámetros acústicos que se calcularán serán: la claridad, definición, EDT, ITDG [2] RT, S/N y la curva de decaimiento (la definición de estos parámetros se puede ver en la Sección 2.3.3). Para tal efecto, se realizó un programa en `Matlab` que obtiene los índices acústicos por bandas de octava. Este programa calcula dichos parámetros tanto para respuestas al impulso medidas como para respuestas al impulso simuladas.

En esta Sección, se presentarán ejemplos de señales medidas y simuladas (para mostrar la versatilidad del programa). Obtendremos los índices acústicos de la sala por bandas de octava. Compararemos los índices acústicos del salón lleno y vacío.

Las mediciones realizadas en el salón de clases se efectuaron con personas en su interior y vacío. Como podremos observar posteriormente, el tiempo de reverberación se reduce con la presencia de público. Esto se debe a que la gente actúa -por extraño que parezca- como material absorbente.

Características físicas del recinto.

SALA 1 (salón de clases)	
VOLUMEN:	59.32 m^3
MATERIAL DE LAS PAREDES:	concreto pintado
TECHO	yeso
PISO	linoleum
CAPACIDAD	

Tabla 4.2: Caracteristicas del salón de clases

4.3.1 Cálculo de los índices acústicos en `Matlab`.

Los programas para calcular los índices acústicos y la curva de decaimiento fueron desarrollados como funciones de `Matlab`. Para mayor detalle vea el Apéndice A.5.

[2] Se implementó una función en `Matlab` que calcula este parámetro, en el caso de tener la pura respuesta al impulso del recinto. Es decir, una respuesta que no contiene la respuesta del altavoz, el micrófono y el amplificador. De lo contrario, se obtendrá un resultado erróneo. Como lo pudimos comprobar durante el desarrollo de este trabajo.

El empleo de las funciones de Matlab para calcular los índices acústicos es muy sencillo. Sólo necesitamos cargar el archivo obtenido al calcular la correlación (ir.dat) y conocer la frecuencia de muestreo (44.1 kHz).

Para cargar un archivo en Matlab se escribe el comando load seguido del nombre del archivo. En este caso,

```
load ir.dat
```

Claridad (C80).

La función se definió como

```
clar80(ir,fs)
```

donde ir corresponde a la respuesta a impulso y fs es la frecuencia de muestreo en Hz. Es imporrtante aclarar que en todas las funciones la frecuencia de muestreo debe expresarse en [Hz]. El valor obtenido para la claridad está en [dB].

Para este caso:

```
clar80(ir,44100)
```

Curva de decaimiento.

La función es

```
decurve1(ir,fs)
```

Esta función entrega un vector. Se piden los mismos datos que para la claridad: la respuesta al impulso (ir) y la frecuencia de muestreo (fs).

para almacenar el vector en una variable sería

```
cd=decurve1(ir,44100);
```

Definición (D50).

```
def50(ir,fs)
```

el valor de la definición está expresado como un porcentaje.

Tiempo central (Tc) en [ms].

```
center(ir,fs)
```

Relación señal a ruido (S/N) en [dB].

```
SN(ir,fs)
```

Las funciones `def50`, `center`, y `SN` se manejan igual que la función `clar80`.

Tiempo de decaimiento temprano (EDT)en [s].

Es muy importante aclarar que el cálculo del EDT y los tiempos de reverberación se realizan con la curva de decaimiento.

```
EDT(cd,fs)
```

entonces

```
EDT(cd,44100)
```

Tiempo de reverberación RT15 y RT_X en [s].

El tiempo de reverberación RT15 corresponde al intervalo de los primeros 15 dB de la curva de decaimiento ajustados a una recta y extrapolando a 60 dB.

la función es

```
RT_15(cd,fs)
```

en este caso

```
RT_15(cd,44100)
```

El tiempo RT_X toma un intervalo de la curva de decaimiento que el usuario desee. Para mejores resultados, se busca un pedazo de la curva que esté lo más recto posible; y es que en algunos casos se presentan curvas de decaimiento con diferentes pendientes. Esta función es parecida a la anterior sólo que tiene la variante de que se ingresa el ancho del intervalo deseado.

La función es

```
RT_X(cd,li,fs)
```

donde li es el ancho del intervalo en dB

entonces si queremos un intervalo de 30 dB

```
RT_X(cd,30,44100)
```

Función filtoct.

Esta función estrictamente hablando, no calcula alguno de los índices acústicos, solamente filtra la señal, en este caso, la respuesta al impulso `ir` a una determinada frecuencia de corte `fc`. El filtrado se realiza con un filtro Butterworth paso-banda digital de orden `n`.

la función es

```
filtoct(ir,fs,fc,n)
```

esta señal entrega un vector. Si se quiere obtener la respuesta a impulso a 100 Hz con un filtro de orden 3 se tendría

```
filtoct(ir,44100,100,3)
```

Matlab cuenta con una ayuda, por ejemplo, si se teclea `help` y el nombre de la función aparece como se usa esa función.

El tiempo de reverberación se calculó a partir de la curva que representa la respuesta al impulso integrada o también llamada curva de decaimiento. En este trabajo no se quiso pasar por alto la importancia de esta curva siendo que con ella obtenemos el tiempo de reverberación, por lo que a continuación la presentamos en la Figura(4.4).

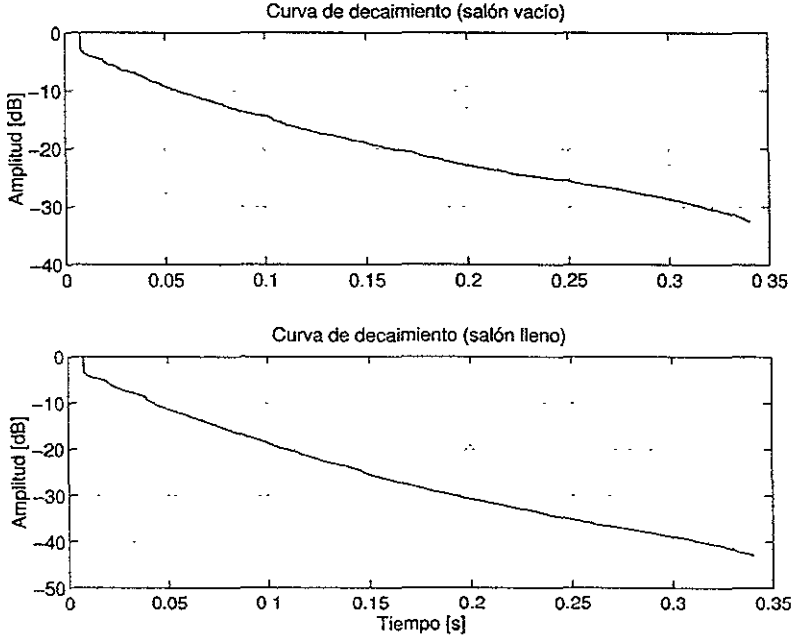


Figura 4.4: Curvas de decaimiento correspondientes al salón de clases lleno y vacío a una frecuencia de muestreo de 44.1 kHz

Como podemos observar, la pendiente de la curva de decaimiento del salón de clases

lleno es más pronunciada que la del salón vacío, lo que se traduce en un tiempo de reverberación más corto para para el salón lleno

4.3.2 Índices acústicos de un salón de clases.

