Universidad Nacional Autónoma de México

Programa de Maestría y Doctorado en Música

Síntesis de Instrumentos Musicales Mediante Segmentación de Forma de Onda y Predicción Lineal

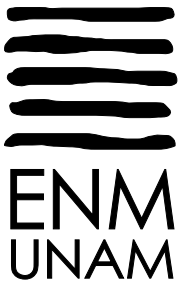
T E S I S

QUE PARA OBTENER EL GRADO DE
DOCTOR EN MÚSICA
EN EL CAMPO DE CONOCIMIENTO DE
TECNOLOGÍA MUSICAL

P R E S E N T A

Francisco Fernández del Castillo Gómez

Tutor: Dr. Felipe Orduña Bustamante



México D.F., octubre de 2012

Miembros del Jurado

Dr. Ricardo Ruiz Bullosa (presidente)

Dr. Manuel Rocha Iturbide (secretario)

Dr. Abel Herrera Camacho (vocal)

Dr. Pablo Luis Rendón Garrido (vocal)

Dr. Felipe Orduña Bustamante (vocal)

Dedicatoria

A mis tres mujeres, Marilupe, Pili y Alicia, por darme las fuerzas para seguir adelante todos los días.

Agradecimientos

Muchas gracias al Dr. Felipe Orduña por todo su apoyo y confianza en mí. Agradezco también a las siguientes personas por su participación en las pruebas auditivas:

Alfonso Meave
Ana Laura Padilla
Arturo Machuca Tzili
Bogdán Chávez
Christian Hernández
David Hernández Prián
Edgar Torres
Enedina Martínez
Ernesto Ramírez
Estela Fernández del Castillo
Flor López
Gabriela Pérez
Hugo Gutiérrez
Pablo Rendón
Raquel Pineda
Ricardo Dorantes
Roberto Velasco
Sebastián Palacios

Un agradecimiento muy especial a mi amigo Victor Pérez por sus palabras de aliento y su gran ayuda.

Resumen

En este trabajo, se propone un nuevo método de síntesis sonora, que combina la *segmentación de forma de onda* de Mitsuhashi [1] con la *predicción lineal* [2]. El problema con la *segmentación de forma de onda*, es que durante la fase de síntesis, se introducen armónicos agudos no controlables causados por la función de interpolación. Utilizando la *predicción lineal*, se pueden filtrar estos armónicos agudos para mejorar la calidad sonora. En la *predicción lineal* se utiliza, por lo general, un tren de impulsos para sintetizar sonidos con componentes casi armónicas. El inconveniente es que el tren de impulsos, no representa adecuadamente el mecanismo de excitación de algunos instrumentos musicales. Se puede sustituir el tren de impulsos, por una forma de onda segmentada, para lograr una mejor síntesis de instrumentos musicales, por medio de *predicción lineal*. Con una implementación en *Pd* [3], se demuestra que esta técnica de síntesis sonora funciona en tiempo real y consume pocos recursos de cómputo. Se ha logrado sintetizar sonidos, en tiempo real, en un ordenador con procesador AMD Turion^{TM}(1.8 GHz), con un retardo de 5.8 milisegundos y una frecuencia de muestreo de 44100 Hz. Se ha probado este modelo con polifonía de cuatro voces sin problema alguno. En un caso típico, se puede reducir aproximadamente 60 % del número de datos que representan una forma de onda, y un 80 % del número de ciclos de forma de onda que contiene la señal en sus totalidad. Estos valores son sólo ilustrativos, ya que varían mucho dependiendo del tipo de instrumento que se está sintetizando, de la frecuencia fundamental del tono y del tipo de aplicación para la que se quiera utilizar. De acuerdo con los resultados de las pruebas de audición, la calidad sonora de este modelo de síntesis, es menor al de la síntesis aditiva. En una escala absoluta, el modelo híbrido es de buena calidad, según la escala de calidad de la metodología MUSHRA [4].

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En la década de 1980, era muy difícil trabajar con audio digital en tiempo real, debido a las limitaciones del hardware disponible. En aquel tiempo, apareció una nueva y prometedora técnica de síntesis sonora. Alan D. Bernstein [6], Ellis D. Cooper y más tarde, Yasuhiro Mitsuhashi [1], desarrollaron un método de síntesis sonora llamado segmentación de forma de onda[1]. Consiste en crear sonidos mediante la concatenación de fragmentos de forma onda para crear formas de onda más grandes. La curva de cada segmento de forma de onda puede ser definida por algún tipo de interpolación para unir los extremos de cada segmento. Según Mitsuhashi, esta técnica es comparable a la síntesis aditiva, con la ventaja de que requiere de menos operaciones. Esto resulta en un ahorro importante de hardware y tiempo de computo [1].

La síntesis aditiva, por otra parte, es una de las técnicas de síntesis sonora más viejas y más estudiadas [7]. El concepto de síntesis aditiva se conoce desde la Edad Media. Se aplicaba a la combinación registros para cambiar el timbre de un órgano. Esta técnica ha sido utilizada desde los inicios de la música electrónica, a principios del siglo XX. Se basa en la adición de ondas senoidales para crear ondas más complejas[2]. La síntesis aditiva ha demostrado ser una poderosa herramienta para simular sonidos, así como para modificarlos. Para simular un sonido por medio de síntesis aditiva es necesario primero realizar un análisis del sonido que se desea simular. Este análisis genera una gran cantidad de datos que serán utilizados en el proceso de síntesis.

[1] En inglés: *Piecewise Interpolation.*

[2] En realidad, se pueden superponer cualquier tipo de ondas pero tradicionalmente se superponen ondas senoidales porque están más relacionadas con nuestra percepción

La síntesis aditiva tiene la ventaja de proporcionar *control completo e independiente* sobre el comportamiento de cada componente espectral[3]. Sin embargo, la gran cantidad de datos requerida para controlar el espectro, hace más complicado el diseño de sonidos para su uso en la música electrónica o la computación musical. Además, la síntesis aditiva presenta ciertas limitaciones desde el punto de vista computacional, ya que se necesita una gran cantidad de osciladores, memoria, datos de control y velocidad de procesamiento para lograr sintetizar con éxito un sonido complejo con espectro dinámico[4]. Estas limitaciones resultan más evidentes cuando se quiere sintetizar un sonido en tiempo real. A pesar de que estas limitaciones son cada vez de menor importancia, gracias al desarrollo de hardware cada vez más potente, la cantidad de parámetros necesarios para la síntesis aditiva hace que en algunos casos esta técnica sea poco práctica para sintetizar sonidos complejos con espectro dinámico.

Al comienzo del presente proyecto, se implementaron los métodos de Mitsuhahsi en Octave [8, 9]. Se descubrió que la segmentación de forma de onda, a diferencia de la síntesis aditiva, genera distorsiones en la forma de onda que dependen de la función de interpolación, disminuyendo la calidad sonora del sonido sintetizado. Este hecho, pone a la segmentación de forma de onda en desventaja con respecto a la síntesis aditiva, a pesar del ahorro de recursos de cómputo.

La segmentación de forma de onda, en efecto, consume menos recursos de cómputo y puede ser utilizada en tiempo real en una computadora personal, pero no produce resultados similares a la síntesis aditiva, ya que su calidad sonora es menor. Afortunadamente, es posible eliminar estas distorsiones utilizando filtros para restituir la envolvente espectral del tono original. Una manera de obtener estos filtros es por medio de predicción lineal.

La contribución original de este trabajo, consiste en la implementación de un modelo de síntesis híbrido que combina la segmentación de forma de onda según Mitsuhashi, con la predicción lineal. Este modelo de síntesis sonora fue implementado en Pd[5]. Al final, se realizaron pruebas subjetivas de audición para evaluar su calidad sonora.

[3]Dentro de un cierto rango de frecuencias, dependiendo de la frecuencia de muestreo y de la resolución del sistema.

[4]Espectro dinámico se refiere a que las características espectrales del sonido evolucionan en el transcurso de su duración.

[5]Pd o Pure Data [3, 10] es un entorno gráfico de programación en tiempo real para procesamiento de audio, gráficos y vídeo.

Con esta implementación se pretende demostrar lo siguiente:

- La síntesis por segmentación de forma de onda se puede realizar en una computadora personal en tiempo real.
- La predicción lineal se puede usar eficazmente para eliminar las distorsiones producidas por la síntesis por segmentación de forma de onda.
- La segmentación de forma de onda se puede utilizar como señal de excitación para la predicción lineal, mejorando su utilidad en la síntesis de instrumentos musicales.

La importancia de este trabajo recae principalmente en sus aplicaciones a la computación musical. En el ámbito de la síntesis sonora, siempre se ha buscado un compromiso entre la calidad sonora, el espacio de almacenamiento y el tiempo de cómputo. El modelo híbrido presentado tiene la ventaja de consumir pocos recursos de cómputo y requiere poco espacio de memoria. Por su escalabilidad, el modelo presentado puede utilizarse en situaciones en que se necesita buena calidad sonora o en situaciones donde los recursos son limitados como es el caso de los dispositivos portátiles. Por su representación paramétrica, este modelo puede utilizarse para la creación de nuevas sonoridades como la combinación de características entre instrumentos y la realización de metamorfosis de instrumentos musicales, para mutar dinámicamente entre dos timbres diferentes. Otras contribuciones originales de este trabajo incluyen la combinación de la segmentación de forma de onda con la predicción lineal, la detección automática del orden de predicción y los métodos de selección del segmento de reproducción cíclica. Otras líneas de investigación que se desprenden de este trabajo son la metamorfosis de instrumentos musicales[6] y la separación de un tono en ruido y líneas espectrales por medio de predicción lineal.

En el capítulo 2 se explica con detalle la segmentación de forma de onda según Mitsuhashi. El capítulo comienza con un poco de historia y antecedentes. Después, se describen los algoritmos de análisis y síntesis. El capítulo 3 está dedicado a la predicción lineal. Al principio de este capítulo, se da un panorama de la historia y la literatura existente. Enseguida, se explican los algoritmos de análisis y síntesis. Se incluye una explicación de la obtención de los parámetros del filtro, cálculo del factor de ganancia y la estimación

[6]Se propone hacer metamorfosis de instrumentos musicales interpolando parámetros de la síntesis por segmentación de forma de onda y predicción lineal.

del orden óptimo de predicción. También se incluyen los detalles de implementación de un filtro de polos con estructura de malla (*lattice*) para la fase de síntesis. En el capítulo 4 se explican las diferentes maneras en que se pueden combinar la segmentación de forma de onda con la predicción lineal, que constituye la contribución original de este trabajo. La implementación en Pd de los algoritmos de síntesis, es presentada en el capítulo 5. Aquí se explica el funcionamiento de los objetos computacionales creados para Pd y se dan ejemplos de su utilización. En el capítulo 6 se describe la metodología utilizada para evaluar, por medio de pruebas subjetivas, la calidad sonora de estos modelos de síntesis sonora. Al final del capítulo, se presentan los resultados de las pruebas de audición y se presentan algunas conclusiones. Finalmente, en el capítulo 7, se presentan las conclusiones de este trabajo.

Capítulo 2

Segmentación de forma de onda

2.1. Antecedentes

Esta técnica de síntesis, surge alrededor de 1980. Era la época en que empezaban a salir al mercado las primeras computadoras personales. El hardware disponible para hacer música electrónica tenía muchas limitaciones en comparación con las herramientas tecnológicas que existen actualmente. Había la necesidad de crear algoritmos eficientes para poder sintetizar sonidos en tiempo real. Entre los métodos de síntesis sonora más populares estaba la síntesis aditiva. Este método consiste en superponer o sumar ondas seonidales para generar el sonido deseado. Para cada senoidal, es necesario controlar, por lo menos, dos parámetros: la frecuencia y la amplitud. Si se quiere crear un sonido realista, los valores de frecuencia y amplitud deben cambiar a lo largo de la duración del sonido. Es así que la síntesis aditiva requiere de la manipulación de un gran número de parámetros y osciladores para poder generar el sonido de un solo instrumento. Las limitaciones del hardware hacían de la síntesis aditiva una técnica poco práctica para ciertas aplicaciones. La segmentación de forma de onda surge como una alternativa a la síntesis aditiva para sintetizar sonidos musicales en tiempo real. Esta técnica consiste en definir una forma de onda por medio de segmentos. Estos segmentos pueden ser definidos por cualquier función de interpolación. Por lo general se utilizan líneas rectas. A la unión de estos segmentos les llamamos puntos o nodos. De manera similar a la síntesis aditiva, para lograr un sonido realista con espectro dinámico, es necesario que la forma de onda cambie de forma de un ciclo a otro. Esto se logra cambiando de lugar los nodos que

la definen en cada ciclo. Como vemos, la síntesis aditiva se controla en el dominio de la frecuencia, mientras que la segmentación de forma de onda se controla en el dominio del tiempo.

Alan D. Bernstein y Ellis D. Cooper [6] desarrollaron una técnica llamada segmentación lineal de forma de onda[1]. Esta consistía en definir una forma de onda por medio de puntos[2] unidos por líneas rectas , tal como se muestra en la figura 2.1.

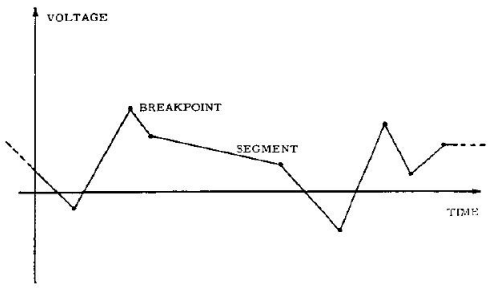


Figura 2.1: Función continua de segmentos lineales

Para implementar esta técnica, Bernstein y Cooper diseñaron un módulo generador de segmentos lineales al que llamaron módulo PL[3]. Este aparato controlaba la abscisa y ordenada de cada nodo por medio de controladores de voltaje de frecuencia variable. Los módulos podían conectarse entre sí para generar una gran variedad de sonidos complejos. Era posible producir, no sólo componentes armónicas, sino también inarmónicas. Estos módulos también se podían usar para controlar parámetros de otros tipos de síntesis. Por ejemplo, un módulo podría servir para controlar la envolvente de una señal o para modular su frecuencia. La figura 2.2 muestra un esquema del módulo generador de armónicos diseñado por Bernstein y Cooper [6].

Basado en el trabajo de Bernstein y Cooper, Yasuhiro Mitsuhashi [1] propone el uso de otras funciones interpoladoras además de las líneas rectas. Mitsuhashi desarrolla un modelo unificado que permite el uso de una función arbitraria para interpolar entre los nodos. La función interpoladora debía estar normalizada a +1 en ambos ejes. Según Mitsuhashi [1], «la técnica propuesta por Bernstein y Cooper no había sido usada extensivamente en

[1]En inglés: *Piecewise Interpolation Technique for Audio Signal Synthesis*

[2]Los términos *nodos* y *puntos* se usarán en este trabajo de manera indistinta.

[3]En inglés: *PL module* o *piecewise-linear generator.*

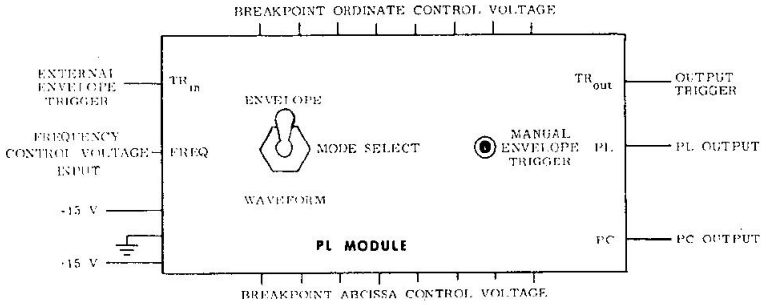


Figura 2.2: Diagrama funcional del Modulo PL de Bernstein y Cooper

el ámbito de la música electrónica debido a que la interpolación lineal producía una gran cantidad de armónicos no controlables». En otras palabras, al conectar los nodos por medio de líneas rectas, se producían distorsiones en la forma de onda generada. Para disminuir este efecto, Mitsuhashi propone el uso de otras funciones interpoladoras, como la función *medio-coseno* o *polinomios de tercer grado*. Mitsuhashi considera que su modelo de síntesis es equivalente a la *síntesis aditiva* en cuanto al número de parámetros necesarios para controlar un cierto número de armónicos. Esto es porque la síntesis aditiva y la segmentación de forma de onda se relacionan por medio de la transformada discreta de Fourier. En otras palabras, controlar un número N de nodos de la forma de onda segmentada en el dominio del tiempo, equivale a controlar N parámetros (N/2 armónicos) de la síntesis aditiva en el dominio de la frecuencia. La ventaja de la *segmentación de forma de onda* sobre la *síntesis aditiva* es su eficiencia en términos de velocidad de cómputo y su fácil implementación. Según Mitsuhashi [1], «siendo que la función *medio-coseno* puede ser aproximada exclusivamente con sumas [11], es posible utilizar esta técnica para sintetizar sonidos en tiempo real, con una computadora digital». La desventaja es que el espectro de la forma de onda, depende mucho de la función interpoladora, especialmente en los armónicos agudos.

En años recientes, Nick Collins [12, 13] desarrolló un programa llamado *SplineSynth*. Este es un sintetizador en software que funciona en tiempo real y está basado en la *segmentación de forma de onda*. Collins, a diferencia de Mitsuhashi, utiliza como funciones interpoladoras, *segmentarias cúbicas* o *splines*. La motivación de Collins para crear el *SplineSynth*, era encontrar un método para realizar metamorfosis de sonidos en el dominio del tiempo,

sin tener que recurrir al *crossfade*[4]. Otra motivación de Collins para el uso de *segmentarias*, es ofrecer al músico una interfaz gráfica que le permita manipular audio digital a bajo nivel. El uso de *segmentarias* permite editar gráficamente una forma de onda, de manera similar a como se editan gráficos vectoriales en el ámbito del diseño gráfico. Aunque podrían utilizarse otros tipos de interpolación para unir los segmentos, Collins utiliza las segmentarias cúbicas por la suave transición que se logra entre los segmentos.

Collins propone el uso de sus técnicas de síntesis para crear nuevas sonoridades, en lugar de sintetizar sonidos existentes. Esto se debe a que «la predicción de componentes espectrales a partir de *segmentarias cúbicas*, es muy difícil de calcular analíticamente» [13].

2.2. Explicación del modelo de Mitsuhashi

La *síntesis por segmentación de forma de onda* consiste en definir un ciclo de forma de onda por medio de puntos y una función interpoladora. La figura 2.3 muestra dicha configuración. Note que la ordenada del último punto (x_N, y_N) y la ordenada del primer punto (x_0, y_0) son iguales $y_N = y_0$. Esto es porque la forma de onda se supone periódica, quiere decir que volverá a empezar cuando se haya llegado al final del ciclo. En el modelo de Mitsuhashi [1], los nodos están uniformemente espaciados, siendo la separación entre cada punto $2\pi/N$. Para completar la forma de onda, se conectan los puntos por medio de una función $f(x)$ normalizada a $+1$ en ambos ejes, de tal forma que la función interpoladora deberá cumplir las condiciones $f(0) = 0$ y $f(1) = 1$. La figura 2.4 es un ejemplo de una función normalizada a $+1$ en ambos ejes. De esta manera se puede representar un ciclo de forma de onda $g(x)$, interpolando los nodos por medio de $f(x)$ usando la siguiente expresión:

$$\begin{aligned} g(x) &= (y_{n-1} - y_n) f\left(\tfrac{N}{2\pi}x - n\right) + y_n, \\ &\tfrac{2\pi}{N}n \leq x \leq \tfrac{2\pi}{N}(n+1), \\ &n = 1, 2, 3, \ldots, N-1 \end{aligned} \tag{2.1}$$

[4] *Crossfade* es un término en inglés que se refiere a una técnica de edición sonora consistente en hacer una transición suave entre dos sonidos. Esta transición se logra variando gradualmente la amplitud de ambos sonidos.

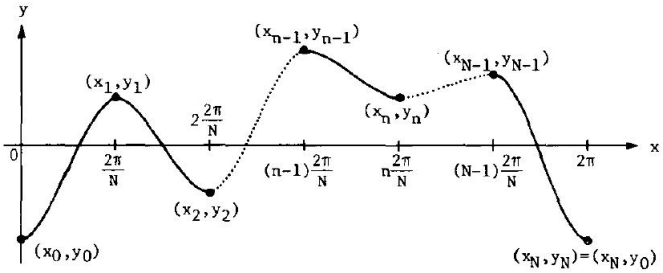


Figura 2.3: Configuración de nodos

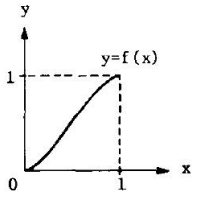


Figura 2.4: Función interpoladora normalizada

Podemos aproximar una forma de onda, utilizando la segmentación de forma de onda, a partir del espectro de una señal de audio. Si se conocen los coeficientes α_k de Fourier de la forma de onda, se pueden calcular las ordenadas y_n de los puntos por medio de una DFT inversa:

$$y_n = \frac{1}{N}\sum_{k=0}^{N-1} Y(k)e^{j(2\pi/N)kn} \tag{2.2}$$

donde $Y(k)$ depende de la función interpoladora. Para el caso de una función lineal $f(x) = x$, $Y(k) = \alpha_k/b_k$ y b_k está dada por:

$$b_k = \begin{cases} \frac{1}{N}, & k = 0 \\ \frac{4N\sin^2(\pi k/N)}{(2\pi k)^2}, & k \neq 0 \end{cases} \tag{2.3}$$

Los coeficientes de Fourier α_k para otras funciones interpoladoras están descritos en el apéndice B.

Si se quiere tener un número N de nodos para definir un ciclo de forma de onda, es necesario especificar $N/2$ coeficientes α_k de Fourier. [1]

Veamos un ejemplo. En la figura 2.5 se muestra un ciclo de forma de onda de un tono de guitarra. Su espectro en frecuencia se muestra en la figura 2.6.

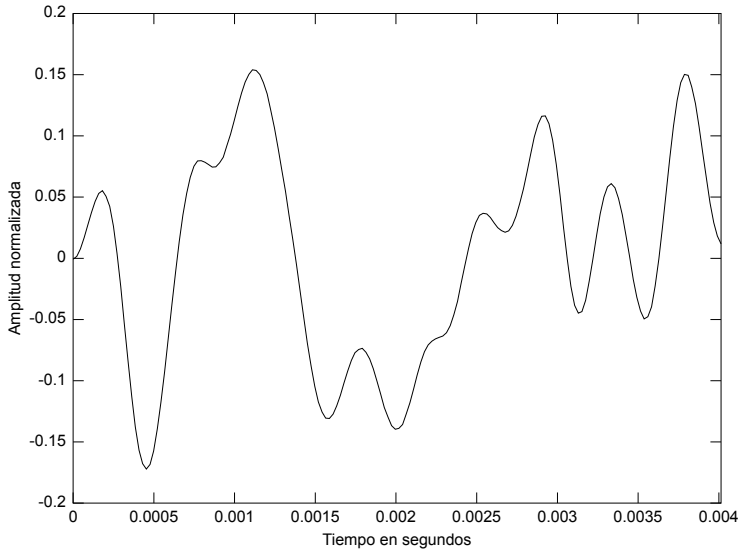


Figura 2.5: Un ciclo de forma de onda.

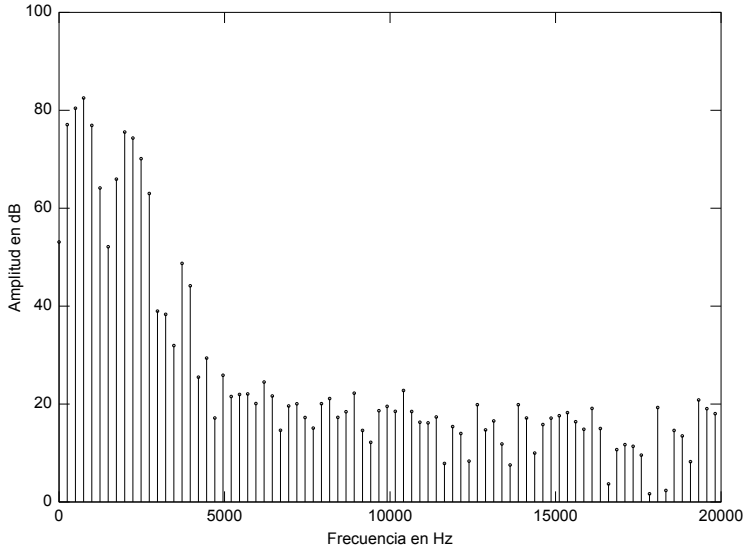


Figura 2.6: Espectro armónico de la forma de onda.

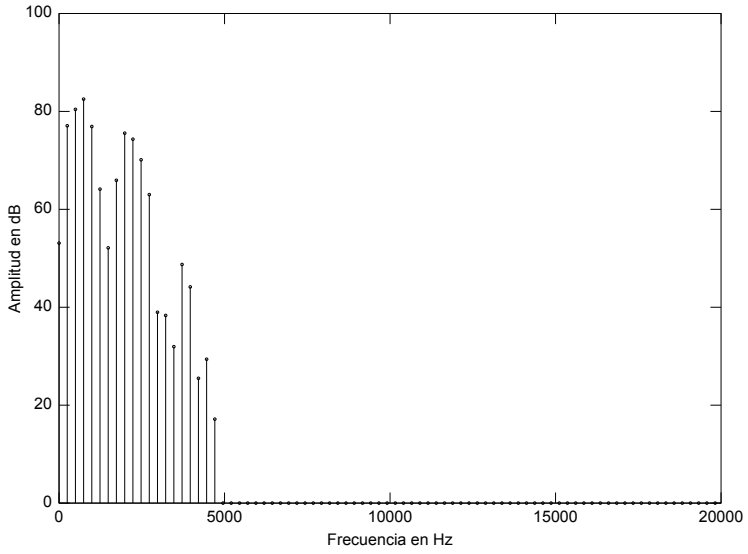


Figura 2.7: Primeras 20 componentes del espectro armónico.

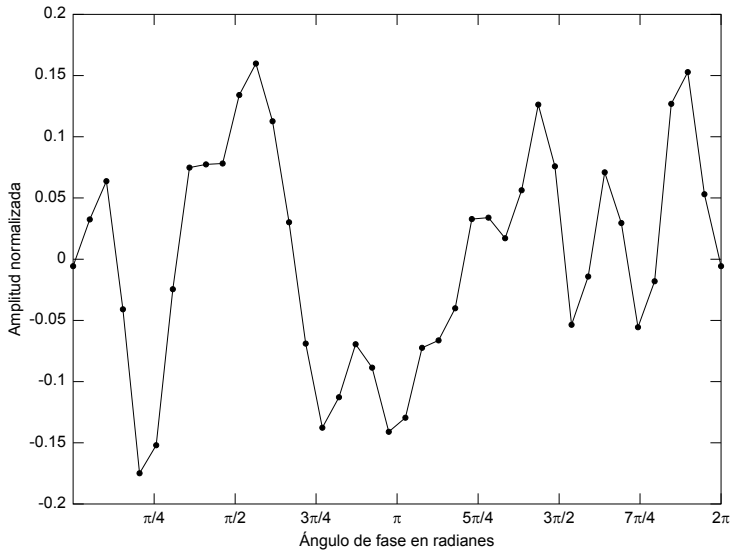


Figura 2.8: Señal segmentada.

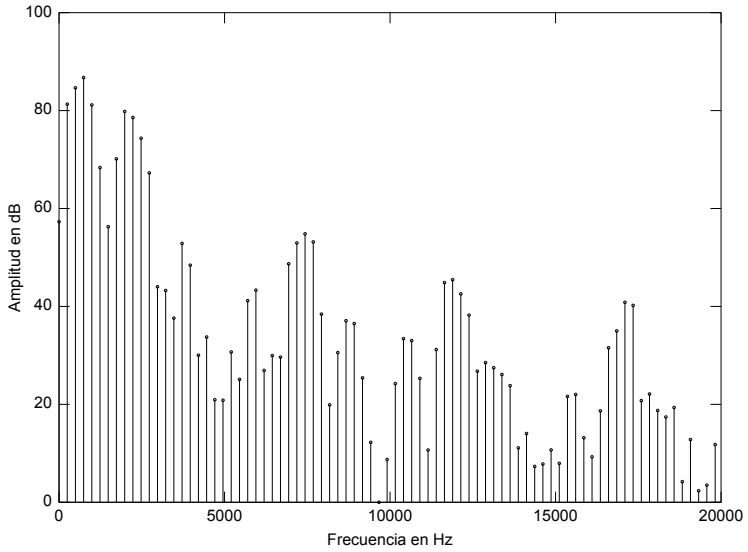


Figura 2.9: Espectro armónico de la señal segmentada.

Supongamos que queremos sintetizar este sonido. Por motivos ilustrativos, tomaremos un número arbitrario de componentes espectrales, por ejemplo 20 armónicos. La figura 2.7 muestra las primeras 20 componentes, el resto se ha eliminado. Con estos datos, se obtienen los nodos mediante el método de Mitsuhashi. La forma de onda segmentada, interpolada por líneas rectas, se muestra en la figura 2.8. Si comparamos las figuras 2.6, 2.7 y 2.9, nos daremos cuenta de que las primeros 20 componentes de la señal reconstruida, coinciden con las de la señal original. El resto de las componentes, que son distintas, corresponden a los armónicos no controlables. Los armónicos no controlables que aparecen en la gráfica de la figura 2.9 son introducidos por la función interpoladora. Para mejorar esta señal, sería necesario filtrar las componentes no controlables (en este caso, todas las que están después del armónico 20) de alguna manera.

2.3. Análisis

A pesar de que la implementación de la *segmentación de forma de onda* en su fase de síntesis es relativamente simple, al igual que la teoría, la fase de análisis presenta varias dificultades que serán abordadas en detalle en esta sección.

2.3.1. Tamaño de la ventana de análisis

Para hacer el análisis de una señal de audio, se necesita determinar el tamaño y forma de la ventana de análisis. Las opciones son muchas. Podemos, por ejemplo, definir un tamaño de ventana fijo para analizar la señal en pequeños segmentos de tiempo, o quizá, cambiar el tamaño de la ventana de acuerdo al tamaño de cada ciclo de la forma de onda. Los segmentos de análisis podrían o no ser traslapados. La forma de la ventana también influye en los resultados del análisis.

Las características de la ventana como el tamaño, la forma y si es fija o variable, dependen en parte de si se quiere reducir información a la hora de almacenar los parámetros de Mitsuhashi. Los esquemas de compresión de audio se basan en dos principios: [14] la redundancia estadística de la señal y su irrelevancia perceptual. La redundancia estadística se basa en que, en una serie de datos, puede haber valores o patrones de valores que se repiten. En este caso, se puede reducir la información, eliminando las copias repetidas. La irrelevancia perceptual, consiste en eliminar los datos que son irrelevantes para nuestra percepción. Por ejemplo, podríamos eliminar las frecuencias inaudibles para el ser humano. Las redundancias en una señal de audio se encuentran más fácilmente en el dominio de la frecuencia. Sin embargo, la *segmentación de forma de onda* se realiza en el dominio del tiempo, donde las redundancias son más difíciles de encontrar porque dependen de la periodicidad de la señal. Si se logra que la ventana de análisis coincida con cada ciclo de la señal de audio, los valores obtenidos en cada segmento de análisis serán muy parecidos y se podrá almacenar la información de manera compacta. Por el contrario, si la ventana no coincide con cada ciclo de la forma de onda, los datos obtenidos serán muy cambiantes y habrá que almacenar más información.

Observe la figura 2.10, en ella se muestran cuatro ciclos de forma de onda de un tono de guitarra muestreado a $44,100\,\text{Hz}$. El tono de guitarra tiene una *frecuencia fundamental* $f_0 = 330.9\,\text{Hz}$. Por lo tanto el periodo de un ciclo de forma de onda es $T = 3.02\,\text{ms}$. Suponga que para analizar esta señal, se escoge una ventana rectangular cuyo tamaño sea igual al tamaño del periodo $T_w = 3.02\,\text{ms}$ que equivale a 133 muestras. Las rayas verticales muestran los límites de cada ventana. Note que la forma de onda en cada ventana es muy parecida. En la figura 2.11 se muestran los nodos calculados por medio del modelo de Mitsuhashi. La posición de cada nodo no cambia mucho de una ventana a otra. La evolución de la posición de los nodos a lo largo de las

cuatro ventanas se muestra en la figura 2.12. Solamente se muestran seis de los veinte puntos para mayor claridad. A este tipo de análisis le llamaremos *análisis sincronizado al periodo*.

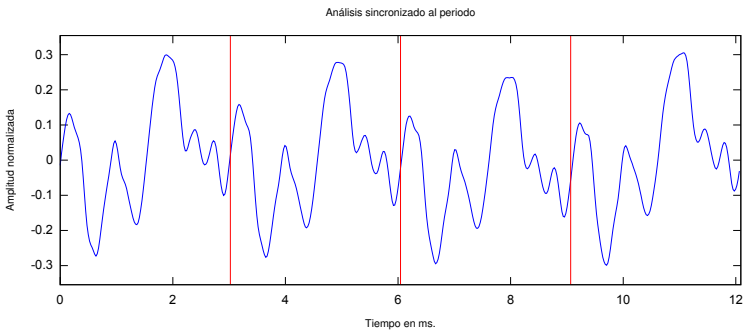


Figura 2.10: Tamaño de la ventana sincronizado al periodo.

Ahora discutiremos el caso de un análisis **no** sincronizado al periodo, con una ventana de tamaño fijo. La figura 2.13 muestra otra vez los mismos cuatro ciclos de un tono de guitarra. La diferencia es que la ventana es ahora de 128 muestras que equivalen a $T_w = 2.9\,\text{ms}$. Otra vez las rayas verticales muestran los límites de cada ventana. Ahora la forma de onda en cada ventana difiere un poco más, y además sobra un segmento pequeño al final, ya que las cuatro ventanas no coinciden con los cuatro ciclos. En la figura 2.14 se muestran los nodos que resultan del análisis. La posición de los puntos cambia mucho más que en el caso anterior. La evolución de las ordenadas de cada punto se muestra en la figura 2.15. Note que las líneas que corresponden a cada nodo son más abruptas. Este tipo de análisis no permite comprimir la información, pero tiene la ventaja de ser más práctica y fácil de implementar, pues no se necesita conocer la *frecuencia fundamental* de la señal de audio con anticipación. Además, se puede aplicar a señales que no tengan necesariamente un carácter tonal, sino percusivo, como el sonido de un platillo, un tambor o una campana.

2.3.2. Detección de la periodicidad

Para poder hacer un análisis sincronizado al periodo, es necesario determinar la *frecuencia fundamental*[5] de la señal de audio que se quiere analizar.

[5] *Periodicidad*, *altura* y *frecuencia fundamental* no son lo mismo, pero están íntimamente relacionados. En esta sección se utilizarán estos términos indistintamente.

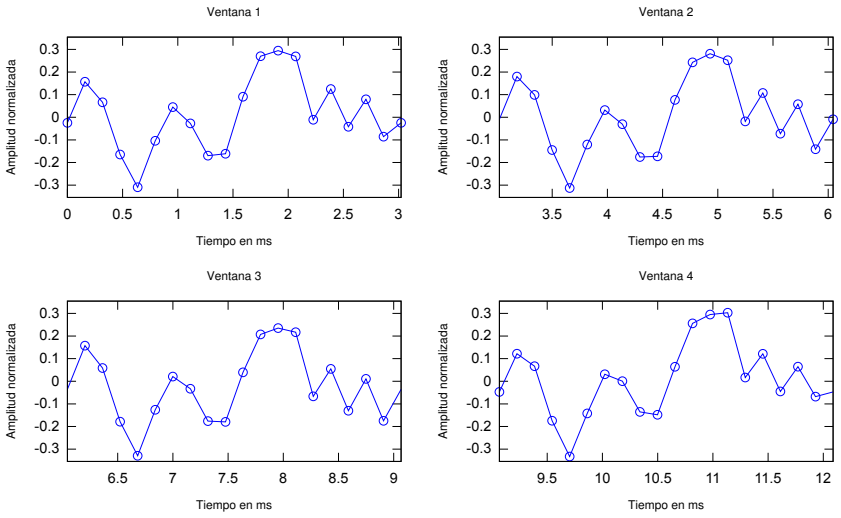


Figura 2.11: Nodos calculados con el modelo de Mitsuhashi por cada ventana de análisis. El tamaño de la ventana está sincronizado al periodo de la forma de onda.

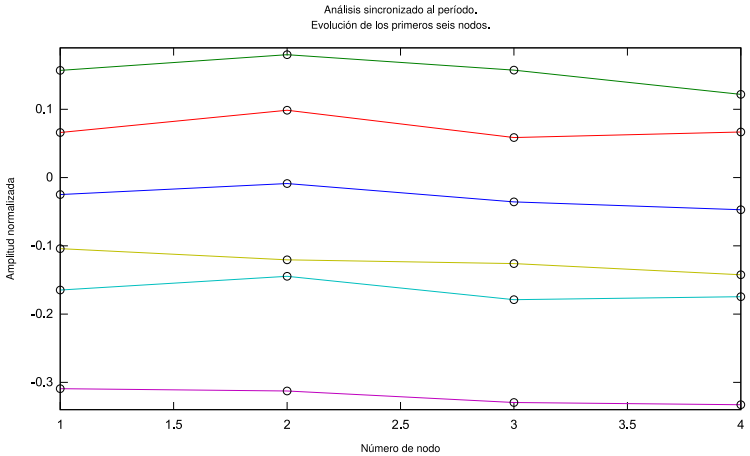


Figura 2.12: Evolución de cada nodo. Cuando la ventana de análisis está sincronizada al periodo, las ordenadas de los nodos tienen variaciones pequeñas. Por esta razón su evolución es poco variable.

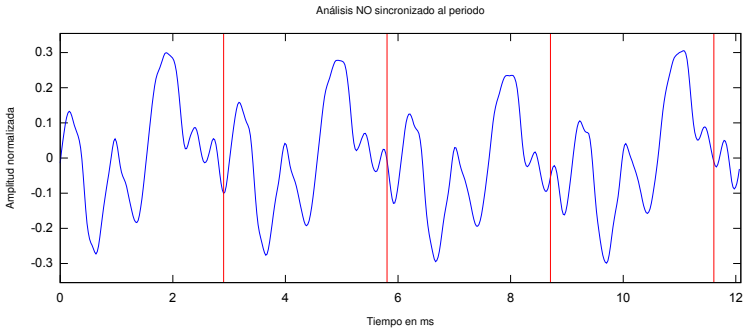


Figura 2.13: Tamaño de la ventana NO sincronizado al periodo.

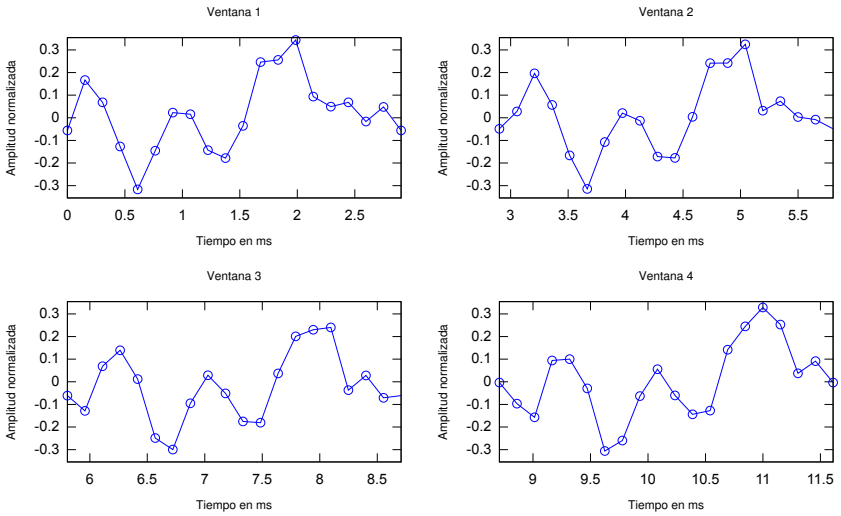


Figura 2.14: Nodos calculados con el modelo de Mitsuhashi por cada ventana de análisis. El tamaño de la ventana NO está sincronizado al periodo de la forma de onda.

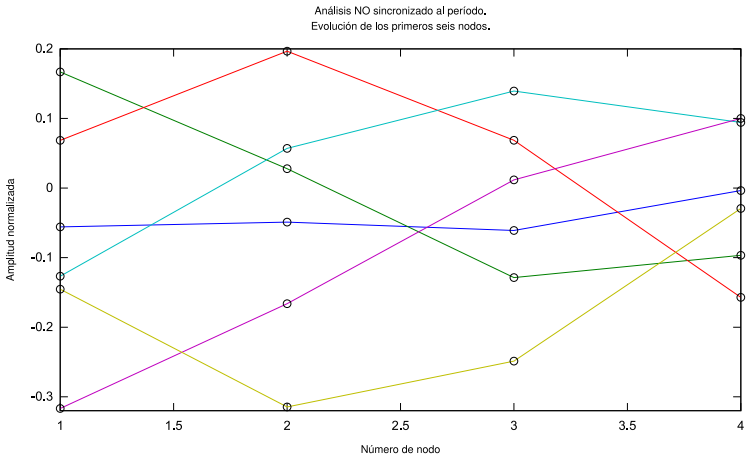


Figura 2.15: Evolución de cada nodo. Cuando la ventana de análisis **no** está sincronizada al periodo, las ordenadas de los nodos varían mucho. Por esta razón su evolución es muy variable.

La detección de la altura ha sido un tema muy estudiado, pero continua siendo un problema difícil de resolver. La mayoría de los algoritmos para la detección de la altura han surgido de investigaciones relacionadas con el procesamiento de voz [7, 15–18]. Aunque ningún detector de altura es cien por ciento preciso, se han logrado buenos resultados con ciertos algoritmos computacionalmente intensos que no funcionan en tiempo real. Algunos detectores necesitan ajustarse a las características de la señal que se quiere analizar, ya que los parámetros pueden cambiar si el algoritmo se aplica, por ejemplo, a un cierto instrumento o rango de frecuencias. Si se tiene un conocimiento previo de las características de la señal de entrada, la detección de la periodicidad se puede facilitar. Este conocimiento permite establecer los parámetros que harán que el análisis converja a los resultados que se están buscando. Esto quiere decir, que en la mayoría de los casos, la detección de la *frecuencia fundamental* de un sonido, es dependiente del tipo de instrumento que se quiere analizar.

Existen muchos algoritmos para la estimación de la *frecuencia fundamental*. Entre los mas usados se encuentran la *autocorrelación*, el *análisis cepstrum*, y los del *dominio del tiempo* que miden la distancia entre puntos *máximos*, *mínimos* y *cruces de cero*. Entre los trabajos más recientes sobre el tema están los de Paul Boersma [16], Jakub Wroblewski y Alicja

Wieczorkowska [17]. El método de Boersma está basado en autocorrelación y está implementado en su programa de análisis y síntesis de voz llamado *praat* [19]. Este algoritmo se usa en combinación con la Predicción Lineal que se explica en el capítulo 3. Una ventaja del método de Boersma es que al mismo tiempo que se obtiene la *frecuencia fundamental* de un sonido, se obtiene su grado de periodicidad[6]. El grado de periodicidad indica, en un rango de 0 a 1, que tan periódica o ruidosa es una señal. Por ejemplo, una señal periódica con un poco de ruido de fondo, arrojaría un valor cercano a 1. El grado de periodicidad se utiliza, en el ámbito de la síntesis de voz, para determinar si un sonido es una vocal (señal periódica) o una consonante (señal ruidosa). El método de autocorrelación puede fallar según el timbre del instrumento que se está analizando. Algunos instrumentos musicales, o incluso la voz humana, pueden presentar formantes o resonancias cuya potencia es mayor a la de la frecuencia fundamental. En este caso, el método de autocorrelación puede fallar, dándonos como resultado, por ejemplo, algún múltiplo de la frecuencia fundamental. Wroblewski y Wieczorkowska, desarrollaron un método multi-algorítmico que obtiene la *frecuencia fundamental* de un sonido sin importar su timbre. En este método se combinan varios algoritmos clásicos de detección de la altura. Así, se obtienen varios candidatos para la *frecuencia fundamental*. Al final se toma la decisión de cuál es el mejor candidato por medio de algoritmos inteligentes para clasificación de instrumentos musicales. El sistema determina las características tímbricas del instrumento y basado en ello determina la *frecuencia fundamental* más probable. Con esta técnica se ha logrado un 98 % de precisión, sin embargo, es computacionalmente intensa y requiere del entrenamiento del sistema para cargar los descriptores de varios instrumentos. En este trabajo se emplea el algoritmo de Boersma [16] que se explica a continuación.

Autocorrelación.

La Autocorrelación es una herramienta matemática que sirve para encontrar patrones repetitivos en una serie de datos. Esto nos permite, por ejemplo, determinar el grado de periodicidad de una señal o encontrar la *frecuencia fundamental* de un sonido con componentes armónicas. De manera intuitiva, la autocorrelación es una función que compara observaciones de una señal en distintos instantes de tiempo. En otras palabras, la autocorrelación mide

[6]En inglés: *Harmonic Strength*.

las similitudes que tiene una señal con versiones de sí misma desfasadas en el tiempo. La variable independiente de la autocorrelación es la diferencia de tiempo que hay entre la señal original y su versión desfasada. A esta variable le llamaremos desfase o retardo, por eso decimos que la autocorrelación está definida en el dominio de los retardos o del desfase[7]

Para una señal $x(t)$ estacionaria[8], la autocorrelación continua $R_x(\tau)$ con un desfase τ está definida como [16]:

$$R_x(\tau) \equiv \int_{-\infty}^{\infty} x(t)x(t+\tau)dt \tag{2.4}$$

En el caso de una señal muestreada $x[n]$, la autocorrelación discreta $R_x[j]$ con un desfase j se puede calcular de la siguiente manera:

$$R_x[j] = \sum_{n=0}^{N} x[n]x[n+j] \tag{2.5}$$

donde n es el índice de las muestras de la señal de entrada y j está en el intervalo $0 < j \leq N$. En la práctica, el cálculo directo de la autocorrelación toma mucho tiempo. En nuestra implementación, la autocorrelación es calculada haciendo uso de la Transformada Rápida de Fourier. Consulte la línea 28 del listado C.1, para ver la implementación del cálculo de la autocorrelación en Octave.

En ocasiones, es conveniente normalizar la función de autocorrelación para obtener valores entre 0 y 1. La autocorrelación continua normalizada $r_x(\tau)$ para un desfase τ se calcula como:

$$r_x[\tau] = \frac{R_x(\tau)}{R_x(0)} \tag{2.6}$$

y la autocorrelación discreta normalizada está dada por:

$$r_x[j] = \frac{r_x[j]}{r_x[0]} \tag{2.7}$$

[7]En inglés *lag domain.*. Otros autores prefieren referirse al dominio de la autocorrelación como *dominio del tiempo.*

[8]Estacionaria quiere decir que es estadísticamente constante dentro de su dominio. En el caso de una señal de audio, se asume que el espectro de la señal no cambia dentro de la ventana de análisis.

Con las figuras 2.16 y 2.17 se ejemplifica gráficamente el proceso para calcular la autocorrelación discreta. La columna izquierda de la figura 2.16 muestra dos curvas. Una curva corresponde a la señal original $x[n]$, y la otra curva corresponde a la misma señal desfasada j muestras $x[n+j]$. Note que mientras una permanece fija, la otra se va moviendo de acuerdo al desfase j. La columna de la derecha muestra las gráficas que corresponden al producto, punto a punto, de estas dos señales $x[n]x[n+j]$. Para obtener el valor de la autocorrelación $R_x[j]$ para un desfase determinado j, se tienen que sumar los valores derivados del producto. En la figura 2.17 se muestra la autocorrelación que resulta de repetir este proceso por cada desfase j.

El valor máximo de la autocorrelación $R_x[j_{gmax}]$ se obtiene cuando se compara la señal original $x[n]$ consigo misma y sin desfase $x[n+0]$, esto es porque la señal es idéntica a sí misma. La autocorrelación normalizada r_x para un retardo $j = 0$ tiene un valor $r_x[0] = 1$. Podemos interpretar los valores de la autocorrelación normalizada, que van de 0 a 1, como el grado de similitud que tiene la señal original $x[n]$ con su versión retardada $x[n+j]$. La función de autocorrelación, siempre tiene su valor más alto o máximo global en $j = 0$, esto se debe a que el producto de una señal por sí misma $x[n]^2$, produce únicamente valores positivos, que al sumarse, darán el valor máximo posible de la autocorrelación. Observe que en la figura 2.17, el punto mas alto se encuentra en $j = 0$. Conforme la señal retardada se va moviendo, el grado de similitud que tiene con la señal original va disminuyendo hasta llegar a 0. La autocorrelación normalizada puede tener valores negativos, siendo -1 el menor valor posible. Cuando la autocorrelación normalizada ha llegado a -1 significa que la señal original es simétrica con respecto al punto $x[j]$ y se cumple que $x[n] = -x[n+j]$.

Hasta ahora no hemos hablado de la relación que hay entre la autocorrelación y la periodicidad de una señal. Supongamos que la señal de nuestro ejemplo es periódica y se repite una y otra vez. Si la señal se va desfasando poco a poco, llegará un momento en que coincida nuevamente con ella misma y los valores de la autocorrelación normalizada irán acercándose otra vez a 1. Si la señal se sigue desfasando, los valores volverán a tender a 0 y más adelante regresarán a 1, y así sucesivamente. Como podemos ver, la autocorrelación de una señal periódica es también periódica, y tendrá otros máximos globales $r_x[j_{gmax}]$ cuya separación entre ellos corresponde al periodo T_0 de la señal $x[n]$. Por consiguiente, los máximos globales en la autocorrelación de una señal periódica se encontrarán en $r_x[nT_0]$ y la *frecuencia fundamental* estará definida por $F_0 = 1/T_0$.

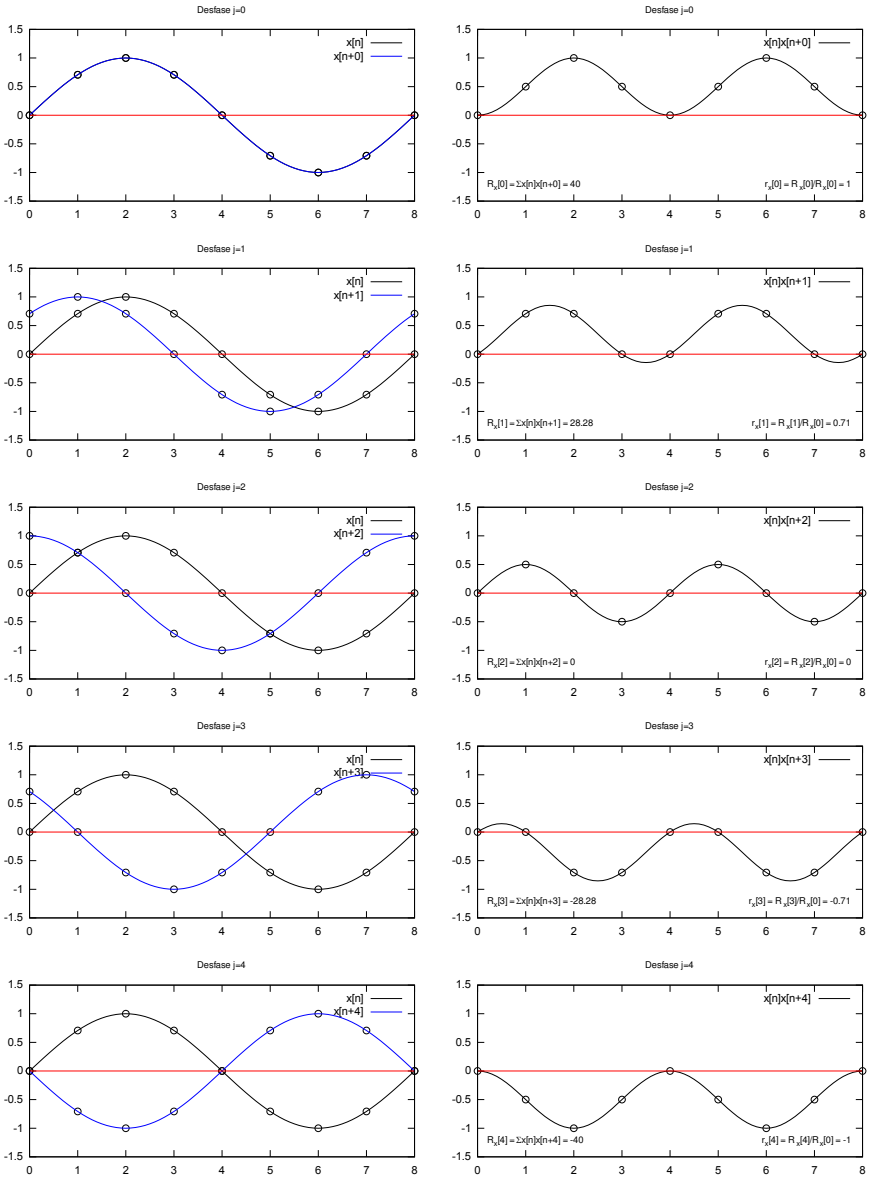


Figura 2.16: Cálculo de la autocorrelación discreta. La columna de la izquierda muestra las gráficas de la señal original x y la señal desfasada j muestras. La columna de la derecha muestra el producto punto a punto de ambas señales $x[n]x[n+j]$. La figura 2.17 muestra la autocorrelación normalizada $r_x[j]$ con base en los valores r_x de estas gráficas.