En la Tabla (4.3) se muestran los índices acústicos calculados por bandas de octava para el salón de clases vacío. Es importante tener un panorama de estos índices a determinadas frecuencias para conocer su comportamiento a bajas, medias y altas frecuencias, ya que algunos parámetros como, la calidez y la brillantez dependen de los valores del tiempo de reverberación a bajas y altas frecuencias.

Kuttruff [10] sugiere un valor de la Claridad entre 0 y -3 dB, para salas de conciertos. Como podemos ver, los valores obtenidos para este parámetro, estan bastante alejados del intervalo sugerido; esto se debe principalmente, al tamaño del recinto, ya que en recinto pequeño el sonido recorre distancias muy cortas antes de chocar con las paredes. Lo que significa que la mayor parte de la energía se concentrará en un intervalo de tiempo muy corto.

[Hz]	CT [s]	C80 [dB]	D50 [%]	EDT [s]	RT/15 [s]	RT/30 [s]	S/N [dB]
nfilt	0.0234	12.3044	0.8792	0.5191	0.5781	0.7684	13.7096
63	0.0897	1.4344	0.1821	0.9578	1.0165	0.7210	1.2379
125	0.0896	2.3507	0.2516	0.9560	0.8285	0.6891	0.7458
250	0.0620	3.7748	0.5459	0.6115	0.6370	0.7153	4.9696
500	0.0545	6.3992	0.6470	0.7008	0.6854	0.6990	7.1472
1000	0.0459	6.7401	0.6922	0.6092	0.6365	0.6438	7.9566
2000	0.0360	9.5691	0.7547	0.4603	0.4879	0.6185	11.5826
4000	0.0183	16.8744	0.9391	0.2884	0.3465	0.9576	18.9311
8000	0.0107	21.7144	0.9866	0.2060	0.2741	1.2748	22.7138
16000	0.0108	20.4990	0.9863	0.1974	0.3200	1.4055	21.0000

Tabla 4.3: Índices Acústicos por bandas de octava correspondientes al salón de clases vacío

4.3.3 Comparación de los índices acústicos.

De los resultados mostrados en la tabla notamos que se reduce considerablemente el tiempo de reverberación (en un 65.4%) con la presencia de personas.

En la Tabla (4.4) se muestran los índices acústicos del salón de clases lleno y vacío.

Salón	CT [s]	C80 [dB]	D50 [%]	EDT [s]	RT/15 [s]	RT/30 [s]	S/N [dB]
vacío	0.0234	12.3044	0.8792	0.5191	0.5781	0.7684	13.7096
lleno	0.0185	15.6207	0.9263	0.3549	0.3772	0.4688	17.6809

Tabla 4.4: Índices Acústicos

Del tiempo de reverberación medido y con la formula de Sabine, podemos calcular el coeficiente de absorción total de la sala. Entonces

$$RT = .16\frac{V}{A}$$

despejando A

$$A = .16\frac{V}{RT}$$

De los valores obtenidos

$$A = .16\frac{59.32}{.5781} = 16.4179[\mathrm{m}^2]$$

donde $A = \alpha S$ y S es la superficie de absorción. Para este caso, $S = 2 \times (4.95 \times 5.35) + 2 \times (5.35 \times 2.24) + 2 \times (2.24 \times 4.95) = 99.109\mathrm{m}^2$, que corresponde a la superficie total del salón. Por lo tanto, el α correspondiente al salón vacío es igual a

$$\alpha = \frac{A}{S} = \frac{16.4179}{99.109} = 0.1657$$

Y el coeficiente de absorción con el salón lleno es $\alpha = .253$

4.3.4 Índices Acústicos de respuestas al impulso simuladas

Obtención de Respuestas al impulso simuladas

Las respuestas al impulso simuladas se obtuvieron con el programa de simulación de recintos `arsim` [9]. Este programa que se basa en el método trazado de rayos, predice la propagación de uno o varios rayos sonoros y obtiene el decremento en el tiempo de la energía del (o los) rayo(s). Para simular un recinto sólo se necesitan las características físicas del cuarto, tales como, sus dimensiones y los coeficientes de absorción de las paredes. Las dimensiones del cuarto se presentan en la Tabla (4.2). Para tener las mismas condiciones que en la medición real, el coeficiente de absorción se calculó con la fórmula desarrollada por Sabine, con el tiempo de reverberación medido y el volumen del cuarto. Para el salón vacío se obtuvo un coeficiente $\alpha = 0.1657$.

La Figura (4.5) muestra la respuesta al impulso simulada del salón de clases vacío.

La Tabla 4.5 muestra los índices acústicos por bandas de octava de la respuesta al impulso simulada obtenida por el programa `arsim`.

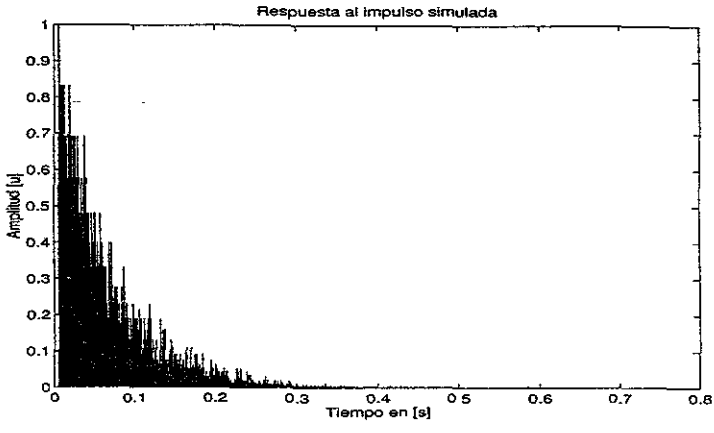


Figura 4.5: Respuesta al impulso energetizada $h^2(t)$ para un salón de clases vacío.

[Hz]	CT [s]	C80 [dB]	D50 [%]	EDT [s]	RT/15 [s]	RT/30 [s]	S/N [dB]
nfilt	0.0498	6.7379	0.6370	0.6236	0.6417	0.7221	7.6406
63	0.0305	13.0753	0.9127	0.4586	0.3764	0.3320	16.9616
125	0.0200	16.0524	0.9350	0.4919	0.4161	0.3661	20.8160
250	0.0191	17.5176	0.9365	0.2957	0.3445	0.3554	18.2711
500	0.0147	24.0383	0.9823	0.1608	0.1910	0.3096	26.0529
1000	0.0183	18.9475	0.9574	0.2232	0.2935	0.3655	21.9283
2000	0.0245	14.6308	0.8890	0.3310	0.3453	0.4022	17.5093
4000	0.0320	12.6438	0.8379	0.3595	0.3734	0.3988	14.7797
8000	0.0333	13.1562	0.8354	0.3274	0.3554	0.4118	14.9294
16000	0.0341	12.4118	0.8186	0.3491	0.3697	0.4090	14.2784

Tabla 4.5: Índices acústicos de una IR simulada.

Capítulo 5

Conclusiones.

La importancia de la evaluación de la calidad acústica de un recinto en base a determidados índices, radica en el hecho de poder emitir un juicio objetivo. Muchas veces, la calidad de las salas (especialmente en salas de conciertos) es calificada de acuerdo al jucio de gusto de varios expertos, lo cual lo hace muy subjetivo.

Como pudimos ver durante este trabajo, estos índices son obtenidos a partir de la respuesta al impulso. Los índices acústicos están definidos por relaciones matemáticas no muy complicadas. Lo que hace posible realizar programas "sencillos" para su cálculo. En este caso, se implementaron funciones en `Matlab`.

Una de las características más importantes en el análisis de los sistemas es la respuesta al impulso. Con ella, podemos predecir el comportamiento del sistema a cualquier excitación. Para nuestro caso particular, a partir de la respuesta al impulso es posible conocer las características acústicas de un recinto.

El sistema de medición desarrollado en este trabajo, que obtiene la respuesta del recinto por medio del sistema de procesamiento digital de de señales (C30) puede generar secuencias de longitud 1 a 1,048,576. Sin embargo, solamente tiene una capacidad de almacenamiento de 16,383 muestras. Esto se debe, a que las muestras son guardadas en la memoria del C30. La principal limitante para cualquier implementación que se realice, será la capacidad de almacenamiento. El realidad, el tamaño que puede almacenar el sistema desarrollado es muy pequeño y no alcanza a registrar la respuesta al impulso de una sala de conciertos. Aunque en si, el método es bastante práctico.

El empleo de secuencias ML como señal de excitación presentan alta inmunidad al ruido presentando relaciones de señal a ruido de $10 \log_{10} \textit{longitud de la secuencia}$. El mismo autor del método (Schroeder) comenta que es posible realizar la medición de la respuesta al impulso del recinto en "vivo". Esto quiere decir, que la medición se puede realizar durante la presentación de un concierto, asegurandose que la señal de exictación sea emitida a un nivel muy bajo y la señal se deberá repetir lo que dure el concierto.

Otra ventaja muy importante que posee ésta técnica, es que es posible utilizar un algoritmo para calcular la correlación con él cual se reduce considerablemente el tiempo de procesamiento. Este procedimiento de cálculo es la Transformada rápida de Hadamard. Lo anterior es muy importante si se requiere desarrollar un programa que trabaje en tiempo real. Este algoritmo sólo se puede utilizar cuando se tienen señales de entrada de la forma de la secuencias ML.

Vale la pena mencionar que éste método no solamente es válido para la obtención de la respuesta al impulso del recinto. Se puede aplicar para conocer la respuesta al impulso de cualquier sistema lineal.

El empleo de programas de simulación de recintos es muy importante en el diseño de recintos, ya que nos proporcionan valores estimados del comportamiento del sonido en una sala con determinada forma y algún posible recubrimiento en las paredes. De cualquier manera, se requiere conocer el comportamiento real del sonido. Algunos autores coinciden en que es necesario medir la respuesta en diferente etapas de la construcción. Por lo que resulta necesario contar con una técnica de medición de la respuesta al impulso. Y qué mejor que usar una técnica que sea sencilla de realizar como la desarrollada en este trabajo.

Apéndice A

Listado de programas

A.1 Programa mls

```
/* MLS.C */
/* PROGRAMA  QUE GENERA SECUENCIAS PSEUDO-ALEATORIAS DE GRADO 1 A 20 */
/* Y ALMACENA LA SEAL CAPTADA POR EL MICROFONO */
asm(" .length 60");
asm(" .width 120");

$include "mlslib.h"

/* DECLARACION DE VARIABLES GLOBALES */
long *flag= (long *) 0x30000;
long *y = (long *) 0x30001;
long *pm=(long *)0x30002;
long *pp=(long *)0x30003;
double *pfrec=(double *)0x30004;
double *pgain=(double *)0x30005;
long *pinc=(long *)0x30006;
long *poutc=(long *)0x30007;

$define FLAG    (flag[0])

long *DACaddress,*ADCaddress,m,output=0,input=0;
int dac_level[2]={0,0};
unsigned long i,j,k,len,indice,bit0,bit,temp,mask;
size_t shift_reg;
 main()
{
  long  *timer,*timerctrl,P;
  double frec,counter,level;
   FLAG=1;
  DACaddress=(long *)0x804000; /* apunta a la direccion del DAC A */
  ADCaddress=(long *)0x804000; /* apunta a la direccion del ADC A */
  timerctrl=     (long *)0x808030;
  timer= (long *)0x808038;
   frec=*pfrec;
   frec=1000000/(frec*.12);
   counter=frec;
  *timerctrl= 0x00000601;
  *timer=counter;
  *timerctrl= 0x000006c1;
```

```
  m=*pm;
   P=*pp;
  FLAG=P;
  level=*pgain;
  input=*pinc;
  output=*poutc;
  len=(1L<<m)-1;
  indice=0L;
  indice=1L<<(m-1);
  i=0;
  j=0;
  mls_init(&shift_reg,m);
  dac_level[output] = (long)(2147483648.0 * level / 3.0); /* 2^31 */
  asm("  OR  2h, IE"); /* habilita interrupcion */
  asm("  OR  2000h, ST"); /* habilita interrupcion global del CPU */
    while(j!=(P+1));


   /* DESHABILITA RUTINA DE INTERRUPCION */
  asm("  AND OFFFFFFFDh, IE");
  asm("  AND OFFFFDFFFh, ST");
  FLAG=P+1;
  }/* end main */

/*******************RUTINA DE INTERRUPCION **********************/
  asm("  .sect  \".int02\"");
  asm("  .word  _c_int02");
  asm("  .text          ");
  c_int02()

  {

   if(j==0) {
       y[i]=0;
   } else {
       y[i]+=(ADCaddress[input]>>16);
   }

   if (mls(&shift_reg,m)) {
      DACaddress[0] = -dac_level[0];
      DACaddress[1] = -dac_level[1];
    } else {
      DACaddress[0] = dac_level[0];
      DACaddress[1] = dac_level[1];
    }
  i=i+1;
  if(i==len){
  i=0;
  j=j+1;
  }


  } /* end of interrupt service routine */
```