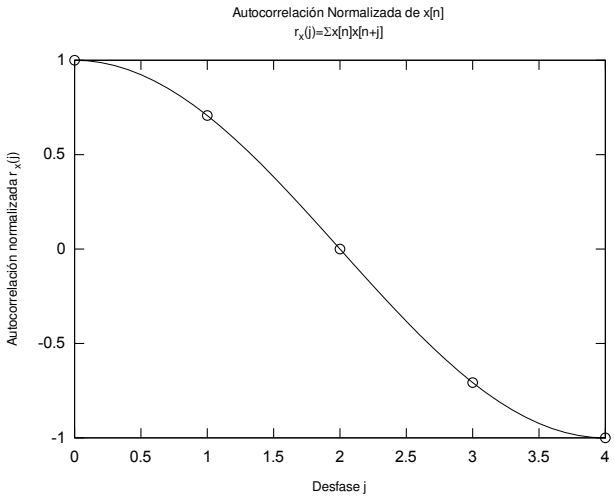


Figura 2.17: Correlación normalizada $r_x[j]$ en función del desfase j. Esta gráfica está basada en los datos de la figura 2.16

En la figura 2.18 se muestran seis ciclos de forma de onda de una señal periódica. Su correspondiente autocorrelación normalizada se muestra en la figura 2.19. Note que la autocorrelación también es periódica. Los máximos globales están marcados con círculos y todos tienen valor de 1. La distancia que hay entre cada máximo global es de 3 milisegundos, que corresponde al periodo $T_0 = 3.015\,\text{ms}$ de la forma de onda y por lo tanto la *frecuencia fundamental* es $F_0 = 1/T_0 = 331.6\,\text{Hz}$.

Si en la función de autocorrelación, no hay máximos globales mas que en $j = 0$, aún pueden existir máximos locales. Cuando un máximo local $r_x[j_{lmax}]$ tiene una altura suficientemente grande, se puede decir que existe una parte periódica en la señal cuyo periodo es igual a $T_0 = j_{lmax}$ y su valor $r_x[j_{lmax}]$ indica el grado de periodicidad[9]. Además, se puede calcular la relación periodicidad-ruido (HNR)[10] por medio de la siguiente expresión [16]:

$$\text{HNR (en dB)} = 10\log_{10}\frac{r_x[j_{lmax}]}{1 - r_x[j_{lmax}]} \tag{2.8}$$

Una señal de este tipo está ejemplificada en las figuras 2.20 y 2.21. La figura 2.20 representa una señal casi periódica con aproximadamente seis ciclos de

[9]En inglés *harmonic strength*.

[10]En inglés: Harmonic to Noise Ratio (HNR)

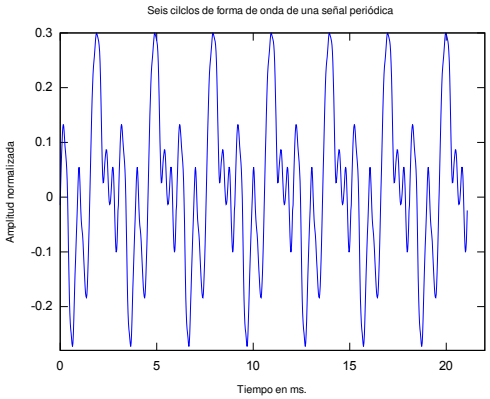


Figura 2.18: Seis ciclos de forma de onda de una señal periódica.

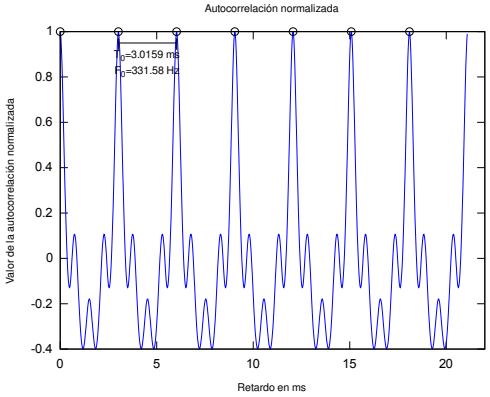


Figura 2.19: Autocorrelación de una señal periódica. Hay varios máximos locales marcados con un círculo, todos con un valor $r_x[nT_0] = 1$. El periodo de la forma de onda es $T_0 = 3\,\text{ms}$ y su *frecuencia fundamental* $F_0 = 331.6\,\text{Hz}$.

forma de onda. Este es un segmento de audio que fue extraído de un tono de guitarra cerca del ataque, donde todavía existe un poco de «ruido», asociado al sonido percusivo inicial (ataque). En la autocorrelación sólo hay un máximo global que está situado en $j_{gmax} = 0$ y se indica con un círculo $\circ$. Los máximos locales que aparecen en la figura 2.21 están por debajo de 1. El máximo local más alto de la autocorrelación está marcado con $\times$ y se encuentra en el desfase $j_{lmax} = 301$. Este pico es lo suficientemente pronunciado como para ser considerado como el periodo de la forma de onda, por lo que podemos concluir que este sonido en particular tiene una *frecuencia fundamental* aproximada de 146.51 Hz. El grado de periodicidad de este tono de guitarra está dado por el valor de la autocorrelación evaluado en el máximo local marcado con $\times$. El valor es $r_x[j_{lmax}] = 0.92$, que indica un alto grado de periodicidad. Esta información sugiere que la señal que se está analizando tiene un carácter tonal y sus componentes espectrales son prácticamente armónicas. El grado de periodicidad suele emplearse para determinar el tipo de excitación que se ha de usar en la síntesis de voz por medio de predicción lineal, que se explica en el capítulo 3.

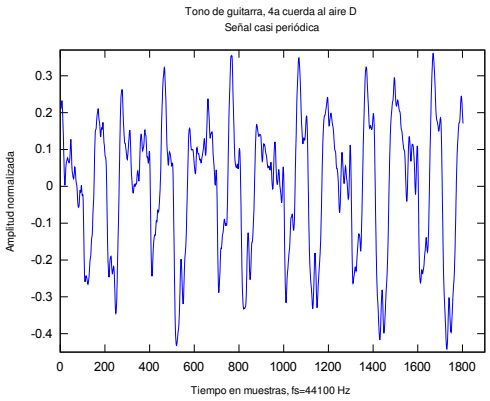


Figura 2.20: Señal casi periódica. Tono de guitarra, 4a cuerda al aire D(146 Hz). Muestreada a 44100 Hz.

Normalmente se escoge el máximo local más alto de la autocorrelación para determinar la *frecuencia fundamental* de una señal, pero este método puede fallar debido a que la altura de los máximos locales depende mucho de la forma de la ventana de análisis. Esto sucede especialmente con señales que tienen formantes muy marcadas. Boersma [16] propone una solución simple para eliminar los efectos de la ventana de análisis que consiste en dividir la

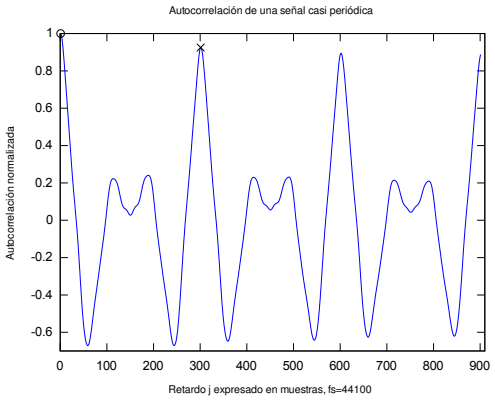


Figura 2.21: Autocorrelación de una señal casi periódica. El círculo $\circ$ indica el máximo global en $j_{gmax} = 0$. La cruz $\times$ marca el máximo local más alto de la autocorrelación normalizada en $j_{lmax} = 301$ que corresponde aproximadamente a una *frecuencia fundamental* de $F_0 = 146.51\,\mathrm{Hz}$

autocorrelación de la señal ventaneada[11] $r_{xw}(\tau)$ entre la autocorrelación de la ventana de análisis $r_w(\tau)$ para obtener la autocorrelación del segmento original $r_x(\tau)$.

$$r_x(\tau) \approx \frac{r_{xw}(\tau)}{r_w(\tau)} \tag{2.9}$$

La precisión con la que se puede calcular la *frecuencia fundamental* de una señal de audio muestreada, depende de la resolución de la FFT y la frecuencia de muestreo. Normalmente el análisis se realiza en segmentos pequeños de tiempo, por lo que la primera estimación $max(r_x[j])$ resulta muy burda. Boersma propone usar la interpolación sinc de la ecuación 2.10 para mejorar la resolución.

$$x(t) = \sum_{-\infty}^{+\infty} x_n \frac{\sin \pi(t - t_n)/\Delta t}{\pi(t - t_n)/\Delta t} \tag{2.10}$$

En nuestra implementación, primero se aplica una ventana Hanning a la autocorrelación discreta $r_x[j]$. La ventana debe estar centrada en la primera

[11] En inglés *Windowed signal*.

estimación del máximo local $r_x[j_{lmax}]$. Luego se aproxima el máximo local de la autocorrelación por medio de la ecuación 2.11.

$$r(\tau) \approx \sum_{n=0}^{N} r_{xH} \frac{\sin \pi(\tau - n)}{\pi(\tau - n)} \tag{2.11}$$

Aunque el método de Boersma reduce significativamente los efectos de la ventana, la autocorrelación aún puede fallar cuando se trata de obtener la *frecuencia fundamental* de una sonido de manera automatizada y a ciegas. Algunos instrumentos musicales, presentan mayor amplitud en sus armónicos superiores que en su primer armónico, por ejemplo la trompeta y el saxofón. En estos casos la autocorrelación puede fallar en detectar la *frecuencia fundamental*, escogiendo por ejemplo, algún múltiplo de la frecuencia fundamental. Afortunadamente, cuando se hace el análisis de un instrumento musical, normalmente se conoce de antemano la nota que fue ejecutada. Con esta información se puede estimar la *frecuencia fundamental* previamente al análisis. Así, se pueden especificar los límites de frecuencia dentro de los cuales el detector de periodicidad debe buscar. No vamos a ahondar más en este tema, pero puede consultar el artículo de Boersma [16] si necesita mayor información. A continuación se presenta el algoritmo de Boersma. El listado C.1, del apéndice C contiene el código fuente en Octave [8, 9]. La figura 2.22 contiene gráficas de los pasos del algoritmo de Boersma.

1. Tome un segmento de la señal lo suficientemente grande para contener tres ciclos de forma de onda si se quiere calcular la *frecuencia fundamental*. Utilice seis ciclos, si se quiere calcular el grado de periodicidad.

2. Reste al valor de cada muestra de la señal, el valor promedio del segmento de análisis. $x_i - \bar{x}$

3. Multiplique el segmento de señal por la ventana de análisis. Boersma recomienda la ventana Hanning o Gausiana. En este proyecto utilizaremos la ventana Hanning que está dada por:

   $$w(t) = \frac{1}{2} - \frac{1}{2} \cos \frac{2\pi t}{T} \tag{2.12}$$

   cuya autocorrelación normalizada se calcula como:

   $$r_w(\tau) = \left(1 - \frac{\mid \tau \mid}{T}\right) \left(\frac{2}{3} + \frac{1}{3} \cos \frac{2\pi\tau}{T}\right) + \frac{1}{2\pi} \sin \frac{2\pi \mid \tau \mid}{T} \tag{2.13}$$

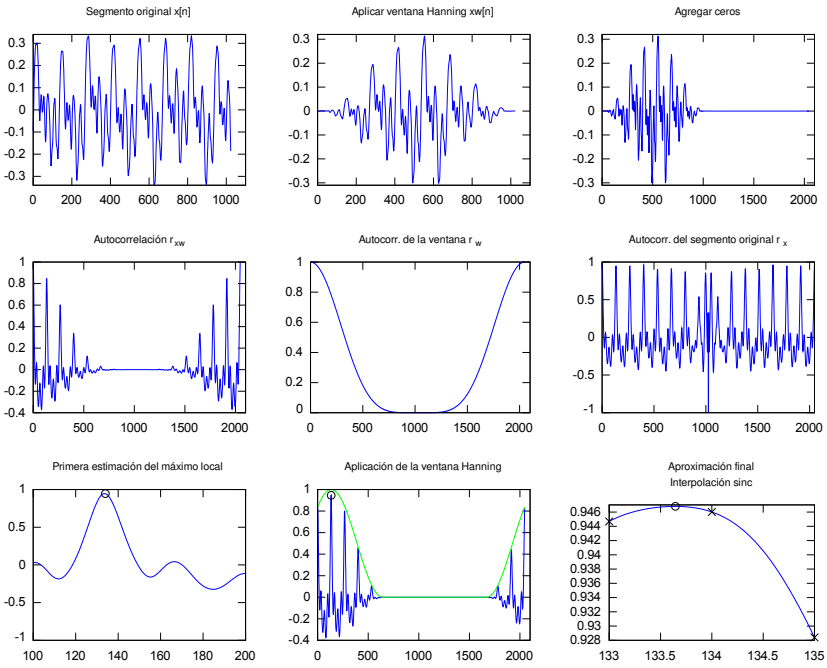


Figura 2.22: Gráficas del algoritmo de Boersma.

4. Agregue ceros al segmento de señal antes de calcular la autocorrelación. Se necesitan agregar por lo menos media ventana de ceros para realizar la interpolación sinc. Es preferible que el tamaño total sea una potencia de dos si se quiere calcular rápidamente la autocorrelación por medio de la Transformada Rápida de Fourier (FFT).

5. Aplique la Transformada Rápida de Fourier (FFT).

6. Calcule la magnitud al cuadrado de cada elemento en el dominio de la frecuencia (FFT^2).

7. Aplique la Transformada Inversa (IFFT). Con esto se obtiene la versión discreta de la autocorrelación de la señal ventaneada $R_{xw}[j]$

8. Obtenga la autocorrelación normalizada dividiendo entre el primer elemento de la autocorrelación $r_{xw}[j] = \frac{R_{xw}[j]}{R_{xw}[0]}$.

9. Divida entre la autocorrelación de la ventana de análisis $r_w[j]$ obtenida con la ecuación 2.13. Esto nos dará la autocorrelación del segmento original $r_x[j] = \frac{r_{xw}[j]}{r_w[j]}$

10. Encuentre el máximo local y refine el resultado por medio de la interpolación sinc y el algoritmo brent [20] que se encuentran en los listados C.2 y C.3 del apéndice C.

2.3.3. Localización de la parte sostenida y establecimiento del segmento de reproducción cíclica

Imaginemos a un tecladista en un concierto en vivo. Cuando presione una tecla, el sintetizador comenzará a producir el sonido de la nota indicada. El problema es que no se sabe cuanto durará la nota, hasta que el ejecutante decida soltar la tecla. Por esta razón, es necesario que un sintetizador tenga definido el algoritmo de síntesis, cuando se presiona o se suelta una tecla. En otras palabras, el sintetizador debe saber que hacer mientras se mantenga presionada una tecla. Para poder definir el comportamiento del sintetizador, será necesario analizar la señal original para localizar su parte sostenida y establecer los puntos de reproducción cíclica. En esta sección se explica con detalle como se implementa este tipo de análisis.

Los instrumentos musicales se pueden clasificar en dos grupos: impulsivos y continuos[12] [21]. Los instrumentos impulsivos son aquellos cuya excitación proviene de un impulso, mientras que los instrumentos continuos necesitan de una excitación continua. Por ejemplo, el tono de un violín que se toca con el arco es considerado *continuo*, debido a que la excitación proviene del frote continuo del arco con las cuerdas. Por otro lado, el sonido se considera impulsivo, si la cuerda del violín es pulsada con el dedo, al estilo *pizzicato*.

Suponga que se graba el tono de un instrumento para luego reproducirlo con un teclado electrónico como se haría con un *sampler*. Si el ejecutante quiere tocar esa nota con una duración mayor que la de la grabación, nos encontraremos con una dificultad, ya que no importa cuan larga sea la nota grabada, el ejecutante podrá siempre requerir una más larga. Además, grabar grandes segmentos de audio resulta poco práctico, pues se requiere mucha memoria para almacenar notas muy largas. Este problema es resuelto en los *samplers* grabando una porción pequeña de audio, que será reproducida de forma cíclica hasta que el ejecutante decida terminar la nota. Encontrar el segmento de audio que será reproducido cíclicamente durante la parte sostenida no es fácil y en muchos casos se obtienen de manera empírica y manual.

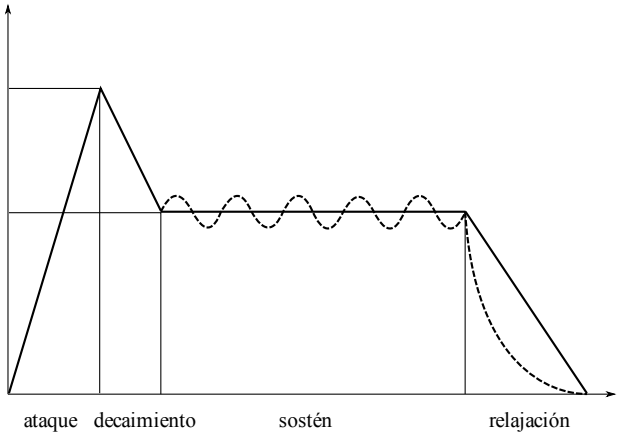


Figura 2.23: Envolvente temporal ADSR.

En varios sintetizadores se suele representar la envolvente temporal de un

[12]Clark [21] utiliza los términos *percusivos* y *no percusivos*. Peeters [22] se refiere a ellos como *no sostenidos* y *sostenidos*. En este trabajo usaremos los términos *impulsivos* y *continuos* tomados del artículo de John Hajda [23] y traducidos al español.

sonido por medio de cuatro segmentos: ataque, decaimiento, sostén y relajación [13]. La figura 2.23 muestra esta configuración. Sin embargo, este modelo no es adecuado para la *síntesis por segmentación de forma de onda*, ya que en esta, la envolvente temporal de amplitud es controlada por la posición de los nodos en cada ciclo de forma de onda. Además, el timbre de un sonido depende, no sólo de la envolvente temporal de amplitud, sino también de las características espectrales de la señal en distintos instantes de tiempo. Nos interesa entonces, partir la señal en sus diferentes segmentos, pero utilizando una representación más simple y tomando en cuenta las características espectrales de la señal.

Una representación más apropiada para este trabajo es la siguiente. Cuando el sonido sea impulsivo (fig. 2.24), se partirá la señal en dos segmentos: el ataque y la relajación. En el caso de un sonido continuo (fig. 2.25), se definirán tres segmentos: el ataque, el sostén y la relajación. Es importante aclarar que no se trata de caracterizar la envolvente temporal de amplitud solamente. Más bien, se trata de dividir la señal de audio en el dominio del tiempo, de acuerdo con distintos criterios, como puede ser su amplitud y la evolución del espectro en distintos instantes de tiempo; y tomando en cuenta el papel que cada segmento desempeñará dentro del algoritmo de síntesis. Será conveniente establecer una definición de cada segmento de audio, en el contexto de la fase de síntesis. Este autor propone las siguientes definiciones:

- *Ataque:* Es el segmento que empieza en el momento que el ejecutante decide comenzar la nota y termina cuando comienza la parte sostenida.

- *Sostén:* Es el segmento enseguida del ataque que será reproducido constantemente[14] hasta que el ejecutante decida terminar la nota.

- *Relajación:* Es el segmento que comienza cuando termina la parte sostenida y que depende del momento en que el ejecutante decide terminar la nota.

Los métodos que se conocen para caracterizar la envolvente temporal de una señal han surgido de estudios psicoacústicos. Muchos de estos estudios tienen como objetivo determinar la importancia relativa del ataque y la parte

[13] En inglés attack, decay, sustain, release (ADSR).

[14] Podría reproducirse de manera cíclica o de alguna otra forma que mantenga sonando la nota.

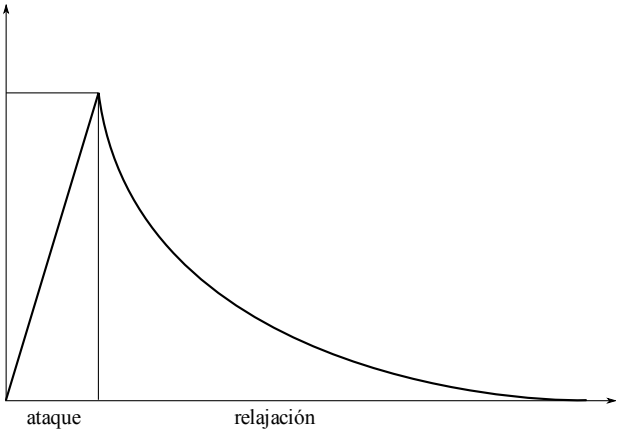


Figura 2.24: Representación de la envolvente de un sonido impulsivo.

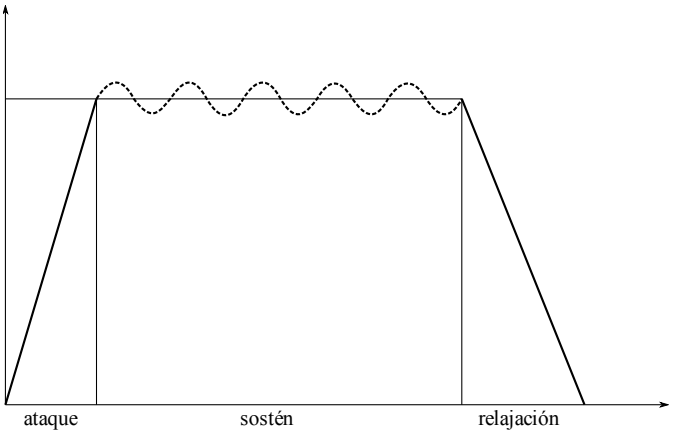


Figura 2.25: Representación de la envolvente de un sonido continuo.

sostenida de una señal, en el reconocimiento del timbre de un instrumento. Hasta la fecha, no se ha llegado a un acuerdo en cuanto a qué parte de la señal es más importante en el reconocimiento del timbre, pero podemos aprovechar estas técnicas de partición de una señal, en el ámbito de la síntesis sonora.

Existen varios métodos de segmentación de la envolvente temporal en la literatura [21–26], pero en este trabajo, discutiremos dos métodos derivados de los trabajos de John Hajda [23] y Geoffroy Peeters [22, 27].

John Hajda [23] propone un nuevo modelo de segmentación de la envolvente temporal llamado *trayectoria amplitud-centroide.*[15] Este modelo se basa en la relación que hay entre el valor cuadrático medio (RMS) de una señal y su centroide espectral.

Hajda define la amplitud RMS de la siguiente manera:

$$rms(t) = \sqrt{\frac{\sum_{k=1}^{n} a_k^2(t)}{n}}, \tag{2.14}$$

donde a es la amplitud lineal de la muestra k y n es el número de muestras de la ventana de tamaño fijo al tiempo t.

Y el centroide espectral esta dado por:

$$sc(t) = \frac{\sum_{b=1}^{m} f_b(t) a_b(t)}{\sum_{b=1}^{m} a_b(t)}, \tag{2.15}$$

donde f es la frecuencia y a es la amplitud de las bandas de frecuencia b hasta m que son obtenidas mediante una FFT.

En este modelo, el ataque se define como la parte inicial de la señal, durante la cual, aumenta la amplitud y disminuye el centroide espectral. El final del ataque lo determina el punto en que la pendiente del centroide espectral cambia de dirección, es decir en el mínimo local. La transición del ataque a la parte sostenida de la señal está definida como el segmento inmediato al ataque durante el cual, la amplitud continúa creciendo y el centroide espectral crece de forma general. La parte sostenida de la señal comienza cuando la amplitud ha llegado a un máximo local y tanto la amplitud como el centroide espectral varían de forma casi paralela. El decaimiento inicia cuando la amplitud y el centroide espectral decrecen. La tabla 2.1 ilustra de manera simple estas definiciones.

La figura 2.26 muestra un ejemplo de cómo se segmenta una señal de audio usando la trayectoria amplitud-centroide. La gráfica de abajo es la

[15] En inglés: *The Amplitude/Centroide trayectory.*

Tabla 2.1: Segmentación de la envolvente temporal basada en la *trayectoria amplitud-centroide*.

Segmento	Amplitud RMS	Centroide espectral
Ataque	+	−
Transición	+	+
Sostén	−/+	−/+
Decaimiento	−	−

representación en el dominio del tiempo de un tono de clarinete muestreado a 44100 Hz. La gráfica de arriba muestra el centroide espectral a lo largo del tiempo. Finalmente, en la gráfica de en medio se muestra el valor RMS. Las líneas verticales indican dónde se ha decidido segmentar la señal. El primer segmento es el ataque, el siguiente segmento es la transición a la parte sostenida. El segmento más largo es el sostén, y por último el decaimiento. Nótese que al final, el centroide espectral crece rápidamente. Este efecto se debe a que la amplitud es tan pequeña en el último segmento que se confunde con el ruido de fondo, produciendo un crecimiento en el centroide espectral.

El otro que nos interesa, es el *método del menor esfuerzo* que propone Peeters [22] para caracterizar la envolvente temporal de una señal de audio. En este método adaptativo, el principio y el final del ataque se estiman de acuerdo al comportamiento de la señal durante el ataque. Para un umbral determinado th_i se estima el tiempo t_i que tarda la envolvente de energía, en sobrepasar este umbral. Para valores de umbrales sucesivos th_i se define un esfuerzo w_i como el tiempo que toma pasar del valor de un umbral al de otro $w_i = t_{i+1} - t_i$. Luego se calcula el valor promedio del esfuerzo $\overline{w}$. Enseguida, se obtiene el mejor umbral para el inicio del ataque th_{st} buscando el primer umbral th_i para el que el esfuerzo w_i está por debajo de un valor $M * \overline{w}$. De manera similar, se obtiene el mejor umbral para el final del ataque th_{end} buscando el primer umbral th_i para el que el esfuerzo w_i está por encima de un valor $M * \overline{w}$. M es una constante cuyo valor típico es $M = 3$. Finalmente, los tiempos exactos de inicio t_{st} y de final t_{end}, se refinan tomando el mínimo y el máximo local respectivamente, alrededor de t_{st} y t_{end}.