A.2 Programa c30mls

```
/*
```

```
runmls.c CORRE MLS.OUT
   */
#include "tms30.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

extern int getopt( int argc, char *const *argv,
const char *optstring );
extern char *optarg;
extern int opterr;
extern int optind;
extern int      optopt;
char progname[]="c30mls";
static const char optlist[]="L:N:R:I:O:G:H";
static const char       usage[] =       /* usage message */
"Usage:\n  %s [-L length -N averages -R rate] [-H]"
"\n -L length of MLS sequence in samples (default: 1024)"
"\n -N number of averages (default: 100)"
"\n -R sampling rate in Hz (default: 48000.0)"
"\n -I input channel (default:1)"
"\n -O output channel (default:1)"
"\n -G gain in volts (default:1)";
time_t time_out;
#define FLAG 0x30000
#define Y    0x30001
#define TIME_OUT(x,y) ((x)*(y))
unsigned long  pow2(long z);
long muestras( long p);
int  main(int argc,char *argv[])
{
long  m=1024,n=100 ,i,inputc=0,outputc=0;
 long y=10;
double  f=48000.0,p,gain=1.0;
register int c;
unsigned long len,lee;
long out;
int carga,flag=0;
SelectBoard(0x390); /* direccion de la tarjeta */
carga=coffLoad("mls.out");
for(i=1;i<=argc;i++){
 c=getopt(argc,argv,optlist);
  switch(c){

 case 'R':

     sscanf(optarg,"%lf",&f);
     break;
 case 'H':
    fprintf(stderr,usage,progname);
    exit(0);
    break;
 case 'L':
     sscanf(optarg,"%ld",&m);
    y=muestras(m);
    break;
 case 'N':
     sscanf(optarg,"%ld",&n);
     break;
```

```
 case 'I':
      sscanf(optarg,"%ld",&inputc);
      inputc=inputc-1;
      break;
 case 'O':
      sscanf(optarg, "%ld",&outputc);
      outputc=outputc-1;

      break;
 case 'G':
      sscanf(optarg,"%lf",&gain);
      break;

 case '?':
      exit(0);
      break;
default:

;
}/* end of switch */
}/* end of for */
if(carga != 0 ){
printf("\n Error al cargar el programa");
exit (0);
}/* end if */

 PutInt(0x30002,DUAL,y);
 PutInt(0x30003,DUAL,n);
 PutFloat(0x30004,DUAL,f);
 PutFloat(0x30005,DUAL,gain);
 PutInt(0x30006,DUAL,inputc);
 PutInt(0x30007,DUAL,outputc);

 PutInt(FLAG,DUAL,0x0L);
 len=(1L<<y)-1;

 Reset(); /* Comienza el programa del DSP */

  time_out=time(NULL)+TIME_OUT(len,n);
  while (GetInt(FLAG,DUAL) != n+1);
      if (time(NULL) > time_out) {
 fprintf(stderr,"Time-out waiting for DSP\n");
 exit(1);
 }
/*printf("los valores medidos por el microfono son: \n");*/
for(i=0;i<len;i++){
  out=GetInt(Y+i,DUAL);
  out=(out/n);
    printf("\n %ld  ",out);
    } /*end of for */
} /* end main */

 unsigned long int pow2(long z){
unsigned long int w;
w=1L<<z;
return w;
}

 long  muestras( long p){
 long m1,m2,y,x=10;
```

```
int band=1;
m1=pow2(x);
if(p>m1){
  while(band==1){
       y=x+1;
       m2=pow2(y);
       if(p>=m1 && p<=m2){
 band=0;
 return y;
 } /* end of if */
 else
 x=y;
} /* end of while */
   } /* end of if */
else
{
while(band==1){
y=x-1;
m2=pow2(y);
if( p>m2 && p<= m1){
band=0;
return x;
} /* end of if */
else{
x=y;
m1=pow2(x);
}
}/*end of while */
}
}
```

A.3 Programa getopt

```
/*
 * Copyright (c) 1994 by Academic Press, Boston, Massachusetts.
 * Written by Douglas A. Gwyn.  Not derived from licensed software.
 * From the book "Software Solutions in C", edited by Dale Schumacher.
 *
 * Permission is granted to anyone to use this software for any
 * purpose on any computer system, and to redistribute it in any way,
 * subject to the following restrictions:
 *
 *   1. The author is not responsible for the consequences of use of
 * this software, no matter how awful, even if they arise
 * from defects in it.
 *
 *   2. The origin of this software must not be misrepresented, either
 * by explicit claim or by omission.
 *
 *   3. Altered versions must be plainly marked as such, and must not
 * be misrepresented (by explicit claim or omission) as being
 * the original software.
 *
 *   4. This notice must not be removed or altered.
 */

/*
getopt -- parse command line arguments
```

```
complies with the following standards:
System V Interface Definition, Third Edition, Volume 1
X/Open Portability Guide -- XSI System Interface and
Headers, Issue 3
Application Environment Specification -- Operating System
Programming Interfaces Volume, Revision A

Declare as:

#include <stdio.h>// for EOF
extern int getopt( int argc, char *const *argv,
   const char *optstring );
extern char *optarg;
extern int opterr;
extern int optind;
extern int optopt; // optionally supported; see below

Notes:

This code fully supports the System V Command Syntax Standard as
specified in the System V Interface Definition, Issue 3.  The
intention is that this interface be used by most C programs to
parse their command-line arguments.

The origin of this code can be traced back to 7th Edition UNIX,
through its release into the public domain in conjunction with the
presentation of the paper "Proposed Syntax Standard for UNIX System
Commands" by Kathleen Hemenway and Helene Armitage at the January
1984 UniForum Conference in Washington, DC (sponsored jointly by
/usr/group and the USENIX Association).  The code has subsequently
been overhauled to increase its portability and to better handle
certain unusual situations.

This implementation does not support multibyte option characters.
(They should not be used by portable programs anyway.)

The second parameter of getopt() could have been written with an
additional "const" type qualifier, but that would require most
invocations of getopt() to use an explicit cast; this was deemed
inappropriate and thus is not in the official interface spec.

Unless RIGID_SPEC is defined, this implementation does not enforce
the rules that an option-with-argument must not be clustered with
other options in the same command-line argument and that its
parameter argument must not be contained within the same command-
line argument.  AT&T's getopt() never enforced these, and SVID
Issue 3 has now officially relaxed the latter rule.  However,
invokers of programs should follow the rules, so you may want to
enable RIGID_SPEC in order to help users develop good habits.

  The original version released into the public domain by AT&T also
set an external variable "optopt" to the actual option character,
which differs from the value returned by getopt() only when the
latter is '?' (error indicator).  "optopt" is not specified as
part of the official interface; if you want it, define SET_OPTOPT.
 */
 #define RIGID_SPEC /* enable to enforce all rules */
 #define SET_OPTOPT /* enable to support "optopt" */
 #define ROBUST /* sane behavior when invoked with bad arguments */
 #define DEBUG /* enable for assertion checking */
```

ESTA TESIS NO SALE
DE LA BIBLIOTECA


```
#ifndef DEBUG
#define NDEBUG
#endif

#include <assert.h>
#include <stdio.h>
#include <string.h>

#define EOS '\0'/* C string terminator */

char *optarg = NULL; /* option parameter if any */
int optind = 1; /* next argv[] index */
static int optci = 1; /* next argv[optind][] index */
int opterr = 1; /* error => print message if set */
#ifdef SET_OPTOPT
int optopt = EOS; /* option letter */
#endif

static int /* prints diagnostic, returns '?' */
Error( const char *name,/* argv[0] (program name) */
const char *mess,/* diagnostic message */
int c /* conflicting option character */
     ) {
assert(name != NULL);
assert(mess != NULL);

if ( opterr )
fprintf( stderr, "%s: %s -- %c\n", name, mess, c );

return '?'; /* erroneous-option marker */
}

int
getopt( int argc,/* argc from main() */
register char *const *argv,/* argv from main() */
const char *optstring /* supported option letters,
   followed by ":" when
   option takes parameter */
     ) {
register int c; /* option letter from argv */
register const char *cp; /* -> option letter in optstring */
#ifdef RIGID_SPEC
register int savci; /* saved optci for parameter test */
#endif

assert(argc > 0);
assert(argv != NULL);
assert(argv[0] != NULL);
assert(optind >= argc || argv[optind] != NULL);
assert(argv[argc] == NULL);
assert(optstring != NULL);

#if defined(ROBUST) && !defined(DEBUG) /* DEBUG passed assert() */
if ( argc <= 0
  || argv == NULL
  || optind < argc && argv[optind] == NULL
  || argv[argc] != NULL
  || optstring == NULL
   )
```

```
return EOF; /* give up on argument parsing */
#endif

optarg = NULL;

if ( optci == 1 ) /* beginning of new argument */
if ( optind >= argc /* no more arguments */
  || argv[optind][0] != '-'/* no more options */
  || argv[optind][1] == EOS /* not option: stdin */
   )
return EOF;
else if ( strcmp( argv[optind], "--" ) == 0 )
{
++optind; /* skip over "--" */
return EOF; /* "--" marks end of options */
}

c = argv[optind][optci]; /* option letter */
#ifdef SET_OPTOPT
optopt = c;
#endif
#ifdef RIGID_SPEC
savci =
#endif
optci++; /* get ready for next letter */

if ( argv[optind][optci] == EOS ) /* end of this argument */
{
++optind; /* advance to next argument */
optci = 1; /* beginning of new argument */
}

if ( c == ':'/* (not a valid option letter) */
  || (cp = strchr( optstring, c )) == NULL /* not listed */
   )
return Error( argv[0], "illegal option", c );

if ( cp[1] == ':' ) /* option takes parameter */
{
#ifdef RIGID_SPEC
/* (Minimize the number of additional spurious messages.) */

if ( savci != 1 )
{
if ( optind < argc )
++optind; /* skip over noisy junk */

optci = 1; /* next option from new argument */

return Error( argv[0],
        "option must not be clustered",
      c
    );
}

if ( optci != 1 ) /* parameter (?) in same argument */
{
++optind; /* skip over noisy junk */
optci = 1; /* next option from new argument */
```

```
return Error( argv[0],
      "option argument must be separate",
      c
    );
}
#else
if ( optci != 1 ) /* parameter in same argument */
{
optarg = &argv[optind][optci]; /* -> parameter */
optci = 1; /* next option from new argument */
}
else
#endif
if ( optind >= argc )
return Error( argv[0],
      "option requires an argument",
      c
    );
else /* parameter in separate argument */
optarg = argv[optind]; /* -> parameter */

/* (factored out the following) */
++optind; /* skip over parameter */
}

return c; /* option letter */
}
```