En la figura 2.27 se muestra cómo se obtiene el inicio y final del ataque por medio del *método del menor esfuerzo*. En la gráfica de arriba aparecen los primeros 300 milisegundos para que se puedan ver los esfuerzos calculados que están marcados con líneas verdes y círculos. Las líneas rojas marcan el

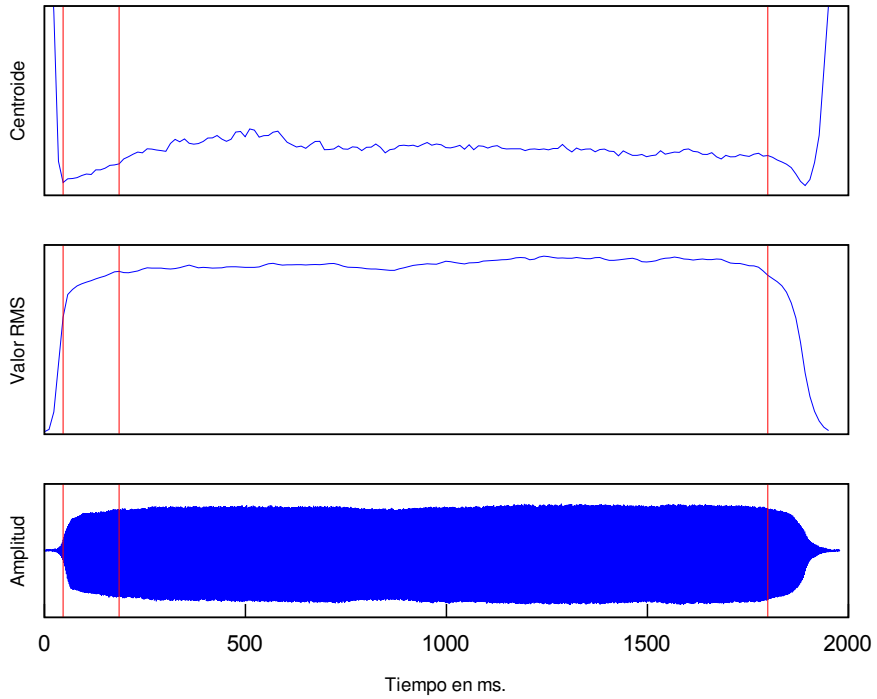


Figura 2.26: Segmentación de un tono de clarinete por medio de la trayectoria amplitud-centroide.

inicio y el final del ataque. La gráfica de abajo representa la señal original muestreada. Las líneas rojas, nuevamente, marcan el inicio y el final del ataque. Este es el mismo tono de clarinete que se usó en la figura 2.26 para que se puedan comparar ambos métodos de segmentación. Desgraciadamente Peeters [22] no explica la forma en que determina la parte sostenida. Además, su representación de la envolvente temporal contempla sólo dos segmentos: el ataque y el resto de la señal. Entonces podríamos tomar como inicio del sostén, el final del ataque.

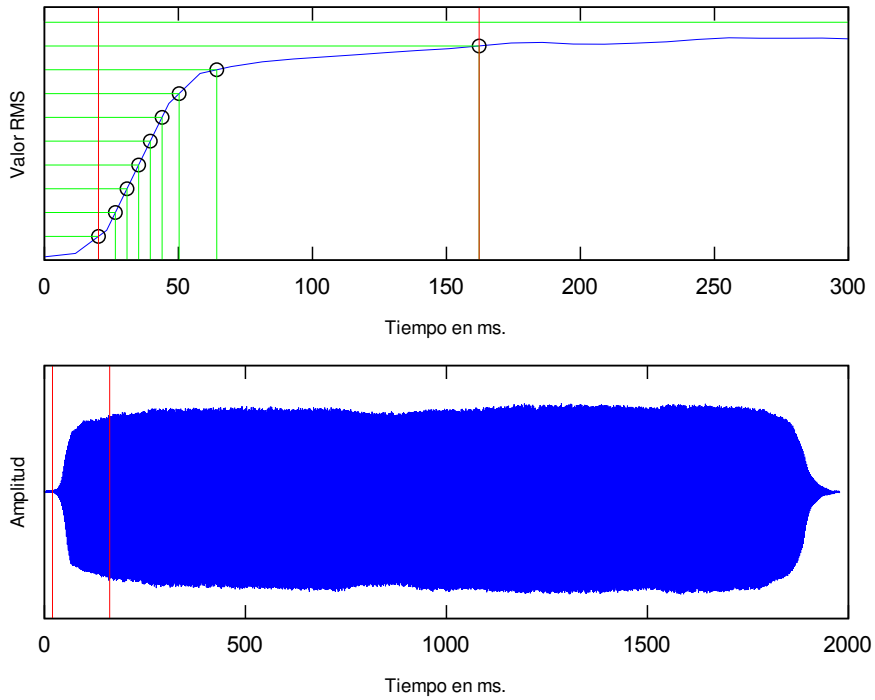


Figura 2.27: Segmentación de un tono de clarinete por medio del método del menor esfuerzo.

Además de la envolvente temporal de amplitud, se pueden usar otros criterios para segmentar una señal en el dominio del tiempo. Cada criterio podría ser más apropiado dependiendo del instrumento que se quiera sintetizar. Este autor, propone segmentar la señal de audio, utilizando como criterio la relación periodicidad-ruido (HNR) que se explicó en la sección 2.3.2 y que se obtiene mediante el algoritmo de Boersma [16]. Normalmente, durante el ataque, la relación periodicidad-ruido es menor que durante la parte soste-

nida, porque los diferentes resonadores de un instrumento musical tardan en estabilizarse, al igual que la periodicidad tarda en manifestarse. En palabras simples, el ataque es por lo general, la parte ruidosa de la señal. Esta es una de las razones por las que el centroide espectral decae durante el ataque, tal como lo sugiere Hajda [23] en su modelo de segmentación. Algunos autores han utilizado un criterio similar para definir el final del ataque. Por ejemplo Kendall [28], utiliza tres criterios para definir los límites del ataque, siendo uno de ellos el "claro establecimiento de la periodicidad de la señal." Podemos definir, entonces, el inicio de la parte sostenida t_{st} como el punto en el que la relación periodicidad-ruido HNR sobrepasa por primera vez un cierto umbral th. De igual manera, podemos definir el final t_{end} de la parte sostenida de una señal como el momento en que la periodicidad-ruido decrece y cruza por última vez un cierto umbral th. El umbral th se puede establecer como el valor promedio de la relación periodicidad-ruido (HNR).

$$th = \frac{1}{N} \sum_{k=0}^{N-1} \mathrm{HNR}[k] \tag{2.16}$$

Observe ahora la figura 2.28, en ella se muestra nuevamente un tono de clarinete, pero esta vez es segmentado con base en la relación periodicidad-ruido (HNR). La gráfica de arriba representa el valor HNR en función del tiempo. La línea horizontal verde indica el valor medio. Las líneas verticales rojas dividen la señal en tres segmentos: el ataque, el sostén y la relajación. La gráfica de abajo es la señal original segmentada.

Es importante tomar en cuenta que, no importa cual método de segmentación se utilice, el lograr resultados satisfactorios depende en gran parte de la calidad de la grabación y de la ejecución. Para sintetizar un sonido de alta fidelidad, es necesario una grabación de calidad y una buena ejecución. El ejecutante debe poder controlar eficientemente la dinámica, el vibrato, el trémolo u otros parámetros requeridos por el análisis para lograr buenos resultados.

Una vez que se ha determinado el sostén de la señal de manera aproximada, se buscará el segmento de reproducción cíclica. Una posibilidad sería tomar el sostén completo y reproducirlo cíclicamente. Esta solución sería viable si el tamaño de la grabación fuera una fracción de segundo. Observe la figura 2.28, la parte sostenida dura poco menos de 2 segundos. Es un segmento demasiado largo. Si el ejecutante decide terminar la nota cuando un ciclo del sostén va comenzando, tendrá que esperar poco más de un segundo

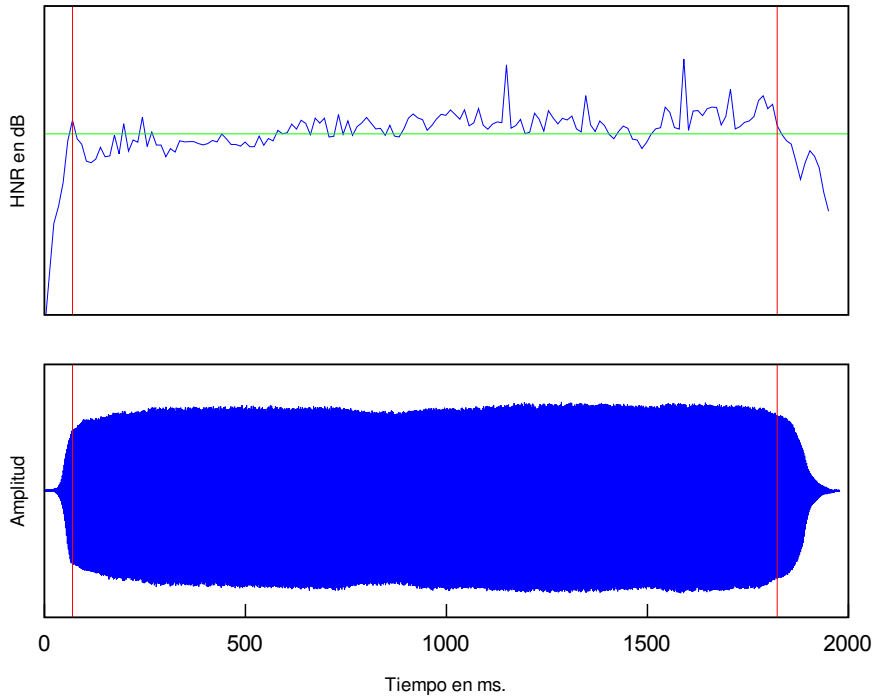


Figura 2.28: Segmentación de un tono de clarinete utilizando la relación periodicidad-ruido (HNR).

para que se apague la nota. El ejecutante desearía una respuesta inmediata a la acción de soltar la tecla. Mientras más pequeño sea el segmento de reproducción cíclica, más rápida será la respuesta. Sin embargo, cuando el segmento es demasiado pequeño[16] , el sonido será menos real, y podrá generar distorsiones o ruidos derivados de la falta de continuidad de un ciclo a otro. Esta discontinuidad puede resultar de la diferencia de la amplitud de un ciclo a otro, del trémolo, del vibrato o incluso de la inarmonicidad de las componentes del sonido. Otra solución es eliminar la relajación y terminar abruptamente la reproducción al momento en que el ejecutante suelte la tecla. En opinión de este autor, este comportamiento resulta adecuado para sonidos impulsivos, pero no para sonidos continuos que necesitan terminar de manera más natural. El tamaño del segmento de reproducción cíclica se puede calcular de manera empírica, dependiendo del sonido que se quiere sintetizar y de su frecuencia fundamental. Si se quiere calcular la duración del segmento de reproducción cíclica de manera automatizada, este autor recomienda una duración aproximada de 50 ms. La razón es que el rango de frecuencias audibles va de 20 Hz a 20 kHz. El límite inferior corresponde a una una frecuencia cuya longitud de onda de es de 50 ms. Si el segmento de reproducción cíclica es menor a 50 ms, podrían producirse frecuencias audibles no deseables causadas por la discontinuidad entre los extremos del segmento de reproducción cíclica. Si se desea utilizar segmentos de menor duración, será necesario escoger los extremos del segmento con mucho cuidado para garantizar la continuidad.

Si decidimos que el segmento de reproducción cíclica sea aproximadamente 50 milisegundos, entonces la parte sostenida tendrá que ser recortada, y tendremos que decidir qué parte del sostén es la más adecuada para ser reproducida cíclicamente. Tenemos tres posibilidades para ello:

1. Tomar 50 milisegundos a partir del inicio de la parte sostenida o final del ataque.

2. Tomar el punto en que la relación periodicidad-ruido (HNR) es mayor para garantizar una mayor continuidad entre ciclos.

3. Tomar los 50 milisegundos anteriores a la relajación.

En cualquier caso, tendremos que enfrentar la dificultad de pegar las diferentes partes de la señal sin que se note auditivamente el cambio de una a

[16]Menor a 50 milisegundos.

otra. En la figura 2.29 se ejemplifica el proceso para segmentar un tono de clarinete y determinar el segmento de reproducción cíclica. En la gráfica de arriba se muestra la señal original. Las líneas rojas delimitan el ataque, el sostén y la relajación. Las líneas verdes muestran el segmento que se ha escogido para la reproducción cíclica. Se escogió este segmento, porque es el que tiene la mayor relación periodicidad-ruido (HNR) como se puede verificar en la figura 2.28. En la gráfica de en medio se muestra la misma señal segmentada, pero se han eliminado las partes de la señal que serán sustituidas por el segmento de reproducción cíclica a la hora de la síntesis. Será necesario ahora, pegar los segmentos de manera que no se produzcan ruidos indeseables al pasar de uno a otro. Observe que las amplitudes de la señal en los diferentes segmentos es diferente y puede causar problemas a la hora de cortar y pegar la señal. Es necesario igualar estas amplitudes. El final del ataque y el inicio de la relajación se reajustan para que tengan una amplitud similar. El segmento de reproducción cíclica se escala para que el cambio de un segmento a otro sea suave y no produzca ruidos indeseables. Otra consideración importante, para evitar discontinuidades, es que, los ciclos de forma de onda deben estar en fase, en los puntos donde se pegarán las señales. La gráfica de abajo es el resultado final. Se han pegado los diferentes segmentos. Nótese que el archivo de audio es mucho más pequeño ahora. Durante la síntesis, el segmento delimitado por las líneas rojas será reproducido cíclicamente mientras el ejecutante mantenga presionada una tecla.

Este método funciona bien para sonidos armónicos y continuos. Aunque este método está pensado para editar sonidos que se usarán en un *sampler*, se puede aplicar a la *segmentación de forma de onda*, con la ventaja de que la continuidad entre ciclos lo determina el algoritmo de síntesis. Por esta razón la segmentación de la señal no necesitará ser tan exacta.

A continuación se presenta el algoritmo para la segmentación, edición y establecimiento del segmento de reproducción cíclica, de un sonido armónico y continuo:

1. Se segmenta la señal por medio de alguno o varios de los métodos explicados anteriormente. En el ejemplo del clarinete se utilizó la relación periodicidad-ruido.

2. Se extrae un ciclo de forma de onda del sostén de la señal y se calcula su periodo por medio del método de Boersma [16]. Se puede extraer, por ejemplo, el ciclo con el mayor valor HNR.

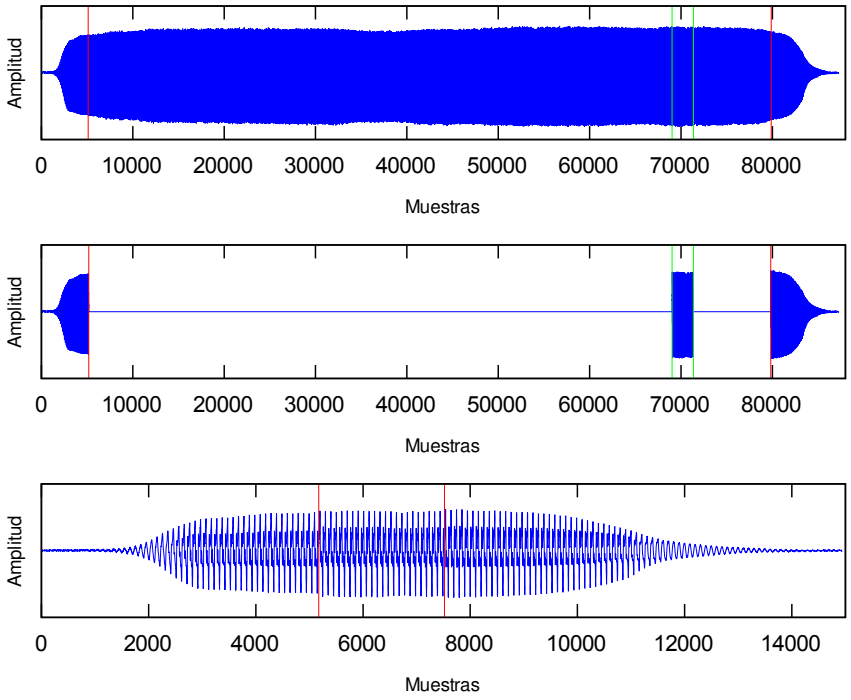


Figura 2.29: Edición de un tono de clarinete y determinación de su segmento de reproducción cíclica.

3. Se determina la amplitud del segmento de reproducción cíclica. En el ejemplo anterior se tomó la amplitud mayor entre el final del ataque y el inicio de la relajación.

4. Se redefinen los segmentos para que el final del ataque y el inicio de la relajación tengan la misma amplitud.

5. Se redefinen de nuevo los segmentos, de tal manera que el último ciclo del ataque esté en fase con el ciclo de forma de onda extraído en el punto 2. Esto se puede lograr haciendo una correlación cruzada entre el ciclo de forma de onda con el mayor valor HNR y la señal original. Se redefine el final del ataque para que coincida con un máximo local de la correlación. El primer ciclo de la relajación, también debe estar en fase con el ciclo de forma de onda con el mayor valor HNR. El inicio de la relajación también se redefine haciendo uso del máximo local de la correlación cruzada.

6. Se extraen aproximadamente 50 ms de la parte sostenida que contenga un número entero de ciclos alrededor del punto de mayor valor HNR. Este será el segmento de reproducción cíclica. El inicio de este segmento deberá estar en fase con el último ciclo del ataque. Se puede usar la correlación cruzada como se explica en el punto 5.

7. Se escala cada ciclo del segmento de reproducción cíclica para que tengan la amplitud que se determinó en el punto 3 y que coincide con el final del ataque y el principio de la relajación.

8. Se pegan las señales y se guarda en un archivo, por ejemplo, en formato WAV.

9. Se puede cargar este archivo en un *sampler* e indicar los puntos de inicio y final del segmento de reproducción cíclica.

2.3.4. Reducción de información

En la sección 2.3 mencionamos que la reducción de información se basa en dos principios: la redundancia de la señal de audio y su irrelevancia perceptual. En esta sección explicaremos algunas técnicas que se pueden emplear

en la segmentación de forma de onda para reducir información y de esta manera, ahorrar espacio de almacenamiento. La primera se basa en la selección del número de armónicos por medio de modelos perceptuales. Así podemos eliminar aquellos armónicos que no son percibidos por el oyente, es decir que son irrelevantes desde el puto de vista perceptual. La segunda técnica consiste en eliminar algunas formas de onda. Se escogen las más representativas y las formas de onda intermedias se interpolan. En este caso estamos utilizando el principio de la redundancia estadística.

Número de armónicos

En la sección 2.2 se explicó que se puede definir una forma de onda por medio de puntos y estos puntos se pueden obtener si se conocen las componentes espectrales del sonido que se quiere sintetizar. El número de puntos es aproximadamente el doble de los coeficientes de Fourier que se utilicen para el cálculo de los nodos. La pregunta que surge ahora es: ¿Cuál es el número óptimo de armónicos que se han de utilizar para la obtención de los nodos? Esta pregunta no es fácil de contestar. Si se utilizan todos los armónicos derivados del análisis, tendremos demasiados nodos que almacenar y manipular. Por otro lado, mientras menos armónicos se utilicen, el timbre se irá pareciendo cada vez menos al sonido original. Sin embargo, se pueden aprovechar las distorsiones que producen las funciones interpoladoras para aproximar los armónicos que han sido truncados. Podemos utilizar también las características de la audición humana para determinar el número de armónicos audibles y descartar los inaudibles. Esto sería similar a lo hacen algunos codificadores perceptuales como el MP3 [14, 29].

El rango de audición humana va de los 20 Hz a los 20000 Hz. Si tenemos un tono de 440 Hz, el máximo número de armónicos que podremos usar son $\frac{20000}{440} \approx 45$. Este número se podría reducir aún más, si tomamos únicamente aquellos armónicos que están por encima del umbral de audición como se muestra en la figura 2.30. En esta figura se presenta el espectro de un tono de clarinete. Los círculos negros marcan los armónicos. La línea roja es el umbral de audición. Como se puede observar, hay 25 armónicos que sobrepasan el nivel del umbral de audición. Se puede reducir aún más el número de armónicos si se considera también el efecto del enmascaramiento acústico, que consiste en que sonidos de mayor amplitud enmascaran, o vuelven inaudibles, a otros sonidos de menor amplitud. Para ilustrar este efecto, podemos pensar que estamos charlando con alguien, cuando de repente, se escucha un

avión pasar, ya no podemos escuchar la conversación y tenemos que esperar a que pase el ruido. Este es un ejemplo de enmascaramiento acústico, donde el ruido del avión enmascara nuestras voces. Si conocemos el espectro de un señal de audio, podemos determinar el umbral de enmascaramiento. Este umbral está determinado por aquellas componentes espectrales de mayor intensidad. En nuestro ejemplo, el umbral de enmascaramiento aparece con una línea verde, en la figura 2.31. Nótese que en este caso ssólo 18 componentes son audibles. Es importante tomar en cuenta la frecuencia máxima a la que se piensa transportar el tono, porque si se escogen muchos armónicos, el tono transportado puede producir *alias*. Esto sucede cuando la frecuencia de algún armónico sobrepasa la mitad de la frecuencia de muestreo $\frac{fs}{2}$. En este trabajo utilizamos el modelo perceptual que se utiliza en el estándar MPEG-1 [29–32] para obtener el umbral de enmascaramiento acústico de la señal de audio y así calcular el número óptimo de armónicos. Obviamente se puede escoger un número menor de armónicos si se quiere reducir aun más la información, pero habrá que considerar la perdida de la calidad sonora. Esto es permisible en algunos sistemas donde la calidad de audio no es tan importante. Este es el caso de algunos dispositivos móviles.

Interpolación de formas de onda

En un tono cuyas componentes son casi armónicas, las formas de onda en cada ciclo son muy parecidas. Para reducir información, nos interesa almacenar los datos de algunos ciclos solamente, no de todos ellos. Para ello, hay que analizar la señal y comparar unas formas de onda con otras para determinar cuales deberán almacenarse y cuales serán reconstruidas a partir de las formas de onda almacenadas. Existen varios algoritmos para determinar las formas de onda de *base*, estas han surgido principalmente de trabajos sobre síntesis aditiva y tablas de forma de onda[17] [33–41]. En este trabajo, se presenta un esquema simple de selección e interpolación de formas de onda. La implementación de algoritmos más complejos, como los algoritmos genéticos y perceptuales, queda pendiente para trabajos futuros.

Antes de continuar con el algoritmo de selección e interpolación de formas de onda, explicaremos dos conceptos importantes: la *diferencia apenas notable* (para la amplitud) y la *selectividad de frecuencias*.

Se llama *diferencia apenas notable*[18] a la diferencia que debe de haber

[17] En inglés *Wavetable Synthesis*.

[18] En ingles *Just Noticeable Difference JND*.

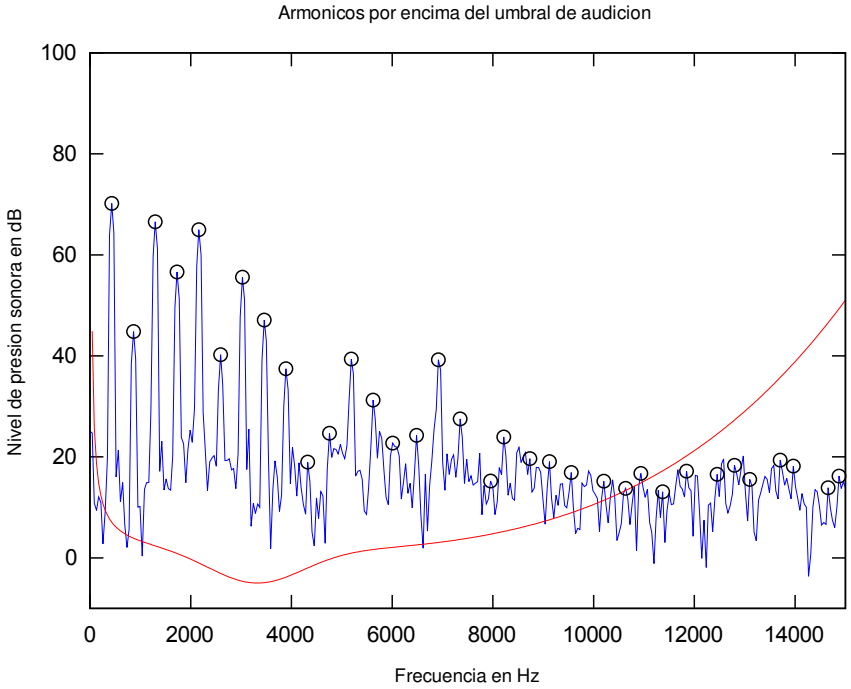


Figura 2.30: Cálculo del número de armónicos usando el umbral de audición.

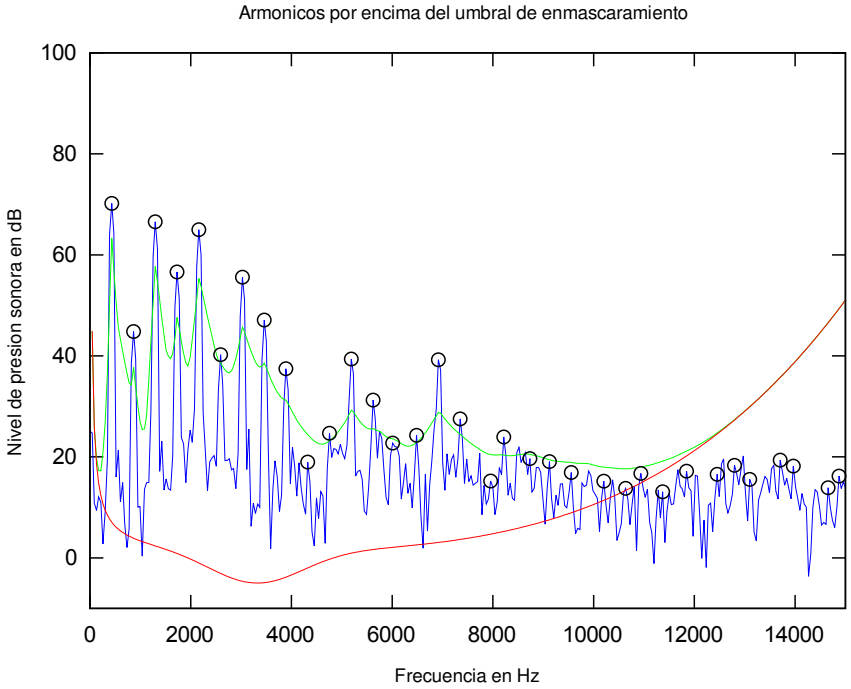


Figura 2.31: Cálculo del número de armónicos usando el umbral de enmascaramiento.

entre las magnitudes de dos de estímulos para ser percibidos como dos sensaciones diferentes. La *diferencia apenas notable* para la amplitud es de aproximadamente 1 dB [42]. En otras palabras, para notar un cambio de intensidad entre dos sonidos, debe de haber entre ellos una diferencia de amplitud de aproximadamente un decibelio. Si los sonidos tuvieran una diferencia de amplitud menor a un decibelio no notaríamos la diferencia.