A.4 Programa mlscorr

A.4.1 mlslib1.c

```
/*
 * MLSLIB.C - Functions to work with maximum-length sequences.
 */

#include "mlslib.h"
/*#include <stdio.h>*/
#include <stdlib.h>

int
a[6][6]={[
{2,3,7,5,4,0}, /* m=6 y 19 */
{5,8,9,6,3,2},/* m=12 */
{9,10,12,3,2,0},/* m=13*/
{2,3,13,11,10,0},/* m=14*/
{11,13,14,4,2,1},/*m=16*/
{13,14,18,5,4,0},/*m=19 */
};


void mls_init(size_t *shift_reg, size_t m)
{
  shift_reg[0] = (1 << (m - 1));
}

int mls(size_t *shift_reg, size_t m)
{
size_t i,bit0,bit,temp,mask,len;
```

```
int output,j;


bit0=(shift_reg[0]&1)<<(m-1);

switch (m) {

   case 2:
   case 3:
   case 4:
   case 6:
   case 7:
   case 15:
bit=shift_reg[0]&(1L<<m-1);
temp=bit0^bit;

   break;

   case 5:   case 11:
bit=(shift_reg[0]&(1L<<m-2))<<1;
temp=bit0^bit;

    break;

   case  8:   case 19:
       for(j=0;j<3;j++){
bit=(shift_reg[0]&(1L<<a[0][j]))<<a[0][j+3];
temp=(bit0^bit);
bit0=temp;
   }

   break;

   case 9:
bit=(shift_reg[0]&32)<<3;
temp=bit0^bit;

    break;

   case 10:   case 17:   case 20:
bit=(shift_reg[0]&(1L<<m-3))<<2;
temp=bit0^bit;

    break;

   case 12:
       for(j=0;j<3;j++){
bit=(shift_reg[0]&(1L<<a[1][j]))<<a[1][j+3];
temp=(bit0^bit);
bit0=temp;
   }

    break;

   case 13:
       for(j=0;j<3;j++){
bit=(shift_reg[0]&(1L<<a[2][j]))<<a[2][j+3];
temp=(bit0^bit);
bit0=temp;
   }
```

```
    break;

   case 14:
       for(j=0;j<3;j++){
bit=(shift_reg[0]&(1L<<a[3][j]))<<a[3][j+3];
temp=(bit0^bit);
bit0=temp;
  }

   break;

   case 16:
       for(j=0;j<3;j++){
bit=(shift_reg[0]&(1L<<a[4][j]))<<a[4][j+3];
temp=(bit0^bit);
bit0=temp;
   }

    break;

   case 18:
bit=(shift_reg[0]&2048)<<6;
temp=bit0^bit;
break;


   default:
    /*fprintf(stderr,"Orden no programado\n");*/
    return -1; /* Orden no programado */

  }/* end of switch */

shift_reg[0]=(shift_reg[0]>>1)|temp;
return (bit) ? 1: 0;
}/*end of mls */

static void mls_perm1(double *y, double *x, size_t m)
{
  size_t shift_reg;
  size_t i, L;

  y[0] = 0.0;
  L = (1 << m) - 1;
  mls_init(&shift_reg,m);
 for (i=0; i < L; i++) {
    y[shift_reg] = x[i];
    mls(&shift_reg,m);
  }
}

static void mls_perm2(double *y, double *x, size_t m)
{
int j;
unsigned long int indice,pow2,len,bitm_1,bit,temp,mask,i;

len=(1<<m)-1;
indice=1<<(m-1);
pow2=1<<(m-1);
for(i=0;i<len;i++){
```

```
y[i]=x[indice];
switch(m){
case  2:
case  3:
case  4:
case  6:
case  7:
case 15:
   bitm_1=(indice&1)<<(m-1);
   bit=indice&pow2;
   temp=bit^bitm_1;
   indice=indice&(0xffffffff-pow2)|temp;
   indice=(indice>>1)|bitm_1;
break;

case  5:
case 11:
   bitm_1=(indice&1)<<(m-2);
   bit=indice&(1<<m-2);
   temp=bit^bitm_1;
   indice=indice & (0xffffffff-(1<<m-2))|temp;
   indice=(indice>>1)|(bitm_1<<1);
break;

case 8:

 for(j=0;j<3;j++){
    bitm_1=(indice&1)<<a[0][j];
    bit=indice&(1L<<a[0][j]);
    mask=0xffffffff-(1L<<a[0][j]);
    temp=bitm_1^bit;
    indice=(indice&mask)|temp;
    }
    indice=(indice>>1)|bitm_1;
    break;
case 9:
   bitm_1=(indice & 1)<<5;
   bit=indice&32;
   temp=bit^bitm_1;
   indice=indice &(0xffffffff-32 )|temp;
   indice=(indice>>1)|(bitm_1<<3);
   break;
case 10:
case 17:
case 20:
   bitm_1=(indice & 1)<<7;
   bit=indice&128;
   temp=bit^bitm_1;
   indice=indice & (0xffffffff-128)|temp;
   indice=(indice>>1)|(bitm_1<<2);
   break;
case 12:
    for(j=0;j<3;j++){
    bitm_1=(indice&1)<<a[1][j];
    bit=indice&(1L<<a[1][j]);
    mask=0xffffffff-(1L<<a[1][j]);
    temp=bitm_1^bit;
    indice=(indice&mask)|temp;
    }
    indice=(indice>>1)|(bitm_1<<2);
```

```
    break;
case 13:
    for(j=0;j<3;j++){
    bitm_1=(indice&1)<<a[2][j];
    bit=indice&(1L<<a[2][j]);
    mask=0xffffffff-(1L<<a[2][j]);
    temp=bitm_1^bit;
    indice=(indice&mask)|temp;
    }
    indice=(indice>>1)|bitm_1;
    break;
case 14:
    for(j=0;j<3;j++){
    bitm_1=(indice&1)<<a[3][j];
    bit=indice&(1L<<a[3][j]);
    mask=0xffffffff-(1L<<a[3][j]);
    temp=bitm_1^bit;
    indice=(indice&mask)|temp;
    }
    indice=(indice>>1)|bitm_1;
    break;
case 16:
       for(j=0;j<3;j++){
 bitm_1=(indice&1)<<a[4][j];
 bit=indice&(1L<<a[4][j]);
 mask=0xffffffff-(1L<<a[4][j]);
 temp=bitm_1^bit;
 indice=(indice&mask)|temp;
 }
    indice=(indice>>1)|(bitm_1<<1);
    break;
case 18:
   bitm_1=(indice & 1)<<7;
   bit=indice&2048;
   temp=bit^bitm_1;
   indice=indice &(0xffffffff-2048)|temp;
   indice=(indice>>1)|(bitm_1<<6);
   break;
case 19:
   for(j=0;j<3;j++){
       bitm_1=(indice&1)<<a[5][j];
       bit=indice&(1L<<a[5][j]);
       mask=0xffffffff-(1L<<a[5][j]);
       temp=bitm_1^bit;
       }
       indice=(indice>>1)|bitm_1;
break;

}/* end switch */

} /* end of  for */
}

static void mls_norm(double *x, size_t m)
{
  size_t L;

  L = (1 << m) - 1;
  for (m=L; m--; x++) {
    (*x) /= L;
```

```
  }
}

int mls_corr(double *x, size_t m)
{
  double *y;

  y = malloc((1 << m)*sizeof(double));
  if (y == NULL) {
    return ERR;
  }
  mls_perm1(y,x,m);
  fht(y,m,0);
  mls_perm2(x,y,m);
  mls_norm(x,m);
  free(y);
  return OK;
}


#if 0
void fht_original(double *x, size_t m, int flag)
{
  double temp;
  size_t i1, i2, j, k, n, n1, n2;

  n2 = 1;
  n = (1 << m);
  for (k=0; k < m; k++) {
    n1 = n2;
    n2 = 2 * n1;
    for (j=0; j < n1; j++) {
      for (i1=j, i2=j+n1; i1 < n; i1+=n2, i2+=n2) {
temp = x[i1];
x[i1] = temp + x[i2];
x[i2] = temp - x[i2];
      }
    }
  }
  if (flag) {
    for (k=n; k--; x++) {
      (*x) /= n;
    }
  }
}
#endif

void fht(double *x, size_t m, int flag)
{
  double temp;
  size_t i1, i2, j1, j2, k, n, n1, n2;

  n2 = 1;
  n = (1 << m);
  for (k=0; k < m; k++) {
    n1 = n2;
    n2 = 2 * n1;
    for (j1=0; j1 < n; j1+=n2) {
      j2 = j1 + n1;
      for (i1=j1, i2=j2; i1 < j2; i1++, i2++) {
```

```
temp = x[i1];
x[i1] = temp + x[i2];
x[i2] = temp - x[i2];
      }
    }
  }
  if (flag) {
    for (k=n; k--; x++) {
      (*x) /= n;
    }
  }
}
```

A.4.2 programa fht.c

```
/*
 * FHT.C - Fast Hadamard transform of data read from stdin
 */

#include "tools.h"
#include "mlslib.h"

char progname[] = "fht";
char usage[] = "Usage: %s [-i] [-h]\n\n"
"   -i  Normalize output 1/N (for inverse transform)\n"
"   -h  Print program usage and exit\n";
char bad_usage[] = "Error in command line.  Try '%s -h'";

int main(int argc, char *argv[])
{
  int flag=0;
  size_t n, m;
  double *x;

  switch (argc) {
   case 2:
    if (strcmp(argv[1],"-i") == 0) {
      flag=1;
    } else if (strcmp(argv[1],"-h") == 0) {
      fprintf(stderr,usage,progname);
      return EXIT_SUCCESS;
    } else {
      fprintf(stderr,bad_usage,progname);
      return EXIT_FAILURE;
    }
    break;
   case 1:
    break;
   default:
    fprintf(stderr,bad_usage,progname);
    return EXIT_FAILURE;
  }

  x = read_data(stdin, &n);
  if (x == NULL) {
    fprintf(stderr,"%s: Out of memory\n", progname);
    return EXIT_FAILURE;
  }
```

```
  if (n == 0) {
    fprintf(stderr,"%s: Empty data set\n", progname);
    return EXIT_SUCCESS;
  }
  m = log2(n);
  n = (1 << m);
  fht(x,m,flag);
  print_data(stdout,x,n);
  return EXIT_SUCCESS;
}
```