La selectividad de frecuencias[19] se refiere a la habilidad de una persona para distinguir o *resolver*[20] las alturas de dos o más tonos puros superpuestos. Para estudiar la selectividad de frecuencias se mide la mínima separación en frecuencia que debe de haber entre dos *tonos puros* para que sean escuchados separadamente. Esta separación corresponde aproximadamente al ancho de banda crítica y puede ser calculada mediante la siguiente expresión:

$$ERB = 24.7(4.37F_c + 1) \tag{2.17}$$

donde F_c es la frecuencia central en kilohertz. En el caso de un *sonido complejo*, la distancia entre las componentes adyacentes es 1.25 veces más grande ($1.25ERB$) que la que se necesita para resolver dos *tonos puros* [43]. En un tono de corno de 220 Hz, por ejemplo, hay una diferencia de 220 Hz entre un armónico y el armónico siguiente. El noveno armónico tiene una frecuencia de $nf_0 = 220 \times 9 = 1980$ Hz. Para que el armónico 9 y 10 sean resueltos tiene que haber una diferencia de 268 Hz, que es mayor a la separación que hay entre ellos (220 Hz). Quiere decir que los armónicos que van más allá del noveno no son resolubles[21]. El número de armónicos resolubles (nr) para un tono con una frecuencia fundamental f_0 está dada por la siguiente expresión:

$$nr \approx \frac{800000\, f_0 + 24700000}{107939\, f_0} \tag{2.18}$$

En este trabajo, el criterio de selección para escoger las formas de onda de base esta basado en estas dos características de la percepción. El algoritmo consiste en comparar los espectros de dos segmentos de la señal original en

[19] En inglés *frequency discrimination.*

[20] Del inglés *resolve*, *resolvable harmonics*, traducido al español como *resolver*, *armónicos resolubles*. Resolver se refiere a la capacidad que tiene una persona para distinguir separadamente las parciales de un tono complejo.

[21] Aunque los armónicos no sean resolubles, sí se escuchan, pero no como dos sonidos diferentes. Los oímos como un sólo sonido con batimentos o con un timbre zumbante. Cuando existen varios tonos dentro de la misma banda crítica, estos se perciben como ruido coloreado.

diferentes instantes de tiempo. Solamente se toman en cuenta los armónicos que son resolubles. Se comparan, una a una, las amplitudes de los armónicos resolubles, si la diferencia de amplitud de alguno de los armónicos es mayor a 1 dB, entonces se establecen los segmentos (ciclos) correspondientes como formas de onda de base. Si la diferencia de amplitud de *todos* los armónicos resolubles es menor a 1 dB, entonces se escogen un segmento mas alejado. De acuerdo con este criterio, dos formas de onda son consideradas auditivamente indistinguibles si se cumple

$$MAX\left(\frac{|a_{(t_0,n)} - a_{(t_1,n)}|}{a_{(t_0,n)}}\right) < 0.25 \tag{2.19}$$

donde $a_{(t_0,n)}$ representa la amplitud del armónico n al tiempo t_0 y n puede tomar los valores $n = \{1, 2, 3, \ldots, nr\}$.

Tomemos como ejemplo un tono de saxofón con una frecuencia fundamental de 220 Hz. El tono tiene una duración aproximada de 3 segundos. En la figura 2.32 se muestran los segmentos de la señal de audio que fueron seleccionados como formas de onda de base. Note que la señal original contiene 649 ciclos en total. Se encontraron 66 puntos, esto representa un ahorro considerable de datos. Observe que al inicio y al final de la señal casi no se logra reducir información. Esto se debe a que en el ataque y la relajación, el espectro de la señal de audio es muy cambiante.

Una vez que se tienen las formas de onda de base, es necesario establecer un procedimiento para interpolar las formas de onda intermedias. La manera más sencilla es interpolar los nodos de la forma de onda de manera lineal. Esto es equivalente a hacer un crossfade entre las dos formas de onda de base. Haciendo un crossfade se reducen mucho el número de cálculos necesarios para interpolar cada punto linealmente. Una forma alternativa, es hacer un crossfade pero estableciendo una constante k que minimice el error cuadrático e_n^2 para cada ciclo. Así se tendría que almacenar un valor k para cada ciclo de forma de onda, en el caso anterior representaría 649 valores.

Sean x_n y y_n dos formas de onda de base, localizadas en tiempos distintos t_x y t_y de la señal. Queremos aproximar una forma de onda intermedia z_n que fue previamente eliminada en el proceso de análisis. La forma de onda z_n se localiza en un tiempo t_z que está entre los dos puntos de base $t_x < t_z < t_y$. Para aproximar el ciclo intermedio z_n, se puede hacer una suma de los ciclos de base, utilizando una constante k de ponderación. La constante k puede tomar cualquier valor, pero se espera, que si la forma de onda intermedia z_n es muy parecida a las otras dos, el valor de k tomará valores entre 0 y 1.

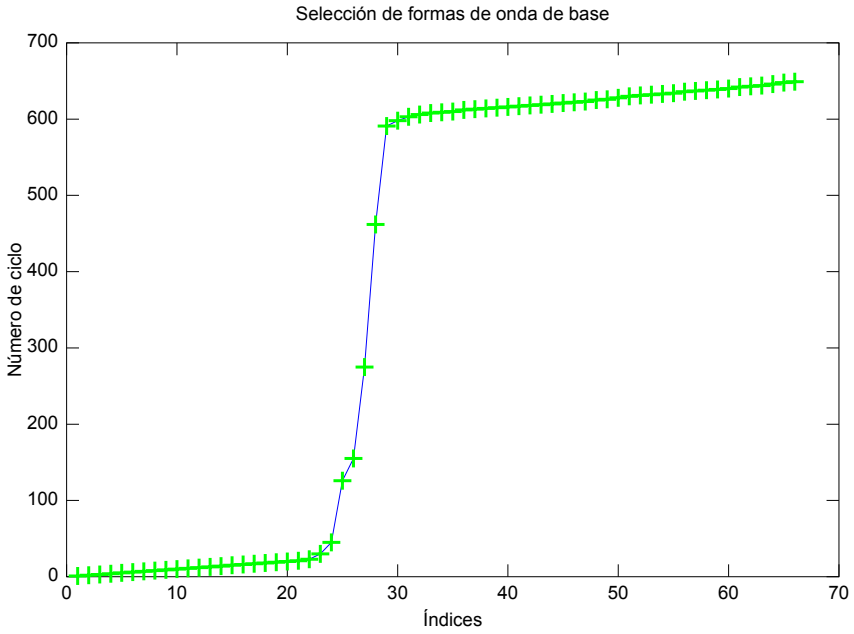


Figura 2.32: Selección de formas de onda de base. Las cruces indican las formas de onda que fueron seleccionadas para ser almacenadas. las demás serán reconstruidas por medio de interpolación.

Si la forma de onda z_n se parece más a x_n, el valor de k se acercará más a 1. Por el contrario, si la forma de onda z_n se parece más a y_n, el valor de k estará más cerca de 0. La aproximación de z_n se expresa como sigue:

$$kx_n + (1-k)y_n + e_n = z_n \tag{2.20}$$

donde la variable e_n representa el error.

$$e_n = z_n + (k-1)\ y_n - k\,x_n \tag{2.21}$$

$$S_r = \sum_{n=0}^{N-1} e_n^2 = \sum_{n=0}^{N-1} \left(z_n + (k-1)\ y_n - k\,x_n\right)^2 \tag{2.22}$$

$$\frac{dS_r}{dk} = 2\sum_{n=0}^{N-1} \left(y_n - x_n\right)\left(z_n + (k-1)\,y_n - kx_n\right) = 0 \tag{2.23}$$

Minimizando la suma del error cuadrático e_n^2 se obtiene la siguiente ecuación para encontrar la constante de ponderación.

$$k = \frac{\sum_{n=0}^{N-1} \left(y_n - x_n\right)\ \left(y_n - z_n\right)}{\sum_{n=0}^{N-1} \left(y_n - x_n\right)^2} \tag{2.24}$$

Para ilustrar este proceso, observe la figura 2.33. En esta figura se presentan cuatro ciclos de forma de onda de un tono de saxofón. La gráfica de arriba muestra la señal original. Los extremos de la señal, pintados en rojo y verde, representan las formas de onda de base. Los ciclos intermedios en color azul no son almacenados. Al momento de la síntesis se interpolan haciendo una suma ponderada de los ciclos de base. El resultado se muestra en la gráfica de abajo, en color negro.

2.3.5. Formato de almacenamiento

Una vez que se ha realizado el análisis, se almacenarán los parámetros en un archivo con una estructura RIFF[22] o IFF[23]. El formato IFF es un formato genérico desarrollado por Electroninc Arts con la colaboración de Commodore-Amiga. Este formato fue creado para intercambiar archivos entre computadoras con diferente arquitectura. El formato RIFF es un formato

[22]Del inglés *Resource Interchange File Format*
[23]Del inglés *Interchange File Format*

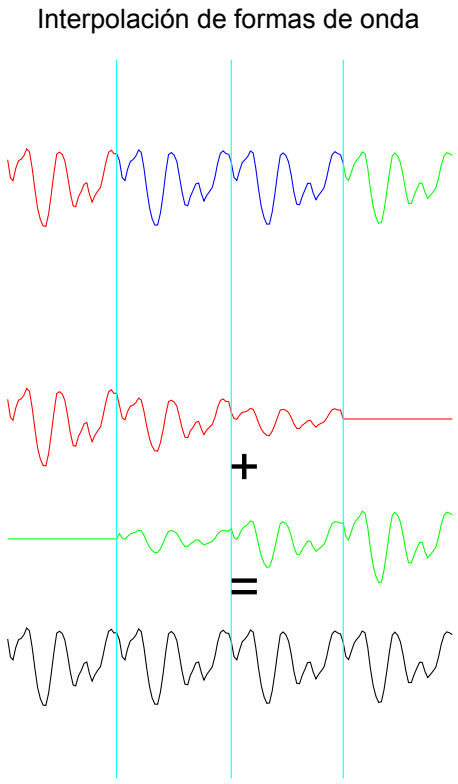


Figura 2.33: Interpolación de formas de onda. La figura de arriba muestra cuatro ciclos de la señal original. Las formas de onda en color verde y azul son las formas de onda de base. Los ciclos pintados de azul, son los que se van a aproximar por medio de interpolación utilizando la constante k. La suma ponderada de las formas de onda produce la señal aproximada que está en la gráfica de abajo, en color negro.

basado en IFF que fue introducido por IBM y Microsoft para el almacenamiento de contenido multimedia. Los archivos de tipo WAVE y AIFF tienen un formato de este tipo. Los formatos RIFF y IFF son esencialmente iguales, solamente difieren en el orden de bytes que se usa para el almacenamiento de datos. Nuestra implementación puede leer y escribir ambos formatos indistintamente.

Para almacenar los parámetros de la *síntesis por segmentación de forma de onda* se diseñó una estructura de tipo IFF con un formato tipo MITS que se explica en la tabla 2.2. Note que el identificador de formato puede ser 'RIFF' o 'FORM' dependiendo del orden de bytes que se esté usando. Cuando se use un ordenamiento *Big-Endian*[24] se utilizará el identificador 'FORM'. Cuando se utilice un ordenamiento *Litle-Endian*, se utilizará el identificador 'RIFF'. El formato MITS consiste en 5 bloques: *mand, loop, modu, time y nods*. Los bloques obligatorios son: *mand, time y nods* y los opcionales son *loop, modu y time*. A continuación se explica cada bloque:

mand Contiene información básica sobre la síntesis por segmentación de onda: número de nodos, número de formas de onda, frecuencia fundamental, amplitud y función interpoladora. Este bloque es obligatorio. Las funciones interpoladoras posibles son: constante, lineal, medio-coseno y cúbica. Existe un parámetro disponible que se piensa implementar en un futuro agregando un sexto bloque al formato de archivo MITS. El parámetro OT indica que se utilizará una función interpoladora establecida por el usuario. La función interpoladora estará descrita en forma de texto en un bloque especial *intf*. Por ahora, está característica no está implementada.

loop Contiene información sobre el segmento de reproducción cíclica. Este bloque es opcional. La ausencia de este bloque le indica al reproductor que el sonido es impulsivo y carece de segmento de reproducción cíclica.

modu En este bloque se especifican los parámetros para la modulación de amplitud y frecuencia. En otras palabras, se establecen los parámetros para el vibrato y/o el trémolo durante la parte sostenida. Este bloque es opcional, si se omite, el reproductor no aplicará ningún tipo de modulación.

[24]En un ordenamiento *Big-Endian* el byte mas significativo (MSB) se almacena primero mientras que en un ordenamiento *Little-Endian* el byte menos significativo (LSB) se almacena primero.

time Este bloque tiene información sobre el tiempo en que se debe de cambiar cada ciclo de forma de onda. Este bloque es obligatorio.

nods Este bloque contiene los valores de los nodos para cada ciclo de forma de onda. Este bloque es también obligatorio.

En un futuro se pueden agregar más bloques de datos. La idea es que los reproductores que utilicen este tipo de archivo puedan ignorar la información de aquellos bloques para los que no se cuenta con una implementación.

2.4. Síntesis

En esta sección describimos en detalle el algoritmo de síntesis. La fase de síntesis es más sencilla que la fase de análisis, esto resulta ventajoso para utilizar esta técnica de síntesis en tiempo real. Partiremos de que el análisis se ha realizado con éxito y se cuenta, por lo menos, con la siguiente información:

- La función interpoladora
- El valor de los nodos para cada forma de onda
- Tiempos de inicio de cada forma de onda
- Los puntos de reproducción cíclica (sólo para sonidos continuos)

Otra información que puede ser de utilidad y que será implementada más adelante es la siguiente:

- Parámetros de la modulación de amplitud (trémolo)
- Parámetros de la modulación de frecuencia (vibrato)
- Variaciones de frecuencia (frecuencia instantánea)

Como veremos más adelante, esta información es guardada en una estructura MitsuData, como se muestra en el listado C.4, para poder utilizarla en el algoritmo de síntesis.

Tipo	Bytes	Contenido
char[4]	4	Identificador "FORM" o "RIFF" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "RIFF"
char[4]	4	Identificador "MITS" en formato ASCII
char[4]	4	Identificador "mand" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "mand"
unsigned int	4	Número de nodos
unsigned int	4	Número de formas de onda
unsigned int	4	Frecuencia de muestreo en Hz (señal original)
float	4	Periodo de un ciclo de forma de onda en muestras (señal original)
unsigned short	2	Resolución de la amplitud en bits (16 bits por omisión)
unsigned short	2	Función interpoladora: "PC", "PL", "PH", "CU", "OT"
char[4]	4	Identificador "loop" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "loop"
unsigned int	4	Inicio del segmento de reproducción cíclica (loopA en muestras)
unsigned int	4	Final del segmento de reproducción cíclica (loopB en muestras)
char[4]	4	Identificador "modu" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "modu"
float 32	4	Frecuencia del vibrato en Hz
float 32	4	Amplitud del vibrato, -1.0 a +1.0
float 32	4	Frecuencia del trémolo en Hz
float 32	4	Amplitud del trémolo, -1.0 a +1.0
char[4]	4	Identificador "time" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "time"
unsigned int	4	Tiempo de inicio del ciclo de forma de onda 0 en muestras
unsigned int	4	Tiempo de inicio del ciclo de forma de onda 1 en muestras
unsigned int	4	Tiempo de inicio del ciclo de forma de onda 2 en muestras
unsigned int	4	Tiempo de inicio del ciclo de forma de onda 3 en muestras
...	...	...
char[4]	4	Identificador de encabezado "nods" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "nods"
signed short	2*	Ordenada del nodo 0, cilco 0
signed short	2*	Ordenada del nodo 1, cilco 0
signed short	2*	Ordenada del nodo 2, cilco 0
signed short	2*	Ordenada del nodo 3, cilco 0
...	...	...
signed short	2*	Ordenada del nodo 0, cilco 1
signed short	2*	Ordenada del nodo 1, cilco 1
signed short	2*	Ordenada del nodo 2, cilco 1
signed short	2*	Ordenada del nodo 3, cilco 1
...	...	...

Tabla 2.2: Estructura del archivo de almacenamiento MITS

2.4.1. Respuesta a mensajes MIDI

En la especificación del protocolo MIDI están definidos muchos tipos de mensajes [44]. En este trabajo, sólo nos ocuparemos de los eventos *note-on* y *note-off*. Cuando se presiona una tecla de un teclado o controlador MIDI, se genera un evento llamado *note-on*. Este mensaje es recibido por un sintetizador o generador de sonidos. El sintetizador comenzará a generar el sonido y continuará generándolo hasta que reciba otro mensaje que le indique que debe terminar. Este mensaje de terminación se llama *note-off* y es enviado al sintetizador en cuanto la tecla es soltada.

Sonidos impulsivos

El comportamiento del sintetizador cuando se presiona o se suelta una tecla, depende del tipo de sonido y de los parámetros que se han obtenido del análisis. El caso más simple, es la síntesis de sonidos impulsivos (sección 2.3.3).

Los sonidos impulsivos no necesitan un segmento de reproducción cíclica, por lo que los parámetros de inicio y final de la parte sostenida no están especificados. En este caso, cuando el sintetizador reciba un mensaje *note-on*, reproducirá una por una las formas de onda, desde el principio hasta el final y sin interrupción, a menos que se reciba un mensaje *note-off*, en cuyo caso la reproducción del sonido terminará de manera abrupta.

La figura 2.34 muestra un tono de guitarra que se está reproduciendo en respuesta a un mensaje *note-on*. Cuando el ejecutante presiona la tecla, el tono de guitarra comienza a reproducirse. En el momento que el ejecutante suelta la tecla, se genera un mensaje *note-off* y el sonido termina abruptamente. En ocasiones, se requiere un comportamiento distinto. Por ejemplo, cuando se suelta una tecla de un piano acústico, el fieltro del apagador hace que las cuerdas dejen de sonar, pero esto no sucede de manera inmediata, la extinción del sonido está amortiguada, es decir que tarda un poco en apagarse por completo. El efecto es el de una nota que se apaga suavemente. Se puede lograr este efecto aplicando una envolvente temporal que baje rápidamente a cero al momento que se recibe el mensaje *note-off*. La figura 2.35 muestra esta opción. Un caso ligeramente diferente es el de algunos instrumentos de percusión que normalmente se dejan sonar hasta su extinción. En este caso, el algoritmo no respondería a mensajes *note-off*, simplemente se apagaría el sonido al terminar la reproducción completa del sonido en cuestión. Un ejem-

plo de esto sería una campana que dejaríamos sonar hasta que su sonido se apague por completo.

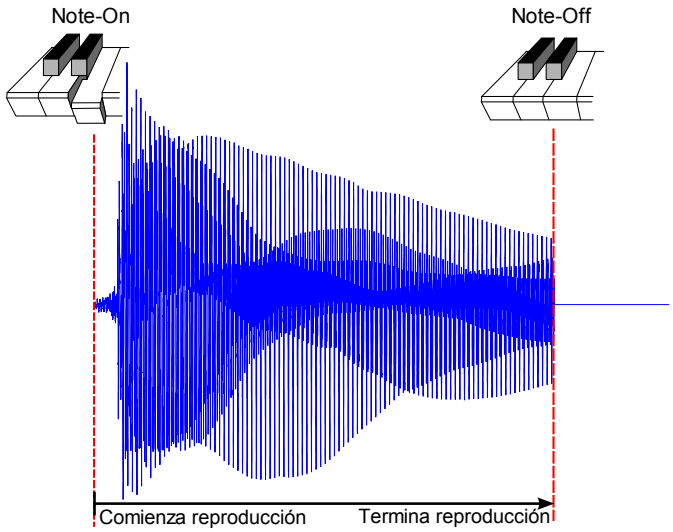


Figura 2.34: Respuesta a mensajes MIDI para sonidos impulsivos.

Sonidos continuos

Los sonidos continuos requieren un tratamiento diferente. Para ahorrar espacio de almacenamiento, se guarda una pequeña porción del sostén de la señal de audio, que será reproducida cíclicamente (sección 2.3.3). En este caso, cuando se recibe un mensaje *note-on*, el algoritmo de síntesis, reproducirá las formas de onda del ataque de forma ininterrumpida. Enseguida comenzará la reproducción cíclica entre dos puntos a los que llamaremos *loopA* y *loopB*. Cuando se llegue al punto *loopB* la reproducción comenzará de nuevo en el punto *loopA* y así sucesivamente hasta que el sintetizador reciba un mensaje *note-off*. En nuestra implementación la reproducción cíclica es un poco más complicada, pero se explicará con detalle en la sección 2.4.3. Por el momento discutiremos la versión simplificada. En el momento que el sintetizador recibe un *note-off*, se puede escoger alguno de los siguientes comportamientos:

- El sonido terminará de manera abrupta (figura 2.36a).
- El sonido terminará de forma amortiguada (figura 2.36b).

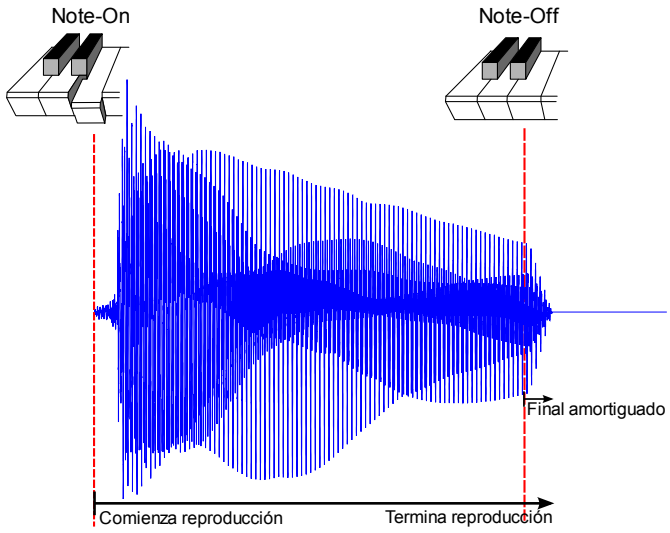


Figura 2.35: Respuesta a mensajes MIDI para sonidos impulsivos con final amortiguado.

- La reproducción continuará hasta el punto *loopB* y luego comenzará la parte final de la señal (figura 2.36c).
- La reproducción continuará hasta que termine el ciclo de forma de onda en curso y luego saltará directamente al punto *loopB* donde comenzará la parte final o relajación del sonido (figura 2.36d).

En la figura 2.36 aparecen los cuatro casos listados anteriormente. Las rayas punteadas rojas representan el momento en que el ejecutante presiona y suelta la tecla respectivamente. Las rayas negras punteadas delimitan el segmento de reproducción cíclica. Note que la tecla se está soltando a la mitad de la reproducción cíclica. Con la figura 2.36a se ilustra que el sonido parará abruptamente al recibirse un mensaje *note-off*. En la figura 2.36b se muestra el caso en que el sonido es amortiguado al momento de recibir el *note-off*. En el ejemplo de la figura 2.36c, la reproducción de la parte cíclica continúa después de recibir el *note-off*, pero al llegar al punto *loopB* la reproducción continúa con la relajación o parte final del sonido. En la última figura, cuando el ejecutante suelta la tecla, el sintetizador no termina de reproducir la parte sostenida, en cambio da un salto directo al punto *loopB* para terminar con la reproducción de la relajación.

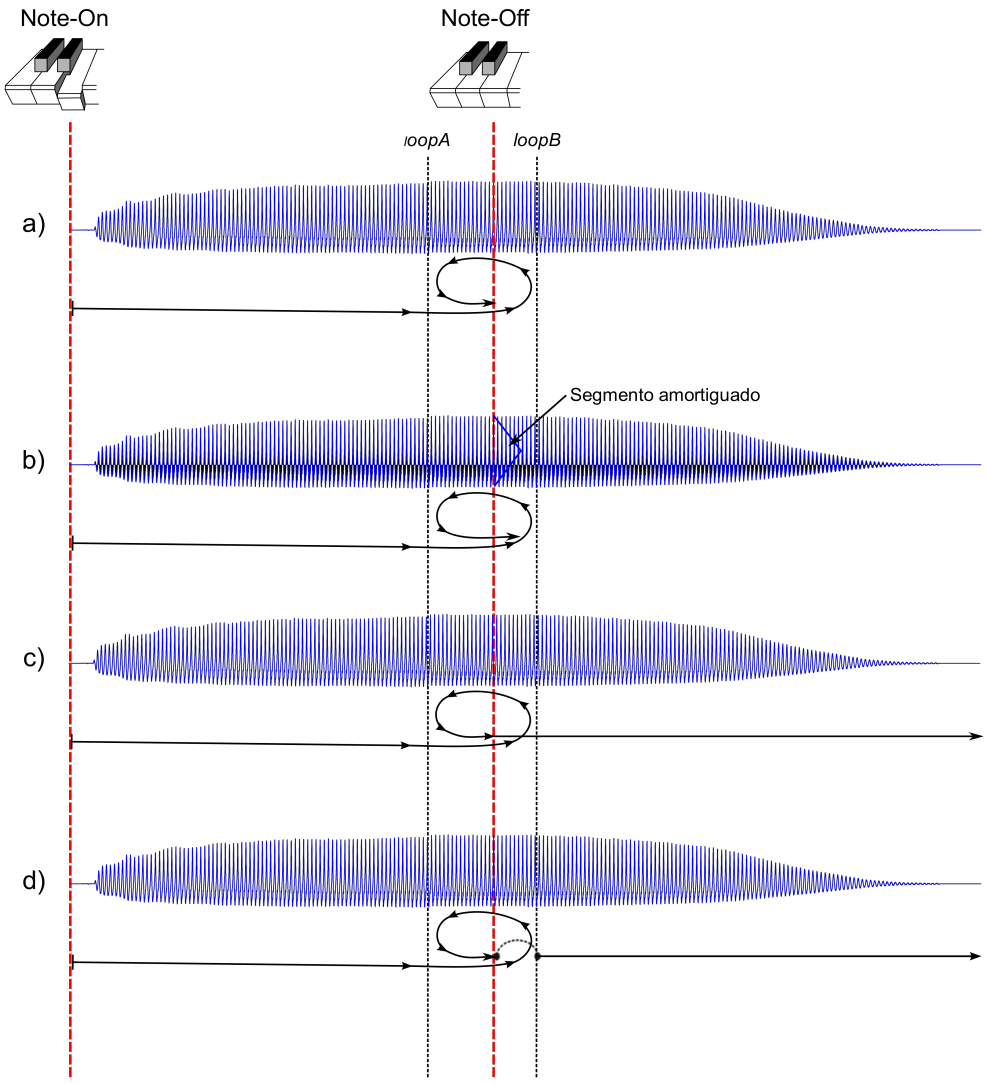


Figura 2.36: Respuesta a mensajes MIDI para sonidos continuos.