A.4.3 programa mls.c

```
/*
 * MLS.C - Maximum-length sequences generator.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "mlslib.h"

char progname[] = "mls";
char usage[] = "Usage: %s [-h] [order]\n\n"
"   -h  Print program usage and exit\n";
char bad_usage[] = "Error in command line.  Try '%s -h'\n";

int main(int argc, char *argv[])
{
  size_t n, m=3, L, shift_reg;

  switch (argc) {
   case 2:
    if (strcmp(argv[1],"-h") == 0) {
      fprintf(stderr,usage,progname);
      return EXIT_SUCCESS;
    } else if (sscanf(argv[1],"%u",&m) != 1) {
      fprintf(stderr,bad_usage,progname);
      return EXIT_FAILURE;
    }
    break;
   case 1:
    break;
   default:
    fprintf(stderr,bad_usage,progname);
    return EXIT_FAILURE;
  }

  mls_init(&shift_reg,m);
  L = (1L << m) - 1;
  for (n=0; n < L; n++) {
    printf("% 1E\n",(double)(1-2*mls(&shift_reg,m)));
  }
  return EXIT_SUCCESS;
}
```

A.4.4 programa mlscorr.c

```
/*
 * MLSCORR.C - MLS cross-correlation using fast Hadamard transform
 */

#include "tools.h"
#include "mlslib.h"

char progname[] = "mlscorr";
char usage[] = "Usage: %s [-h]\n\n"
"   -h  Print program usage and exit\n";
char bad_usage[] = "Error in command line.  Try '%s -h'\n";

int main(int argc, char *argv[])
{
  size_t n, m;
  double *x;

  switch (argc) {
   case 2:
    if (strcmp(argv[1],"-h") == 0) {
      fprintf(stderr,usage,progname);
      return EXIT_SUCCESS;
    } else {
      fprintf(stderr,bad_usage,progname);
      return EXIT_FAILURE;
    }
    break;
   case 1:
    break;
   default:
    fprintf(stderr,bad_usage,progname);
    return EXIT_FAILURE;
  }

  x = read_data(stdin, &n);
  if (x == NULL) {
    fprintf(stderr,"%s: Out of memory\n", progname);
    return EXIT_FAILURE;
  }
  if (n == 0) {
    fprintf(stderr,"%s: Empty data set\n", progname);
    return EXIT_SUCCESS;
  }
  m = log2(n);
  if (n != (1 << m) - 1) {
    fprintf(stderr,"%s: Wrong length in data set\n", progname);
    return EXIT_FAILURE;
  }
  if (mls_corr(x,m) == ERR) {
    fprintf(stderr,"%s: Error in correlation function\n", progname);
    return EXIT_FAILURE;
  }
  print_data(stdout,x,n);
  return EXIT_SUCCESS;
}
```

A.4.5 programa tools.c

```
/*
 * TOOLS.C - Auxiliary functions
 */

#include "tools.h"

size_t log2(size_t n)
{
  size_t p;

  for (p=0; (1 << p) < n; p++);
  return p;
}

void clear_data(double *x, size_t n)
{
  while (n--) {
    (*x++) = 0.0;
  }
}

double *read_data(FILE *fp, size_t *np)
{
  double *x=NULL, *x_old, temp;
  size_t n=0, m=16, m_old;

  x = calloc(m,sizeof(double));
  if (x == NULL) {
    *np = 0;
    return NULL;
  }
  while (fscanf(fp, "%lf", &temp) == 1) {
    if (n == m) {
      m_old = m;
      x_old = x;
      m = 2 * m_old;
      x = realloc(x_old,m*sizeof(double));
      if (x == NULL) {
free(x_old);
*np = 0;
return NULL;
      }
      clear_data(x+m_old,m_old);
    }
    x[n++] = temp;
  }
  *np = n;
  return x;
}

void print_data(FILE *fp, double *x, size_t n)
{
  while (n--) {
    fprintf(fp, "% 1E\n", *x++);
  }
}
```

A.4.6 mlslib.h

```
/*
 * MLSLIB.H - Functions to work with maximum-length sequences.
 */

#ifndef MLSLIB_H
#define MLSLIB_H

#include <stdlib.h>

#define OK 1
#define ERR 0

void mls_init(size_t *shift_reg, size_t m);
int mls(size_t *shift_reg, size_t m);
int mls_corr(double *x, size_t m);
void fht(double *x, size_t m, int flag);

#endif
```

A.4.7 tools.h

```
/*
 * TOOLS.H - Auxiliary functions
 */

#ifndef TOOLS_H
#define TOOLS_H

#include <stdio.h>
#include <stdlib.h>

size_t log2(size_t n);
void clear_data(double *x, size_t n);
double *read_data(FILE *fp, size_t *np);
void print_data(FILE *fp, double *x, size_t n);

#endif
```

A.5 Programas Índices Acústicos

Los siguientes programas fueron realizados en Matlab.

A.5.1 Claridad C80

```
function [y,ok]=clar80(ir,fs)
% CLARIDAD (C80) EN [dB]
% [y,ok]=clar80(ir,fs)
% fs   frecuencia de muestreo en [Hz]
% ir   respuesta al impulso

limite=.08;
n=length(ir);
t=limite*fs;
if t <= n
   if \ min(ir)>= 0
```

```
        xx=ir;
   else
        xx=ir.*ir;
   end
ok=1;
y=10*log10(trapz(xx(1:t))/trapz(xx(t:n)));
else
ok=2;
end;
```

A.5.2 Curva de decaimiento

```
function cd=decurve1(ir,fs)
% CURVA DE DECAIMIENTO.
% function cd=decurve1(ir,fs)
% ir=respuesta al impulso   fs=frecuencia de muestreo en [Hz]
% cd es el vector que contiene los valores de la curva de decaimiento

n=length(ir);
  if \ min(ir)>= 0
        xx=ir;
   else
        xx=ir.*ir;
   end
xx=xx/max(xx);
Ts=1/fs;
f(n)=0.0;
for i=n-1:-1:1
   f(i)=f(i+1)+0.5*(xx(i)+xx(i+1));
end
cd=f(1:n-1);
cd=Ts*cd;
cd=10*log10(cd/max(cd));
```

A.5.3 Definición D50.

```
function [y,ok]=def50(ir,fs)
%DEFINICION (D50) EN [%] [y,ok]=def50(ir,fs)
%fs frecuencia de muestreo en [Hz],ir=respuesta al impulso

limite=.05;
n=length(ir);
t=limite*fs;
if t <= n
   if min(ir)>=0
      xx=ir;
   else
      xx=ir.*ir;
   end
ok=1;
hh=trapz(xx);
y=trapz(xx(1:t))/hh;
else
ok=2;
end;
```

A.5.4 Intervalo de Tiempo Inicial ITDG

```
function y=ITDG(ir,fs)
% INTERVALO DE TIEMPO INICIAL (ITDG) EN [s] y=ITDG(ir,fs)
```

```
% y=ITDG(ir,fs) ir es la respuesta al impulso,
% fs es la frecuencia de muestreo en Hz

xx=ir.*ir;
T=1/fs;
umb10=.1*max(xx);
valmax=umb10
t2=0;
t1=0;
i=1;
while t2 ==0

     if xx(i) >= valmax
       if t1==0
          t1=i
       else

          t2=i
       end
     end
i=i+1;
end
xx(i)
i
y=(t2-t1)*T;
```