2.4.2. Interpolación de formas de onda

En la sección 2.2 se explicó la teoría de la segmentación de forma de onda según Mitsuhashi. Aquí vamos a explicar en la práctica como se logra esta técnica de síntesis. En el modelo de Mitsuhashi se tiene un número N de nodos igualmente espaciados. Los nodos están unidos por una función interpoladora $f(x)$ normalizada a $+1$ en ambos ejes. Se puede encontrar el valor de cualquier punto de la forma de onda aplicando la ecuación 2.1. Si se utiliza esta forma de onda para sintetizar un sonido, pero nunca cambiamos las ordenadas de los nodos, entonces tendremos un sonido estático y con poco interés. La idea es cambiar los valores de los nodos en cada ciclo que pasa. En otras palabras, cada que se termina un ciclo, es necesario actualizar los valores de los nodos para generar una nueva forma de onda. Así podremos obtener un sonido más realista, cuyo espectro evoluciona a través del tiempo.

Podemos calcular los valores de los nodos para cada ciclo de forma de onda, mediante las técnicas de análisis explicadas en la sección 2.3. Por ejemplo, supongamos que se graba un segundo de un tono de corno de 440 Hz con una frecuencia de muestreo de 44100 Hz. El periodo de cada forma de onda será de aproximadamente $T = 100$ muestras. Esto quiere decir que obtendremos aproximadamente 440 ciclos en total. Si cada ciclo está representado por 20 nodos, necesitaremos un total de 8800 valores para representar un segundo del tono de corno. Resultan ser demasiados valores, por ello es aconsejable hacer una reducción de información. Para ello, se guardan solamente los valores de algunos ciclos. En la sección 2.3.4 se explica con detalle este procedimiento. Supongamos que, de los 440 ciclos, sólo guardamos la información de 20 de ellos. ¿Cómo reconstruiremos los valores de los demás ciclos? Tenemos que interpolar los valores entre los nodos. En nuestra implementación se interpolan los valores linealmente. Esto puede causar algunas distorsiones que disminuirán con el uso de los filtros obtenidos con predicción lineal, que es el tema del capítulo 3. De esta manera, se puede lograr un gran ahorro de memoria.

En la figura 2.37 se muestra un ejemplo. Se analiza un tono de guitarra para calcular los nodos de cada forma de onda. La gráfica de arriba muestra todos los ciclos de forma de onda, aproximados por el modelo de Mitsuhashi. La gráfica de en medio es un acercamiento a la zona de la señal marcada con un recuadro rojo en la gráfica de arriba. Se reduce información eliminando algunas formas de onda, en este ejemplo se eliminan los ciclos pintados de azul. Los ciclos pintados con rojo son los que se guardan. Para reconstruir

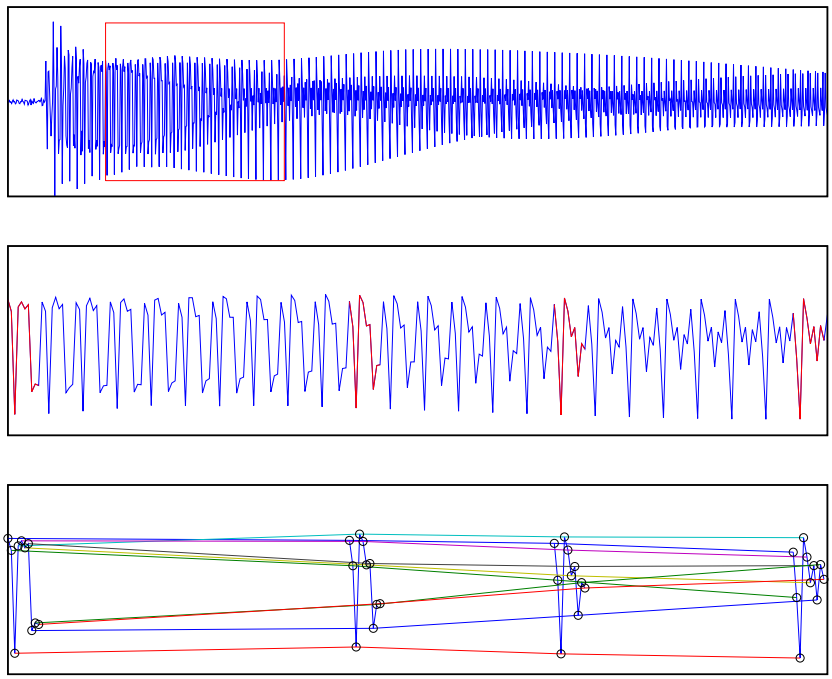


Figura 2.37: Interpolación de formas de onda.

la señal, se interpolan linealmente los nodos para obtener los valores de la forma de onda en cada ciclo. La gráfica de abajo ilustra la interpolación lineal entre formas de onda. Otra manera de realizar la interpolación de formas de onda, es por medio de *crossfade*, como se explica en la sección 2.3.4. En el apéndice C.2 se explica con detalle el algoritmo de reproducción. El listado C.4 presenta este algoritmo en lenguaje C de forma simplificada.

2.4.3. Reproducción cíclica

La reproducción cíclica es la parte más complicada de esta fase de síntesis porque es susceptible a producir discontinuidades y ruidos no deseados en la señal de audio. Una ventaja de la síntesis por segmentación de forma de onda, es que el establecimiento de los puntos de reproducción cíclica no necesita ser tan preciso como se sugiere en el algoritmo de la sección 2.3.3. Esto es porque el algoritmo de síntesis, actualiza los valores de la forma de onda al final del ciclo y no a la mitad, evitando así discontinuidades. En otras palabras, aunque los puntos *loopA* y *loopB*, que delimitan el segmento de reproducción cíclica, se establezcan a la mitad de un ciclo, no entrarán en efecto hasta que haya pasado un ciclo completo.

En la sección 2.4.2 se presentó una versión simplificada de la reproducción cíclica. Ahora contaremos el resto de la historia. Cuando se recibe un mensaje *note-on*, el sintetizador reproduce las formas de onda, desde el ataque, hasta llegar al inicio de la reproducción cíclica *loopA*. El sintetizador continúa reproduciendo hacia adelante las formas de onda hasta llegar al punto *loopB*. En ese instante, el algoritmo de síntesis comienza el conteo de ciclos en reversa, pero mantiene la reproducción de la forma de onda hacia adelante. Esto se logra incrementando el contador *phase* como se explicó en la sección anterior, pero decreciendo el contador *sample*. Para entender mejor este procedimiento, observe la figura 2.38. La gráfica de arriba muestra cinco ciclos de forma de onda con diferentes colores. Los de color negro, que son el *ciclo 1* y el *ciclo 5*, representan el ataque y la relajación respectivamente. Los ciclos de en medio, con colores rojo, azul y verde, representan el segmento de reproducción cíclica. Note que aunque los puntos *loopA* y *loopB* están establecidos a la mitad del ciclo, la reproducción cíclica comienza y termina al final de cada ciclo y no a la mitad. La gráfica de abajo muestra la síntesis de un sonido haciendo uso de la reproducción cíclica. En ella se muestran los colores y números de ciclo, en el orden en que serían reproducidos. Al principio, la reproducción de cada ciclo se da hacia adelante, es decir que el

número de ciclo va creciendo: *ciclo 1, ciclo 2, ciclo 3 y ciclo 4*. En cuanto se ha llegado al punto *loopB*, comienza el conteo de ciclos hacia atrás, en otras palabras, el número de ciclo decrece: *ciclo 4, ciclo 3, ciclo 2*. La reproducción cíclica continuará de la misma manera, una y otra vez, hasta que el usuario suelte la tecla del piano, produciendo un mensaje *note-off*. El resto de la señal continuará reproduciéndose hacia adelante hasta llegar a la última forma de onda. Es importante aclarar, que aunque el conteo de ciclos vaya hacia atrás, la reproducción de cada punto al interior del ciclo será siempre hacia adelante o de izquierda a derecha como se muestra con las flechas punteadas de la gráfica de arriba.

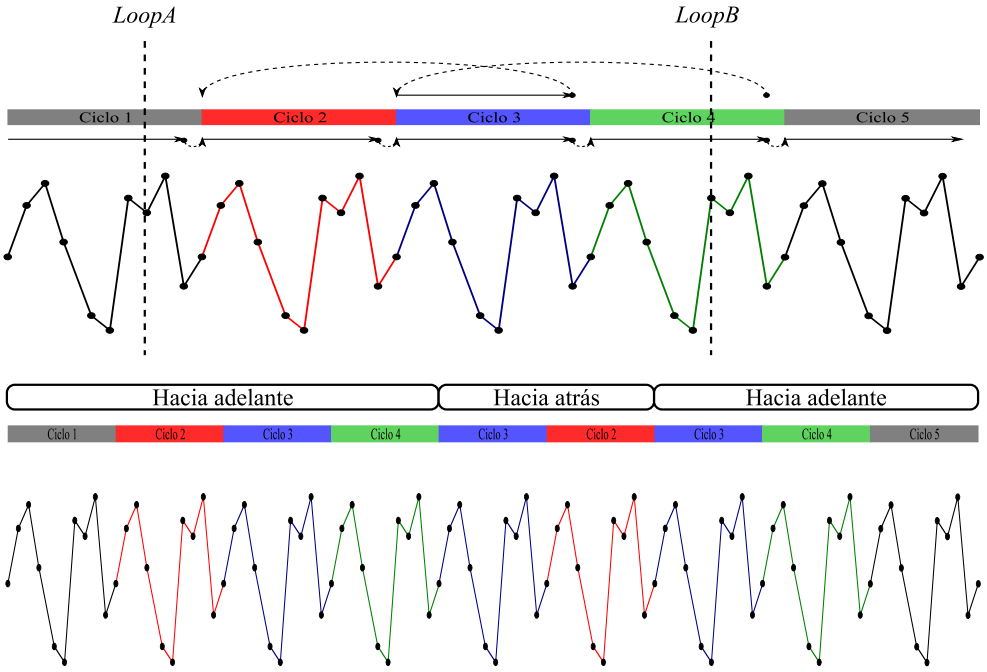


Figura 2.38: Reproducción cíclica.

Para mayor información del algoritmo de reproducción cíclica, consulte el listado C.5. Este es una versión modificada del listado C.4.

2.4.4. Cambio de altura

Un mensaje *note-on* consta de tres valores: el *número de canal*, el *número de nota* y la *velocidad*. Por ahora nos ocuparemos solamente del número de

nota. Una nota MIDI puede tomar un valor entre 0 y 127. La nota A de 440 Hz corresponde al valor MIDI 69. Se puede encontrar la frecuencia f de cada nota MIDI m por medio de la expresión 2.25. Enseguida se explica cómo el algoritmo de síntesis interpreta la información de altura o número de nota y la influencia que el cambio de frecuencia tiene en la duración de la nota.

$$f = 440\left(2^{\frac{m-69}{12}}\right) \tag{2.25}$$

Recordemos que la síntesis de forma de onda se realiza aplicando la ecuación 2.1. Esta ecuación está definida en el dominio $0 \leq x < 2\pi$. En este sentido, resulta similar a una función senoidal. Al igual que en la implementación de un oscilador senoidal, se necesita una variable para la fase θ y otra para el incremento de la fase Δ_θ. El incremento de la fase Δ_θ se relaciona con la frecuencia fundamental f_0 y la frecuencia de muestreo fs por medio de la ecuación 2.26.

$$\Delta_\theta = \frac{2\pi f_0}{fs} \tag{2.26}$$

Cada vez que se calcula una muestra del oscilador, se incrementa el contador θ. Cuando θ llega a 2π, θ comenzará de nuevo desde 0. Una vez que se ha calculado el valor de la fase, se substituye x por θ en $\sin(x)$ para un oscilador senoidal, o en la ecuación 2.1 en el caso de la síntesis por segmentación de forma de onda. El listado C.6 es un ejemplo sencillo, en lenguaje C, de como se implementa un oscilador senoidal.

El valor de Δ_θ también se puede expresar en función del periodo τ. En este caso el periodo τ está dado en segundos.

$$\Delta_\theta = \frac{2\pi}{fs\tau} \tag{2.27}$$

Para que el incremento de la fase Δ_θ no dependa de la frecuencia de muestreo fs, se puede calcular Δ_θ expresando el periodo $\tau_n = fs/f_0$ en muestras en lugar de en segundos.

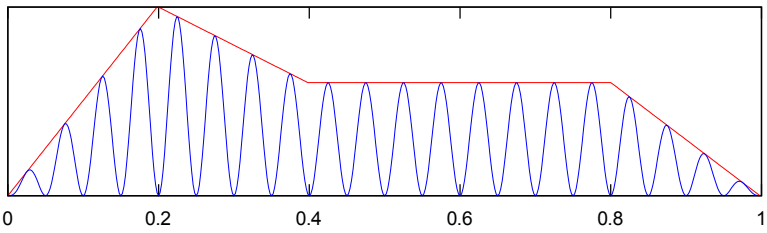
$$\Delta_\theta = \frac{2\pi}{\tau_n} \tag{2.28}$$

El listado C.5 muestra la implementación de la síntesis por segmentación de forma de onda. Observe el uso de las variables phase, phaseIncrement y period.

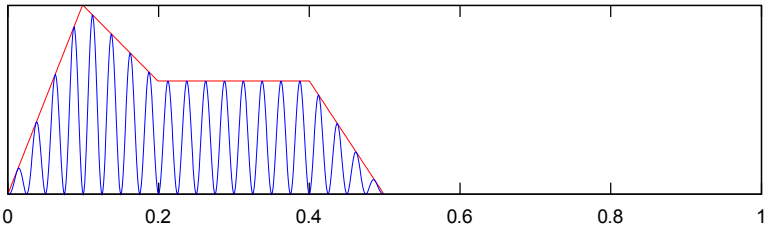
¿Qué sucede con la duración de una nota cuando se cambia su frecuencia? Supongamos que se graba un segundo de un tono de guitarra de 330 Hz que corresponde a la nota *mi* de la primera cuerda al aire. Si queremos utilizar esta señal de audio para producir el mismo tono una octava mas alta, tendremos que duplicar la frecuencia $e' = 2e = 2(330\,\text{Hz}) = 660\,\text{Hz}$. Esto equivale a reproducir la señal al doble de velocidad. Por consiguiente, la reproducción del tono grabado, durará la mitad del tono original, es decir 0.5 sec. Este efecto se da comúnmente en los *samplers*. La nota durará cada vez menos, mientras más aguda sea la nota que se quiere reproducir. Esto provoca que el ataque y los cambios en la forma de onda sucedan más rápido, produciendo así, un cambio en el timbre del sonido. Lo ideal es que se pueda cambiar la frecuencia fundamental del tono, sin alterar la duración de los diferentes segmentos de la señal.

La figura 2.39 ejemplifica el cambio de altura. En la figura 2.39(a) se muestra una señal de 10 Hz con una envolvente típica ADSR y una duración de un segundo. La figura 2.39(b) es la señal transportada una octava. Al transportar la señal se duplica la frecuencia (20 Hz) y se comprime el tiempo (0.5 seg) a la mitad. Para poder mantener la duración de la señal y la forma de la envolvente temporal originales, es necesario agregar más ciclos. Note que la figura 2.39(c), tiene la duración original de 1 s pero contiene el doble de ciclos.

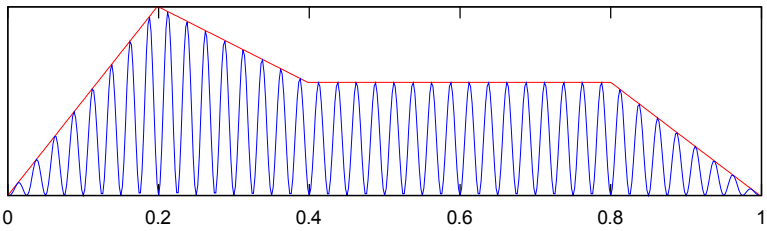
La ventaja del algoritmo de síntesis por segmentación de forma de onda presentado en los listados C.4 y C.5, es que no se altera la duración de la nota cuando se cambia la frecuencia fundamental. Esto es porque el instante de tiempo en que sucede el cambio de una forma de onda a la otra está especificado en muestras y no en número de ciclos. Esta información de tiempo está almacenada en el vector timeStamp de la estructura MitsuData. Esto implica, como se muestra en la figura 2.39, que el número de ciclos dentro de un segmento determinado de tiempo, cambia de acuerdo con la frecuencia fundamental.



(a) Señal original de 10 Hz



(b) Señal transportada una octava (20 Hz) con compresión del tiempo



(c) Señal transportada una octava (20 Hz) sin compresión del tiempo

Figura 2.39: Cambio de frecuencia fundamental

Capítulo 3

Predicción Lineal (LPC)

3.1. Antecedentes

La teoría de la predicción lineal, que data de los años 1940's, ha tenido un gran impacto en el ámbito del procesamiento de señales [45]. Por ejemplo, la predicción lineal es el fundamento de varias técnicas modernas de extracción del espectro de potencia de una señal. La predicción lineal ha sido usada exitosamente en la representación, modelado, compresión y síntesis de voz por computadora. La idea de pares de líneas espectrales se usa en la compresión de voz basada en modelos perceptuales. Otro producto de esta teoría, son las estructuras de filtrado *lattice*. Estas estructuras han sido de interés debido a su estabilidad y robustez numérica [45]. Entre los pioneros de la predicción lineal se encuentran Levinson [46], Weiner [47] y Masani [48]. El excelente tutorial de Makhoul [2] resulta indispensable para cualquiera que quiera obtener información sobre el tema. Vaidyanathan [45] presenta la teoría de la predicción lineal de una manera muy clara y detallada. A pesar de que la predicción lineal data de los años 1940's, en la actualidad se siguen encontrando nuevas aplicaciones.

En el área de la codificación de voz se pueden consultar los trabajos de Atal y Hanahuer [49], Itakura y Saito [50], Makhoul [2], Rabiner y Schafer [51]. Para información más reciente sobre codificación de voz, consulte los trabajos de Deller [52], Gold [53], den Brinker [54], García [55], Ghido [56], Schroeder [57] y Spanias [14].

Existen varios trabajos sobre Predicción Lineal aplicada a la síntesis de instrumentos musicales. Moorer [58] presenta detalles de implementación de

la predicción lineal aplicada a la computación musical. Sandler [59,60] aplica la predicción lineal a la síntesis de instrumentos de percusión. Lansky [61] proporciona una buena introducción a la predicción lineal aplicada a la música por computadora. Gagné [5] describe una forma de sintetizar una guitarra-bajo por medio de predicción lineal. Según su artículo, el tono sintetizado por medio de predicción lineal es indistinguible del sonido original cuando se utiliza en un modelo de doceavo orden.

En este capítulo se explica la teoría de la predicción lineal y se presentan detalles de implementación aplicados a la síntesis de instrumentos musicales. También se discute su uso en combinación con la síntesis por segmentación de forma de onda.

3.2. Explicación del modelo LPC

Considérese un modelo de producción sonora que consiste en una fuente de excitación y un resonador. En dicho modelo, una señal, que funciona como fuente excitadora, es pasada por un filtro que simula las resonancias (Fig. 3.1). La Predicción Lineal es una herramienta que nos permite obtener este filtro a partir de una señal dada [62].

Figura 3.1: Modelo de producción sonora

En la Predicción Lineal, una señal de salida s_n, es representada como un promedio ponderado de muestras anteriores, más una señal de entrada u_n [2]. El valor p representa el *orden* de predicción y determina el número de coeficientes de ponderación a_k que definen el filtro.

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + G u_n \tag{3.1}$$

La señal de entrada u_n es una señal arbitraria que es usada como fuente de excitación en la fase de síntesis. Para extraer la envolvente espectral del sonido, hay que encontrar los coeficiente de ponderación a_k que definen el filtro. Cuando no se conoce la señal excitadora u_n, se asume que la señal

original s_n puede ser aproximada por el promedio ponderado $\tilde{s}_n$ de p muestras anteriores.

$$\tilde{s}_n = -\sum_{k=1}^{p} a_k s_{n-k} \tag{3.2}$$

Así se obtiene la diferencia o error e_n entre el valor real s_n y valor aproximado $\tilde{s}_n$.

$$e_n = s_n - \tilde{s}_n = s_n + \sum_{k=1}^{p} a_k s_{n-k} \tag{3.3}$$

Si despejamos s_n de la ecuación 3.3, se puede expresar la señal original s_n, como la suma de la señal aproximada $\tilde{s}_n$ y la señal de error e_n.

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + e_n \tag{3.4}$$

Al comparar las ecuaciones 3.1 y 3.4, podemos observar que sólo difieren en la señal de error e_n y la señal de excitación Gu_n. Entonces, para aproximar la señal original s_n, podemos sustituir la señal de error e_n por una señal de excitación Gu_n que tenga un espectro similar. Como más adelante veremos, la señal de error e_n tiene un espectro plano. Esta característica plana, es una consecuencia del cálculo de los coeficientes a_k del filtro predictor. Por consiguiente, podrá utilizarse como señal de excitación u_n, cualquier señal con espectro plano.

Los parámetros a_k se obtienen como resultado de minimizar el error cuadrático total e_n^2 . Para obtener los parámetros a_k se resuelve un sistema de ecuaciones por medio de la recursión de Levinson-Durbin [2], que involucra el cálculo de la auto-correlación de la señal de entrada. El factor de ganancia G se calcula de tal manera que la energía total de la señal de entrada Gu_n sea igual a la energía total de la señal de error e_n. En la sección 3.3.3 se explica este procedimiento con mayor detalle. Lo importante, por ahora, es notar que cuando se minimiza el error cuadrático para obtener los parámetros del predictor, el error resultante tiene un espectro plano. En otras palabras, por medio de la Predicción Lineal, se puede representar el espectro de la señal original en términos de su envolvente espectral.

Si se utiliza la señal de error e_n como fuente de excitación u_n en la fase de síntesis, la señal original puede ser reconstruida de manera exacta. Sin

embargo, guardar la señal de error no es práctico, pues contiene tanta información como la señal original. Es por ello que la característica plana del espectro de la señal de error, ha sido aprovechada para codificar voz de una manera económica. Dos tipos de señal son de particular interés: un impulso y ruido aleatorio. Estas dos señales tienen un espectro plano y pueden ser utilizadas como señal de excitación a la hora de la síntesis. Ambas son fáciles de generar y tan sólo se requiere almacenar el factor de ganancia, lo que representa un gran ahorro de almacenamiento. Tradicionalmente se utiliza un tren de impulsos cuando el sonido a sintetizar tiene un carácter tonal, como por ejemplo, una vocal. Por el contrario, si se quiere sintetizar un sonido de carácter no tonal, como el sonido de la letra *s*, entonces se alimenta el predictor con ruido.

Normalmente, en la síntesis de instrumentos musicales, se utiliza el mismo criterio que en síntesis de voz: un tren de impulsos para sintetizar sonidos con componentes armónicas [5], y ruido para generar sonidos percusivos como el de un tambor [60]. Esta visión de la LPC limita su aplicación a un cierto tipo de instrumentos cuya señal de excitación es similar a un tren de impulsos o ruido aleatorio. El hecho de que la fuente de excitación puede ser cualquier señal con espectro plano, nos da un gran número de opciones. El problema está, en que otras señales son más difíciles de generar que un impulso o ruido. Una solución es utilizar la síntesis por segmentación de forma de onda como fuente de excitación, para alimentar el predictor. El método de Mitsuhashi es fácil de implementar en el dominio del tiempo y aumentaría las posibilidades de aplicación de la LPC.

3.3. Análisis

La predicción lineal en su fase de análisis es más sencilla que el análisis de Mitsuhashi presentado en la sección 2.3. Afortunadamente, ambas fases de análisis son independientes y pueden realizarse de manera modular. A pesar de que la obtención de parámetros a partir de la predicción lineal es muy sencilla, el cálculo del factor de ganancia y la determinación del orden óptimo del predictor presentan algunas dificultades.

3.3.1. Tamaño y avance de la ventana de análisis

Según Moorer [58], en análisis de voz, comúnmente se utiliza una ventana de tamaño fijo, por ejemplo de 25 milisegundos. La ventana de análisis se avanza en pasos uniformemente espaciados, como por ejemplo, cada 15 milisegundos. Así, los segmentos de análisis se traslapan. El problema con este esquema es la inconsistencia en el número de ciclos que se capturan de la señal. Por ejemplo, una ventana de 25 milisegundos podría capturar dos ciclos de un segmento de señal y tres ciclos en otro segmento, dependiendo de la frecuencia instantánea de la señal. Esto nos da una gran variación en la estimación espectral de diferentes segmentos. Una manera de aminorar estas variaciones, es utilizar una ventana más grande que contenga un número mayor de ciclos. La desventaja, en el caso de la síntesis de voz, es que el uso de ventanas grandes puede causar traslapes de segmentos de consonantes y vocales, haciendo más difícil la determinación del tipo de señal de excitación (ruido o tren de impulsos). Además, al utilizar una ventana más grande que un ciclo, el análisis puede detectar la estructura fina del espectro, en lugar de su envolvente, aunque ello también depende del orden de predicción, como veremos en la sección 3.3.4. Moorer sugiere hacer un análisis sincronizado al periodo, de tal forma que tanto el tamaño de la ventana, como su avance en el tiempo, sea de exactamente un ciclo. La figura 3.2 muestra el análisis sincronizado al periodo propuesto por Moorer.

Gagné [5] sugiere realizar un análisis sincronizado al periodo, de forma que la ventana de análisis contenga cuatro ciclos de la forma de onda. Es decir, que la ventana tendrá un tamaño de cuatro ciclos. Al segmento de análisis se le aplica una ventana Hanning para mejorar el cálculo de la autocorrelación por medio de la FFT. El análisis se realiza en bloques traslapados que van avanzando de un ciclo a otro. En la figura 3.3 se muestra el análisis traslapado propuesto por Gagné [5]. Observe que cada ventana de análisis contiene exactamente cuatro ciclos. En opinión de este autor, no es necesario que el análisis está sincronizado al periodo, siempre y cuando haya un número suficiente de ciclos para que la periodicidad del tono se manifieste. Si se avanza la ventana en cada ciclo, se generarán muchos datos. Se puede hacer un traslape más grande si se quiere ahorrar información. También se puede hacer un traslape mas fino en las partes de la señal que son más cambiantes, como el ataque, y un traslape más burdo en las partes de la señal que son más estáticas, como la parte sostenida. En nuestra implementación se realiza el análisis con un tamaño de ventana y avance fijos, que son establecidos por

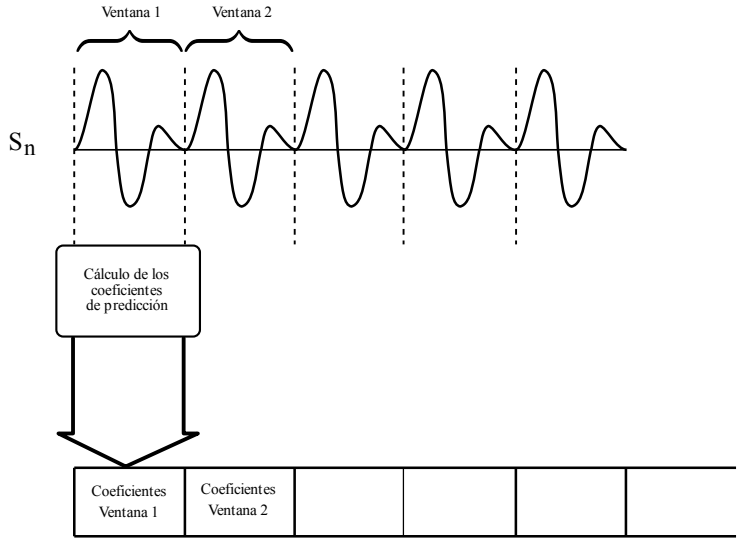


Figura 3.2: Análisis sincronizado al periodo según Moorer. [5]

el usuario.

3.3.2. Obtención de los coeficientes del predictor

Como se explicó en la sección 3.2, la diferencia o error e_n entre el valor real s_n y valor predicho $\tilde{s}_n$ está dada por la ecuación 3.3. Los parámetros a_k del predictor se obtienen como resultado de minimizar el error cuadrático total $E = \sum_n e_n^2$.

$$E = \sum_n \left(s_n + \sum_{k=1}^{p} a_k s_{n-k} \right)^2 \tag{3.5}$$

En el método de autocorrelación descrito por Makhoul [2], se minimiza el error cuadrático total en el intervalo $-\infty < n < \infty$, obteniendo lo que se conoce como *ecuaciones normales*, pero expresadas en forma de autocorrelación:

$$\sum_{k=1}^{p} a_k R(i-k) = -R(i), \qquad 1 \leq i \leq p. \tag{3.6}$$

Podemos expandir la ecuación 3.6 en forma matricial.

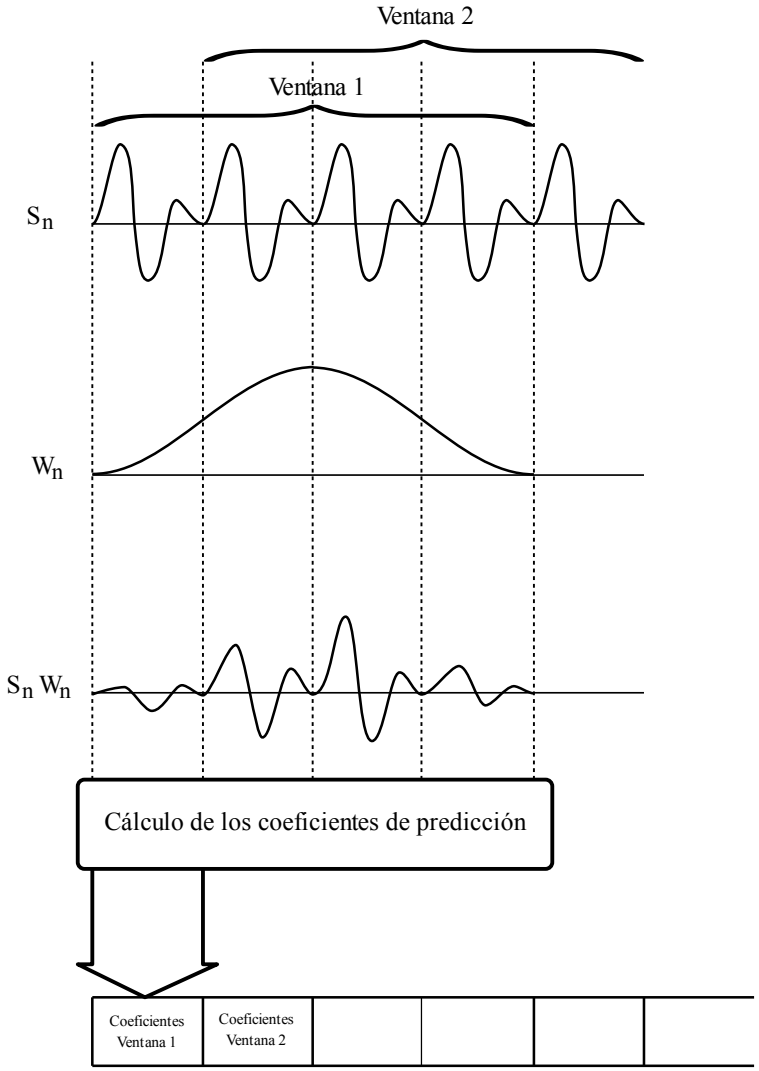


Figura 3.3: Tamaño de la ventana de análisis y traslape según Gagné [5]

$$\begin{pmatrix} R_0 & R_1 & R_2 & \cdots & R_{p-1} \\ R_1 & R_0 & R_1 & \cdots & R_{p-1} \\ R_2 & R_1 & R_0 & \cdots & R_{p-1} \\ \vdots & \vdots & \vdots & & \vdots \\ R_{p-1} & R_{p-2} & R_{p-3} & \cdots & R_0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_p \end{pmatrix} = - \begin{pmatrix} R_1 \\ R_2 \\ R_3 \\ \vdots \\ R_p \end{pmatrix} \tag{3.7}$$

Podemos encontrar un conjunto único de coeficientes de predicción a_k, siempre que la matriz de autocorrelación 3.7 no sea singular. Esta condición se cumple, en general, cuando se utilizan un segmento de señal muestreada que es considerada estacionaria localmente, es decir, que sus propiedades estadísticas son constantes dentro de ese segmento de análisis[1].

Para obtener los coeficientes a_k, hay que resolver el sistema de ecuaciones 3.7 a partir de los valores conocidos de la autocorrelación $R(k)$. Note que la matriz es simétrica y los valores a lo largo de cada diagonal son iguales. Este tipo de matriz es conocida como matriz *Toeplitz*. Una ventaja de este tipo de matriz es que se puede resolver el sistema de ecuaciones utilizando la recursión de Levinson-Durbin[2] que es un algoritmo que requiere menos tiempo de cómputo y memoria que otros algoritmos. La recursión de Levinson-Durbin es muy rápida, pero es susceptible a errores de redondeo. Una solución alternativa, es la descomposición de Cholesky, que es numéricamente más estable, pero requiere un mayor número de operaciones. En el tutorial de Makhoul [2], se menciona que en la práctica, no se ha encotrado que la inestabilidad numérica sea un problema. La recursión de Levinson-Durbin está dada por las siguientes expresiones:

$$E_0 = R(0) \tag{3.8}$$

$$k_i = \frac{-1}{E_{i-1}} \left(R(i) + \sum_{j=1}^{i-1} a_j^{(i-1)} R(i-j) \right) \tag{3.9}$$

[1]Formalmente, se dice que la señal es *estacionaria en sentido amplio*. Como consecuencia de esta propiedad estacionaria, la matriz de autocorrelación R_k tiene la característica *Toeplitz* y el sistema no es singular. Consulte [45] para una explicación detallada sobre las propiedades de la matriz de autocorrelación.

[2]El algoritmo de Levinson-Durbin fué propuesto primeramente por Levinson y mejorado posteriormente por Durbin.

$$a_i^{(i)} = k_i \tag{3.10}$$

$$a_j^{(i)} = a_j^{(i-1)} + k_i a_{i-j}^{(i-1)}, \qquad 1 \leq j \leq i-1 \tag{3.11}$$

$$E_i = (1 - k_i^2) E_{i-1}. \tag{3.12}$$

Las ecuaciones 3.9-3.12 se resuelven recursivamente para $i = 1, 2, \ldots, p$ y la solución final está dada por

$$a_j = a_j^{(p)}, \qquad 1 \leq i \leq p. \tag{3.13}$$

En el listado C.7 se presenta un pseudocódigo para el algoritmo de Durbin. El código es una traducción casi directa de las ecuaciones 3.8-3.13. Una ventaja del algoritmo de Levinson-Durbin es que durante el cálculo de los coeficientes, se obtiene el valor del error E_i en cada iteración. Además, como consecuencia del cálculo, se obtienen los valores intermedios k_i llamados *coeficientes de reflexión*. Es preferible utilizar estos coeficientes de reflexión k_i para representar el filtro predictor, que los coeficientes a_k del modelo autorregresivo de la ecuación 3.1 por varias razones:

- La respuesta a impulso a_k y la autocorrelación $R(k)$ son muy susceptibles a causar inestabilidades en el filtro debido a errores de redondeo derivados de la precisión numérica o cuantización[3].
- Los coeficientes de reflexión k_i, por otra parte, garantizan la estabilidad del filtro, siempre y cuando se se mantenga $|k_i| < 1$ después de la cuantización [45].
- Los coeficientes de reflexión se pueden utilizar para implementar filtros lattice[4].
- Los coeficientes de reflexión se pueden interpolar sin riesgo a que se produzcan inestabilidades, siempre y cuando la magnitud de los coeficientes sea menor a la unidad $|k_i| < 1$. [58]

[3]La representación de un valor por medio de un número finito de dígitos. Mientras mayor sea la precisión numérica, menores serán los errores de redondeo.

[4]Palabra tomada del inglés *lattice filter*. En este trabajo usaremos la palabra *lattice* en inglés sin traducción.

- De los coeficientes de reflexión k_i, se pueden derivar *parejas de líneas espectrales*[5] que son utilizadas en la compresión de señales. La razón es que las *parejas de líneas espectrales*, a diferencia de los coeficientes de reflexión, se pueden interpretar mejor desde el punto de vista de la percepción humana. En este trabajo no ahondaremos más en este tema, pero puede consultar el libro de Vaidyanatan [45] para más información.

En la obtención de los parámetros del predictor, el cálculo de la autocorrelación $R(k)$ consume más recursos de cómputo que la solución al sistema de ecuaciones 3.7. Una manera de reducir el tiempo de cómputo, es calcular la autocorrelación por medio de la FFT, como la transformada inversa del espectro de potencia:

$$R(i) = \frac{1}{M} \sum_{m=0}^{M-1} P(\omega_m) \cos(i\omega_m). \tag{3.14}$$

El algoritmo completo para calcular los parámetros de la predicción lineal es el siguiente:

1. Escoger un segmento de la señal de tiempo con una longitud de por lo menos 4 ciclos.

2. Aplicar una ventana Hanning.

3. Calcular la autocorrelación de la señal ventaneada por medio de la FFT. Es opcional, pero recomendable, normalizar a 1 la autocorrelación de esta manera $r(k) = R(k)/R(0)$.

4. Obtener los coeficientes a_k del filtro y/o los coeficientes de reflexión k_i utilizando la recursión de Levinson-Durbin.

Como vimos anteriormente, en el cálculo de la autocorrelación, se asume que la señal es locamente estacionaria. Esto implica que las señales se pueden aproximar mejor, cuando su espectro evoluciona mas lento en el tiempo. Los sonidos con transiciones rápidas, serán más difíciles de modelar. El éxito del modelado por medio de predicción lineal, depende en gran parte, de los parámetros que se utilizan en el análisis. De particular importancia es el orden de predicción, del cual hablaremos en la sección 3.3.4, donde también

[5]Del inglés *line-spectrum pairs*.

se explican los tipos de procesos que se pueden aproximar por medio de este modelo.

Utilizando el entorno de programación Octave [8, 9], se pueden calcular los parámetros de la predicción lineal utilizando el paquete *Signal* que se puede obtener de la página web de *Octave-Forge* [63]. Este paquete contiene una función levinson que realiza la recursión de Levinson-Durbin.

El listado C.8 muestra una función lpc escrita en Octave que ilustra cómo se obtienen los parámetros de la predicción lineal.

3.3.3. Cálculo del factor de ganancia

El factor de ganancia G en la ecuación 3.1, se utiliza para controlar el nivel de amplitud de la señal sintetizada. Para calcular este factor, se asume que la señal sintetizada $\hat{s}_n$ tiene la misma energía que la señal original s_n sin importar cual sea la señal de excitación u_n. Esto implica que la energía total de la señal excitadora Gu_n sea igual a la energía total de la señal de error e_n. Si se utiliza un impulso o ruido aleatorio como señal de excitación u_n, se puede calcular el factor de ganancia G a partir de la autocorrelación $R(i)$ de la señal original s_n, utilizando la siguiente expresión. [2]

$$G^2 = E_p = R(0) + \sum_{k=1}^{p} a_k R(k), \tag{3.15}$$

donde G es el factor de ganancia, $R(i)$ es la autocorrelación de la señal original y E_p es la energía total de la señal de error.

En la práctica, no conviene obtener el factor de ganancia de manera directa, aplicando la ecuación 3.15. Una razón es que al realizar el análisis se aplica una ventana, por ejemplo, una ventana Hanning. Entonces, el cálculo de la energía de la señal de error E_p, a partir de la autocorrelación $R(i)$, estará influenciado por la ventana de análisis. En consecuencia, la energía de la señal sintetizada, no coincidirá con la energía de la señal original. Aunque se puede aplicar un factor de corrección, existe otro problema. El filtro que se utiliza en la fase de síntesis, es un filtro recursivo que tiene una respuesta a impulso infinita. Esto quiere decir, que el filtro seguiría emitiendo una respuesta en los bloques posteriores, y por consiguiente, la energía se irá acumulando de un bloque a otro. En el cálculo de la ganancia, a partir de la ecuación 3.15, no se toma en cuenta la "memoria" del filtro.

Atal y Hanauer [49], proponen un método para calcular el factor de ganancia, tomando en cuenta la "memoria" del filtro. En el método de Atal y Hanahuer, el valor de cada muestra de la señal sintetizada s_n se puede descomponer en dos partes. Primero, la memoria del filtro q_n, cuyos valores contienen la respuesta acumulada de segmentos de síntesis anteriores, donde q_n está dado por la expresión 3.16.

$$q_n = \sum_{k=1}^{p} a_k q_{n-k} \tag{3.16}$$

Segundo, la contribución de la señal de excitación Gv_n en el segmento de síntesis actual. Es decir, $s_n = q_n + Gv_n$, donde v_n está dado por la expresión 3.17 y u_n representa la señal de excitación a la entrada del filtro.

$$q_n = \sum_{k=1}^{p} a_k v_{n-k} + u_n \tag{3.17}$$

Si tomamos un segmento de audio que va de la muestra $n = 1$ a la muestra $n = M$, podemos calcular el valor cuadrático medio P_s de ese segmento de la manera siguiente:

$$P_s = \frac{1}{M} \sum_{n=1}^{M} (q_n + Gv_n)^2 = \overline{(q_n + Gv_n)^2} \tag{3.18}$$

Se puede escribir la ecuación 3.18 como sige:

$$G^2 \overline{v_n^2} + 2G\overline{q_n v_n} + \overline{q_n^2} - P_s = 0, \tag{3.19}$$

En el método de Atal y Hanauer se calcula el factor de ganancia G resolviendo la ecuación cuadrática 3.19.

donde G es el factor de ganancia, q_n es la memoria del filtro con la respuesta acumulada de segmentos de síntesis anteriores, v_n es la contribución de la señal de excitación en el segmento actual y P_s es el valor cuadrático medio del bloque correspondiente de la señal original. Al resolver la ecuación 3.19 para G se escoge un valor real positivo. En caso de que no exista solución, se establece $G = 0$.

El cálculo del factor de ganancia a partir de la ecuación 3.19 tiene ciertas desventajas. Cuando la energía representada por la respuesta acumulada del filtro en bloques anteriores, excede la energía deseada del bloque actual, la

solución a la ecuación cuadrática sería un número complejo. Además, este método presenta inestabilidades, por ejemplo, cuando la energía del bloque anterior es menor pero muy cercana a la energía del bloque actual, el cálculo de la ganancia sería un valor muy pequeño. Esto causará que la contribución de la "memoria" del filtro, en el bloque siguiente, sea muy poca y el cálculo del siguiente factor de ganancia resultará muy grande [58].

En este trabajo, se utilizan tres métodos de control de la ganancia. El primero, consiste en normalizar la señal sintetizada igualando las energías de la señal original y la sintetizada. En el segundo método, también se normaliza la señal sintetizada, pero se igualan sus amplitudes máximas. El tercero, es un método experimental para encontrar la envolvente temporal de amplitud por medio de predicción lineal [64]. En la práctica, la normalización por medio de amplitudes máximas, parece dar mejores resultados.

El algoritmo para el control de la amplitud, normalizando la energía de la señal sintetizada se describe a continuación:

- Se analiza la señal original s_n en pequeños segmentos de tiempo.
- Se obtienen los coeficientes a_k del filtro predictor de cada segmento, por medio de predicción lineal.
- Se calcula el valor RMS de cada segmento de la señal original s_n y se almacenan estos valores junto con los coeficientes de predicción. Para obtener el valor RMS en otros puntos de la señal, se interpolan los valores linealmente.
- En la fase de síntesis, se almacena el valor de cada muestra de la señal sintetizada $\hat{s}_n$ en un arreglo temporal.
- Se calcula el valor RMS del arreglo temporal para encontrar el factor de ganancia G, de tal manera que la energía de la señal de original s_n coincida con la energía de la señal sintetizada $\hat{s}_n$ en el correspondiente instante de tiempo.

Enseguida se describe el algoritmo para el control de la amplitud, normalizando la amplitud máxima de la señal sintetizada:

- Se analiza la señal original s_n en pequeños segmentos de tiempo.
- Se obtienen los coeficientes a_k del filtro predictor de cada segmento, por medio de predicción lineal.

- Se obtiene la amplitud máxima en ciertos segmentos de la señal original s_n y se almacenan estos valores. Para obtener el valor de la envolvente de amplitud en otros puntos de la señal, se interpolan los valores linealmente.

- En la fase de síntesis, se almacena el valor de cada muestra de la señal sintetizada $\hat{s}_n$ en un arreglo temporal.

- Se obtiene la amplitud máxima del arreglo temporal para encontrar el factor de ganancia G, de tal manera que la amplitud máxima de la señal original s_n coincida con la amplitud máxima de la señal sintetizada $\hat{s}_n$ en el correspondiente instante de tiempo.

Para el algoritmo anterior, se puede encontrar la envolvente temporal por medio de predicción lineal usando los siguientes pasos:

- Se toma la señal original s_n como si fuera el espectro de amplitud S_n.

- Se completa el espectro S_n, agregando los valores complejos conjugados del mismo espectro invertido.

- Se obtiene la autocorrelación R_n aplicando la transformada inversa de Fourier al espectro de potencia $P_n = S_n^2$.

- Utilizando el algoritmo de Levinson-Durbin, se obtienen los coeficientes a_k del filtro.

- La envolvente temporal estará definida por la respuesta en frecuencia de los coeficientes a_k.

El problema con este último algoritmo es decidir el orden p más adecuado del predictor 3.1. Además, los valles no se aproximan tan bien como las crestas. Esto puede ser un problema para aproximar el inicio del ataque de un instrumento musical.

Observe las figuras 3.4, 3.5, 3.6 y 3.7. Con estas figuras se ilustran los tres métodos de cálculo de la ganancia. La figura 3.4 representa la señal original en el dominio del tiempo. La figura 3.5 muestra el valor cuadrático medio de la señal original. La figura 3.6 muestra la envolvente temporal de amplitud. En este caso, la envolvente se obtuvo al encontrar la amplitud máxima local para diferentes segmentos de la señal. Otra manera de obtener la envolvente

temporal, es utilizando la predicción lineal. El resultado se muestra en la figura 3.7.

Las figuras 3.8 y 3.9, muestran un diagrama de flujo de los algoritmos anteriormente expuestos. Primero, se analiza la señal original s_n y se obtienen los coeficientes de predicción a_k que serían guardados en un archivo. Por otro lado, se calcula el valor cuadrático medio s_{rms} o la envolvente de amplitud s_{amp}, según sea el caso. Estos valores para el control de la ganancia se almacenan también en un archivo. En la fase de síntesis, una señal excitadora cualquiera u_n, es pasada por el filtro predictor, cuyos coeficientes a_k, habían sido guardados en un archivo. Note que durante el filtrado, no se aplica ningún factor de ganancia. La señal de excitación filtrada $\hat{s}_n$ se multiplica por el factor de ganancia G para obtener la aproximación final $\tilde{s}_n$.

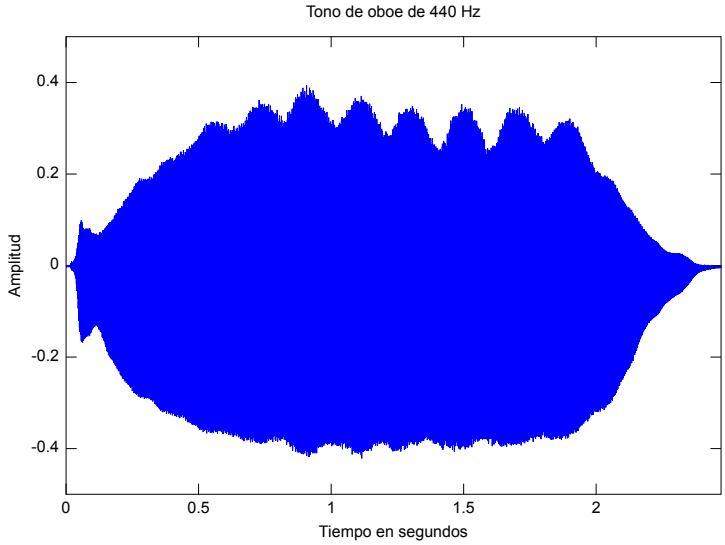


Figura 3.4: Tono de oboe de 440Hz.

3.3.4. Orden de predicción

Otro aspecto importante en la predicción lineal, es la determinación del orden de predicción p. La elección de un orden de predicción «óptimo» no es tarea fácil. Se tienen que tomar en cuenta varios factores, por ejemplo: la aplicación que se quiere hacer de la predicción lineal, el tipo de proceso generador de la señal original y las características propias de la señal. Es difícil

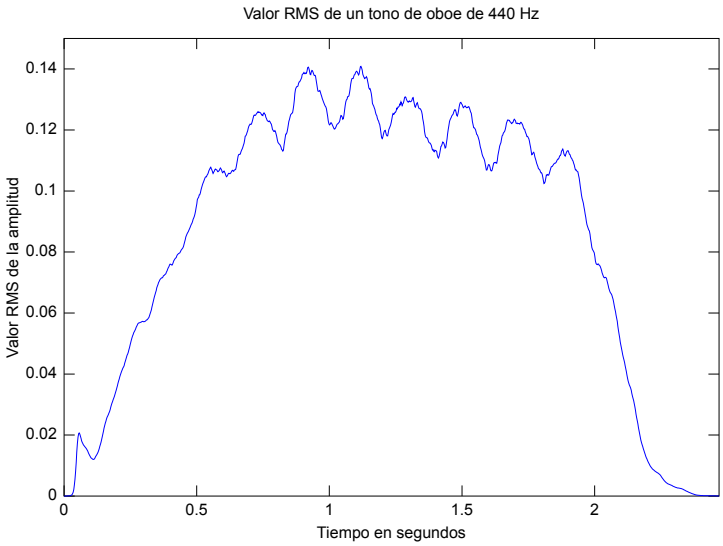


Figura 3.5: Valor cuadrático medio de un tono de oboe de 440Hz.

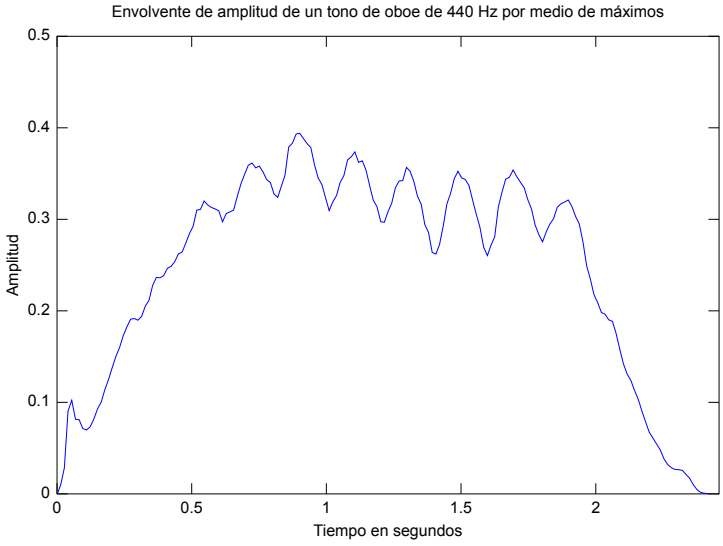


Figura 3.6: Envolvente temporal de amplitud de un tono de oboe de 440 Hz calculado por medio de máximos locales.

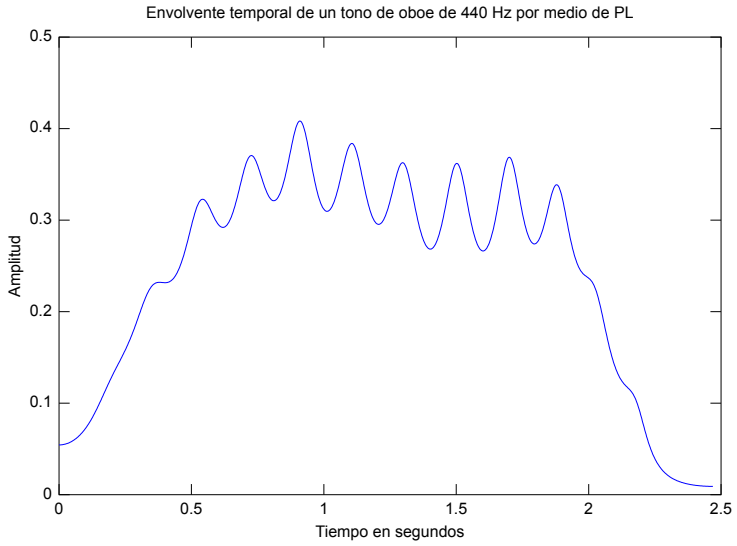


Figura 3.7: Envolvente temporal de amplitud de un tono de oboe de 440 Hz calculado por medio de predicción lineal.