A.5.5 Relación Señal a ruido S/N

```
function [y,ok]=SN(ir,fs)
%RELACION SEAL A RUIDO (S/N) EN [dB] function [y,ok]=SN(ir,fs)
% fs=frecuencia de muestreo en Hz
%ir=respuesta al impulso

n=length(ir);
limite1=.035;
limite2=.095;
t1=limite1*fs;
t2=limite2*fs;
      if n >= t2
         ok=1;
         if min(ir)>=0
            HH=ir;
         else
            HH=ir.*ir;
         end
         int1=trapz(HH(1:t1));
         i=(.035:1/fs:.095)';
         a=-(i-.095)/.060;
         int2=trapz(a.*HH(t1:t2));
         y=10*log10((int1+int2)/trapz(HH(t2:n)));
      else
        ok=2;
      end;
```

A.5.6 Tiempo Central

```
function y=center(ir,fs)
% TIEMPO CENTRAL (CT) EN [s]
% ir es la respuesta al impulso y fs es la frecuencia de muestreo en Hz
```

```
Ts=1/fs;
n=length(ir);
if min(ir)>=0
   xx=ir;
else
   xx=ir.*ir;
end
HH=trapz(xx);
t=(0:Ts:(n-1)*Ts)';
y=trapz(t.*xx);
y=y/HH;
```

A.5.7 Tiempo de Decaimiento Temprano EDT

```
function [y,ok]=EDT(cd,fs)
%TIEMPO DE DECAIMIENTO TEMPRANO (EDT) EN [s] EDT(cd,fs)
%cd=curva de decaimiento calculada con la respuesta al impulso
%fs=frecuencia de muestreo en Hz

n=length(cd);
Ts=1/fs;
t_5=cd<=-5;
t_15=cd<=-15;
if min(cd)<=-15
[p_5,i_5]=max(t_5);
[p_15,i_15]=max(t_15);
t=Ts*i_5:Ts:Ts*i_15;
coef=polyfit(t,cd(i_5:i_15),1);
y=abs(60/coef(1));
ok=1;
else
ok=2;
end
```

A.5.8 Tiempo de reverberación RT15

```
function [y,ok]=RT_15(cd,fs)
%TIEMPO DE REVERBERACION (RT15) EN [s]  [y,ok]=RT_15(cd,fs)
%cd=curva de decaimiento,fs=frecuencia de muesteo en Hz
n=length(cd);
Ts=1/fs;
t_5=cd<=-5;
t_25=cd<=-20;
  if min(cd)<= -20
     [p_5,i_5]=max(t_5);
     [p_25,i_25]=max(t_25);
     t=Ts*i_5:Ts:Ts*i_25;
     coef=polyfit(t,cd(i_5:i_25),1);
     y=4*abs(15/coef(1));
     ok=1;
  else
    ok=2;
    end;
```

A.5.9 Tiempo de reverberación RT_X

```
function [y,ok]=RT_X(cd,li,fs)
%TIEMPO DE REVERBERACION (RT_X) EN [s]  [y,ok]=RT_X(cd,li,fs)
%fs=frecuencia de muestreo en [Hz]
%li=limite inferior en [dB]
```

```
%cd= curva de decaimiento

n=length(cd);
t_5=cd<=-5;
t_li=cd<=-(li+5);
      if min(cd)<= -(li)
         ok=1;
         [p_5,i_5]=max(t_5);
         [p_li,i_li]=max(t_li);
         t=(i_5:i_li)/fs;
         coef=polyfit(t,cd(i_5:i_li),1);
         factor=60/li;
         y=factor*abs(li/coef(1));
      else
         ok=2;
      end
```

A.5.10 Función que filtra

```
function y=filtoct(ir,fs,fc,n)
%irfilt Respuesta al Impulso filtrada  a fc [Hz] irfilt=filtoct(ir,fs,fc,n)
%empleando un filtro butterworth digital de grado n
%fs=frecuencia de muestreo y fc=frecuencia central en Hz, n=orden del filtro
%ir=respuesta al impulso

w1=(2*fc)/(sqrt(2)*fs);
if w1 >= fs/2
irfilt=[];
end
lims=sqrt(2)*fc;
if lims > fs/2
   w2=1
else
   w2=2*sqrt(2)*fc/fs;
end
w=[w1 w2];
[b,a]=butter(n,w);
irfilt=filter(b,a,ir);
```

[15] M.R. Schroeder. New method of measuring reverberation time. *Journal of the Acoustical Society of America*, 37:409–412, 1965.

[16] M.R. Schroeder. Integrated–impulse method measuring sound decay without using impulses. *Journal of the Acoustical Society of America*, 66(2):497–500, August 1979.

[17] M.R. Schroeder. *Number Theory in Science and Communication with apps in Cryptography,Physics,Digital Inf, Computing,and self similarity.* Springer-Verlag, New York, third edition, 1997.

[18] S.Norcross and J.S. Bradley. Comparison of room impulse response measurement methods. *Canadian Acoustics*, 22(3):47–48, 1994.

[19] Spectrum Signal Processing Co. *TMS320C30 System Borad User's Manual*, August 1990.

[20] Robert D. Strum and Donald E.Kirk. *First Principles of Discrete Systems and Digital Signal Processing.* Addison Wesley, 1988.

[21] Texas Instruments. *TMS320C30 Floating Point DSP Optimizing C Compiler User's guide*, 1991.

FE DE ERRATAS

Página 16:
La ecuación (1.35) dice: $x_{xf}[n]=x[-n]*f[n]$; debe decir: $\Phi_{xf}[n]=x[-n]*f[n]$

Página 17:
La ecuación (1.41) dice: $\Phi_{fx}[n]=f[n]*x[-n]$; debe decir: $\Phi_{fx}[n]=x[n]*f[-n]$

La ecuación (1.42) dice : $y[n]=h[n]*x[n]=\sum_{m=-\infty}^{\infty}h(n)x(n-m)$;

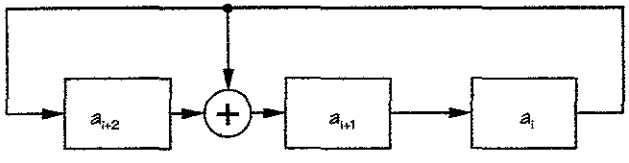
debe decir : $y[n]=h[n]*x[n]=\sum_{m=-\infty}^{\infty}h(m)x(n-m)$

Página 31:

La ecuación (2.11) dice : $C80=10\cdot\log_{10}\left[\frac{\int_0^{50ms}h^2(t)dt}{\int_0^{\infty}h^2(t)dt}\right]$;

debe decir : $C80=10\cdot\log_{10}\left[\frac{\int_0^{80ms}h^2(t)dt}{\int_0^{\infty}h^2(t)dt}\right]$

Página 52:
La dirección del registro de corrimiento de la Figura(3.9) es opuesta a la que se presenta, o sea:



Y las condiciones iniciales son $a_0=0$, $a_1=0$ y $a_2=1$.