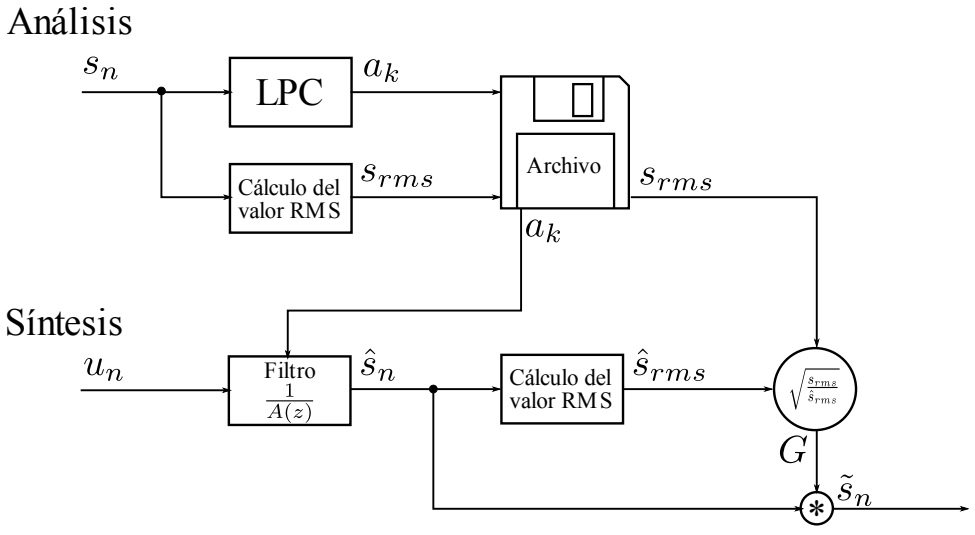


Figura 3.8: Control de la ganancia igualando el valor cuadrático medio.

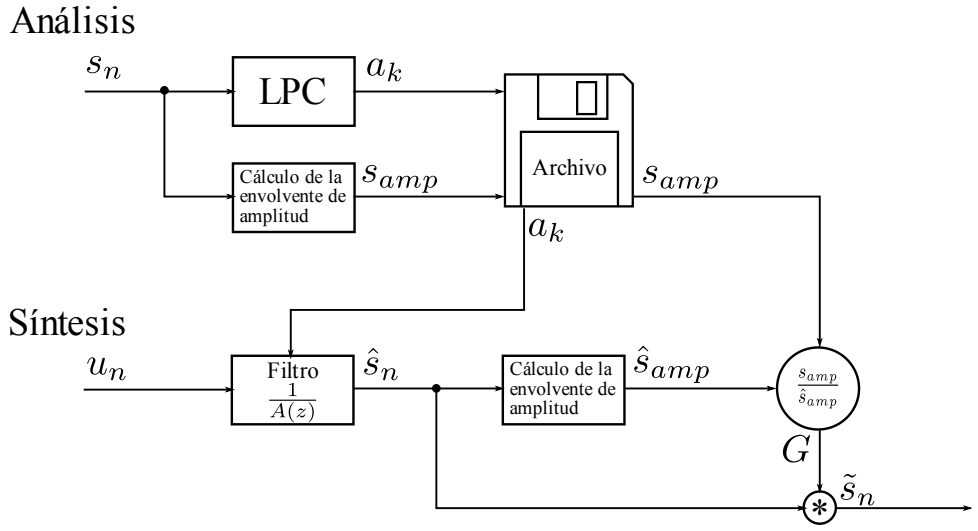


Figura 3.9: Control de la ganancia igualando amplitudes.

establecer un método general, y en muchos casos, el orden se determina de manera empírica. En esta sección se exponen diferentes soluciones y se sugiere un método automático para la elección del orden de predicción. Este método no funciona en todos los casos; pero en caso de fallo, se pueden modificar los parámetros manualmente y ajustar el orden de predicción por medio de ensayo y error para mejorar los resultados.

En esencia, la predicción lineal convierte una señal s_n, en una serie de valores $\{a_k\}$ y una señal de error e_n. Hemos visto que el error e_n tiene un espectro relativamente plano en comparación con la señal original s_n. Para un orden grande p, la señal de error e_n es casi blanca,[6] y la información espectral está contenida principalmente en los coeficientes del filtro $\{a_k\}$ [45]. Suponiendo que se calculan los coeficientes de predicción lineal para órdenes sucesivamente más grandes $p = \{1, 2, 3, 4, \cdots\}$, entonces el error cuadrático total e_p irá decreciendo $e_{p+1} \leq e_p$, mientras que el orden crece. Esto quiere decir, que la señal aproximada $\tilde{s}_n$ se ajusta cada vez mejor a la señal original s_n. ¿Si el modelo de aproximación mejora conforme el orden de predicción aumenta, entonces, en qué momento se debe de parar? Claramente, usaríamos el valor más pequeño de p, adecuado para el problema que se quiere resolver [2]. El orden «óptimo» de predicción depende, entonces, de la aplicación, del tipo de señal y del proceso generador. Por ejemplo, si se quiere utilizar la

[6]Un espectro blanco contiene la misma energía en todas sus bandas de frecuencia.

predicción lineal para comprimir una señal de audio, no conviene utilizar órdenes muy altos, porque el objetivo es ahorrar espacio de almacenamiento. Además, órdenes grandes aumentan la inestabilidad numérica y el consumo de recursos de cómputo.

Ahora, veamos cómo el tipo de proceso involucrado en la generación de una señal, puede influir en la elección del orden óptimo de predicción. Para nuestro estudio, podemos distinguir tres tipos de procesos:

- Autoregresivo *AR(N)*
- No-Autoregresivo *No-AR*
- Líneas espectrales, *Linespec(L)*

La predicción lineal puede revelar el tipo de proceso del que proviene la señal s_n, analizando las características error cuadrático total e_p en función del orden de predicción p. En los párrafos siguientes se discuten y definen estos procesos de manera informal, puede consultar [45] para mayor información.

Proceso Autorregresivo

Se dice que un proceso s_n es autorregresivo, si puede ser generado por la ecuación 3.4 [45]. Dijimos que si se calculan los coeficientes de predicción lineal para órdenes sucesivamente más grandes $p = \{1, 2, 3, 4, \cdots\}$, el error cuadrático total e_p irá decreciendo $e_{p+1} \leq e_p$ en función de p. Ahora, supongamos que cuando el predictor ha llegado a un determinado valor $p = N$, el error se estanca y no decrece más

$$e_N = e_{N+1} = e_{N+2} = e_{N+2} = \cdots \tag{3.20}$$

A esto se le llama *condición de estancamiento* y quiere decir que la precisión del modelo predictor no va a mejorar más, aunque se utilice un número mayor de muestras pasadas. En consecuencia, la señal de error e_n tendrá un espectro blanco. Cuando una señal s_n es la salida de un filtro todos-polos $1/A(z)$, la entrada e_n tiene un espectro blanco y se da la *condición de estancamiento*, se dice que la señal s_n es el resultado de un proceso autorregresivo de orden N. La figura 3.10 muestra un proceso AR(4). La gráfica superior muestra el espectro de potencia. La gráfica inferior muestra la condición de estancamiento, cuando el orden es mayor a 4, el error de predicción permanece constante.

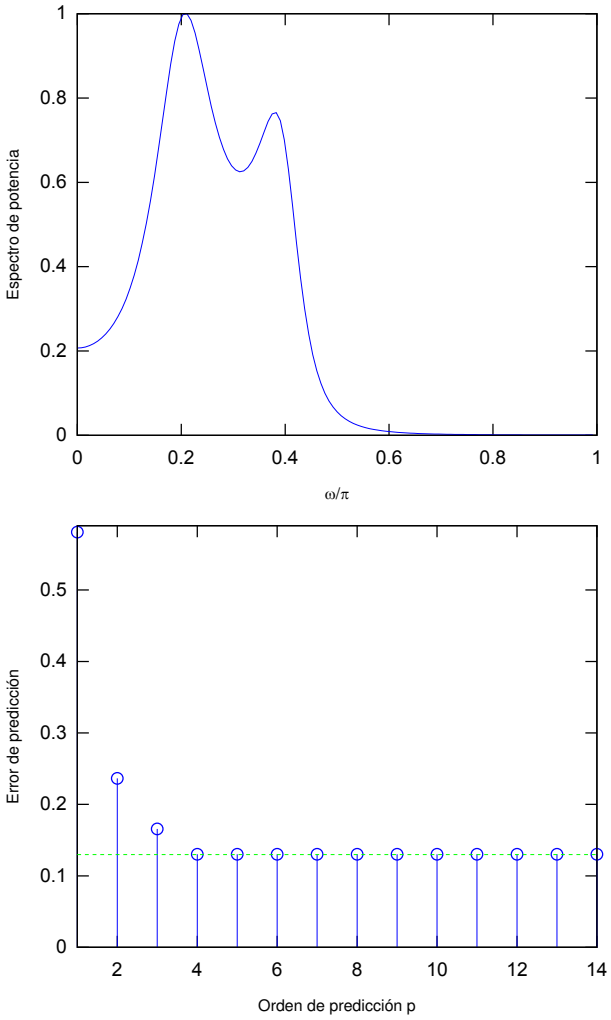


Figura 3.10: Condición de estancamiento en un proceso autorregresivo. Arriba se muestra el espectro de potencia de un proceso AR(4). La gráfica de abajo representa el error de predicción en función del orden. Note que para ordenes $p > 4$ el error permanece constante.

Para determinar el orden del predictor, en el caso de un proceso autorregresivo, se toma el valor del orden donde comienza el estancamiento. En otras palabras, en un proceso AR(N), el orden «óptimo» del predictor sería $p = N$. Sin embargo, como veremos más adelante, este orden puede no ser el «óptimo» para ciertas aplicaciones. Un ejemplo es cuando se trata de extraer una envolvente espectral.

Proceso No-Autorregresivo

Cualquier proceso, sea o no autorregresivo, se puede aproximar por medio de un modelo AR. Cuando se aproxima un proceso No-AR por medio de predicción lineal, el error de predicción e_n no llega a ser blanco; pero en la práctica, la señal de error llega a ser casi blanca para ordenes p razonablemente largos [45]. En este caso, se puede reemplazar la señal de error e_n que es casi blanca, por una señal de excitación u_n con espectro blanco. De esta manera se está aproximando un proceso No-AR con un proceso AR.

Observe la figura 3.11, en ella se muestra el caso de un proceso No-AR. La gráfica de arriba muestra el espectro de potencia. La gráfica de abajo muestra el valor del mínimo error cuadrático total en función del orden de predicción. Note que mientras mayor es el orden, menor son los valores del error, pero estos nunca llegan a estancarse. Esta es una característica de un proceso que no es autorregresivo.

Para determinar el orden «óptimo» de predicción en el caso de un proceso No-AR, se necesita otro criterio, ya que nunca se da la condición de estancamiento. Makhoul [2] sugiere la siguiente prueba para determinar si la curva de error se ha aplanado suficientemente.

$$1 - \frac{V_{p+1}}{V_p} < \delta \tag{3.21}$$

Uno debe aplicar esta prueba para varias muestras consecutivas para asegurarse de que, en verdad, la curva del error normalizado V_p se ha aplanado. Otro criterio mencionado por Makhoul es el de Akaike, dado por la ecuación 3.22.

$$I(p) = \log V_p + \frac{2p}{N_e} \tag{3.22}$$

El orden de predicción «óptimo» estará dado por el valor de p para el que

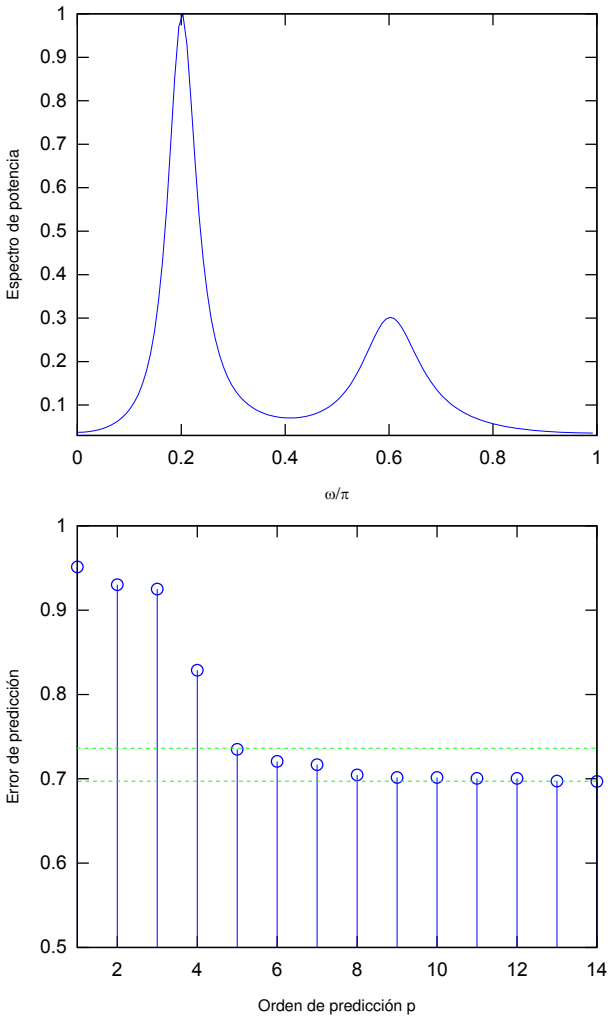


Figura 3.11: Proceso no autorregresivo. El error continúa decreciendo pero no llega a estancarse. Para órdenes grandes, la señal de error es casi blanca.

la función de máxima verosimilitud[7] $I(p)$ es mínima. En la ecuación 3.22, V_p representa la curva del error normalizado y N_e es el número efectivo de datos en la señal. Para una ventana Hanning, el número efectivo de datos es de $N_e = 0.375N$.

En la figura 3.12 se muestra el mismo proceso No-AR que se presentó en la figura 3.11. En la gráfica superior se muestra nuevamente la curva del error en función del orden de predicción. En la gráfica de abajo, se muestra la prueba de Akaike para determinar el orden óptimo de predicción. En este caso, el orden óptimo, marcado con un círculo, es $p = 5$ y corresponde al punto mínimo de la curva $I(p)$. En la siguiente sección veremos que cuando se aplica la predicción lineal a un tono armónico, el orden óptimo obtenido con la prueba de Akaike coincide con el periodo $p = T$ del tono. Por esta razón la prueba de Akaike podría servir para estimar la frecuencia fundamental de un tono armónico, pero de manera muy burda.

Proceso de líneas espectrales

Un proceso de líneas espectrales de orden L es, en esencia, una suma de $L/2$ senoidales. El espectro de potencia de un proceso de líneas espectrales consiste únicamente de impulsos o líneas verticales, de allí el nombre de líneas espectrales.

$$x(n) = \sum_{i=1}^{L/2} A_i \cos(\omega_i n + \theta_i) \tag{3.23}$$

En su forma compleja:

$$x(n) = \sum_{i=1}^{L} d_i e^{j\omega_i n} \tag{3.24}$$

Note que en la ecuación 3.24 no se perdió la fase θ_i. Cada senoidal de la ecuación 3.23 corresponde a dos líneas espectrales de la ecuación 3.24. Por eso el límite superior de la sumatoria es L en lugar de $L/2$. La ecuación 3.24 podría escribirse de la siguiente manera:

$$x(n) = \sum_{i=1}^{L/2} (d_i e^{j\omega_i n} + d_i^* e^{-j\omega_i n}) \tag{3.25}$$

[7]En inglés, Maximum likelihood

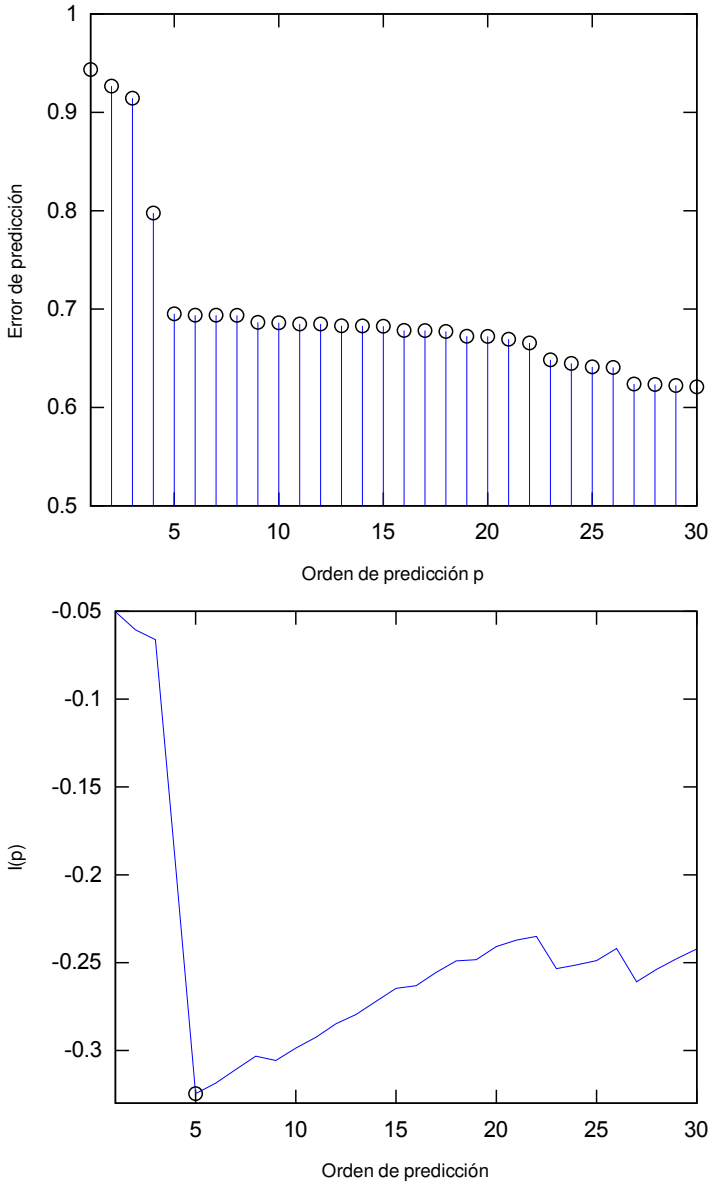


Figura 3.12: La prueba de Akaike para encontrar el orden óptimo de predicción. En este caso el orden óptimo esta marcado con un círculo y es el punto donde I(p) tiene el valor mínimo global.

donde d_i^* es el complejo conjugado de d_i.

Para que la suma de senoidales en 3.23 sea considerada $Linespec(L)$, las amplitudes A_i y las fases θ_i deben ser estadísticamente independientes[8]. Para una definición formal de un proceso $Linespec(L)$ consulte [45].

En un proceso $Linespec(L)$, uno puede predecir todas las muestras futuras sin ningún error, si se conoce un bloque de L muestras sucesivas. Una de las propiedades de un proceso $Linespec(L)$, es que si se le aplica la predicción lineal, el error de predicción sería cero $e_L = 0$. Esto significa que el proceso es completamente predecible. Se pueden extraer todas las amplitudes y frecuencias de un proceso $Linespec(L)$ aplicando la predicción lineal con un orden L. Esto implica que para extraer las líneas espectrales, se necesita conocer de antemano el número L de líneas espectrales o senoidales presentes en la señal.

Muchas de las señales que usamos para el análisis-síntesis de instrumentos musicales, podrían se consideradas como procesos $linespec(L)$ mezclados con ruido, por eso es importante comprender su comportamiento. Una consecuencia de esto, en el caso específico de tonos armónicos, es que el valor mínimo de $I(p)$ se encuentra aproximadamente en $p = L$. Este valor coincide con el periodo T de la forma de onda expresado en muestras. Veamos un ejemplo para ilustrar esta idea. Se graba un tono de corno de 440 Hz con una frecuencia de muestreo de $fs = 44100$ Hz. El periodo de la forma de onda tendrá un tamaño aproximado de $T = 100$ muestras. En un rango de $22,050$ Hz caben 50 armónicos para una frecuencia fundamental de $F_0 = 440$ Hz. Estos 50 armónicos equivalen a 100 valores en el dominio de la frecuencia. En otras palabras, el máximo de líneas espectrales para un tono de 440 Hz muestreado a 44100 Hz es de 100. Si asumimos que el tono de corno es un proceso de líneas espectrales de orden $L = 100$ combinado con ruido, entonces el orden de predicción «óptimo» sería $p = L = T = 100$. Esto no quiere decir que este sería el orden adecuado para todas las aplicaciones. Este orden es «óptimo» solamente si lo que se busca es aproximar la señal por medio de predicción lineal. Más adelante veremos que si se quiere extraer la envolvente temporal, se necesitará un valor menor que L, es decir $p < L = T$. Si no se

[8]Intuitivamente quiere decir que los valores de las amplitudes A_i y las fases θ_i no dependen los unos de los otros. En otras palabras, A_i no evoluciona en función de θ_i ni viceversa. Estas dos variables tampoco evolucionan en función de n. En estadística se dice que existe la misma probabilidad de observar el valor A_i en presencia o en ausencia del valor θ_i. Para que un proceso sea considerado $Linespec(L)$, los valores de A_i y θ_m deben ser estadísticamente independientes para todo i, m.

conoce de antemano el valor de L, se puede buscar el momento en que el error de predicción se estanca en cero $e_p = 0$ y se toma ese valor como el orden óptimo de predicción. Es importante mencionar que los resultados pueden variar dependiendo de la frecuencia de muestreo. Es posible que se deba a la inestabilidad numérica de la recursión de Levinson-Durbin. Para mejorar la inestabilidad numérica, se puede utilizar la descomposición de Cholesky [2] para resolver la ecuación 3.7, en lugar de la recursión de Levinson-Durbin. Otra solución es aumentar la precisión numérica a la hora del cálculo.

La figura 3.13 muestra una señal generada por medio de un proceso de líneas espectrales. La señal está formada por 10 senoidales con amplitudes aleatorias y frecuencias armónicas. La frecuencia fundamental es de 440 Hz. La primera gráfica muestra un ciclo de la señal muestreada, en el dominio del tiempo. La segunda gráfica muestra el espectro de potencia. Note que sólo se muestran 10 líneas espectrales. Las otras diez componentes son el complejo conjugado de las primeras. Al realizar la predicción lineal por medio de la recursión de Levinson, nos damos cuenta que la señal es completamente predecible a partir del orden 20. La tercera gráfica muestra el error de predicción en función del orden. Observe que el error se estanca en cero a partir del orden 20. Esta señal viene de un proceso de lineas espectrales de orden 20. Las amplitudes y las frecuencias se pueden extraer por medio del método de Pisarenko [45].

La figura 3.14 muestra el orden óptimo de predicción de un tono de saxofón de 440 Hz. El orden de predicción se calcula por medio del mínimo valor $I(p)$. El orden óptimo $p = 103$ está marcado con un pequeño círculo. La gráfica de arriba representa el espectro de potencia en decibeles. Este espectro está formado en su mayor parte por picos equidistantes. Hay aproximadamente 50 armónicos presentes en la señal, que equivale a 100 líneas espectrales. Los últimos armónicos se confunden con el ruido. Esta señal podría considerarse como un proceso Linespec mezclado con ruido. Por esta razón, la mejor aproximación se da alrededor del orden $p = 103$. La gráfica de en medio muestra la función de máxima verosimilitud $I(p)$. El valor mínimo, marcado con un círculo, muestra el orden óptimo del predictor. Si observamos la gráfica de abajo, que es la respuesta en frecuencia del filtro predictor, nos podemos dar cuenta que los coeficientes del filtro contienen la mayor parte de la información espectral del tono. Claramente se ve que este no es el orden adecuado para obtener la envolvente espectral del tono de saxofón.

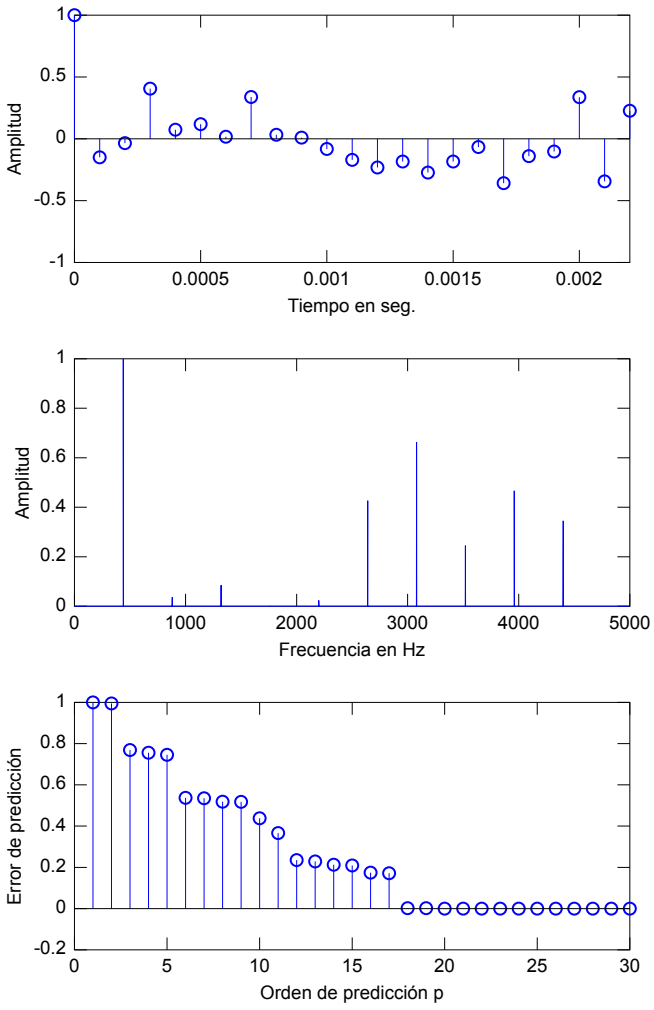


Figura 3.13: Predicción lineal de un proceso Linespec. Note que el error de predicción se estanca en cero cuando llega al orden 20. La razón es que la señal esta compuesta de 20 líneas espectrales. El orden óptimo de predicción en este caso es $L = 20$ y se dice que es un proceso Linespec(20).

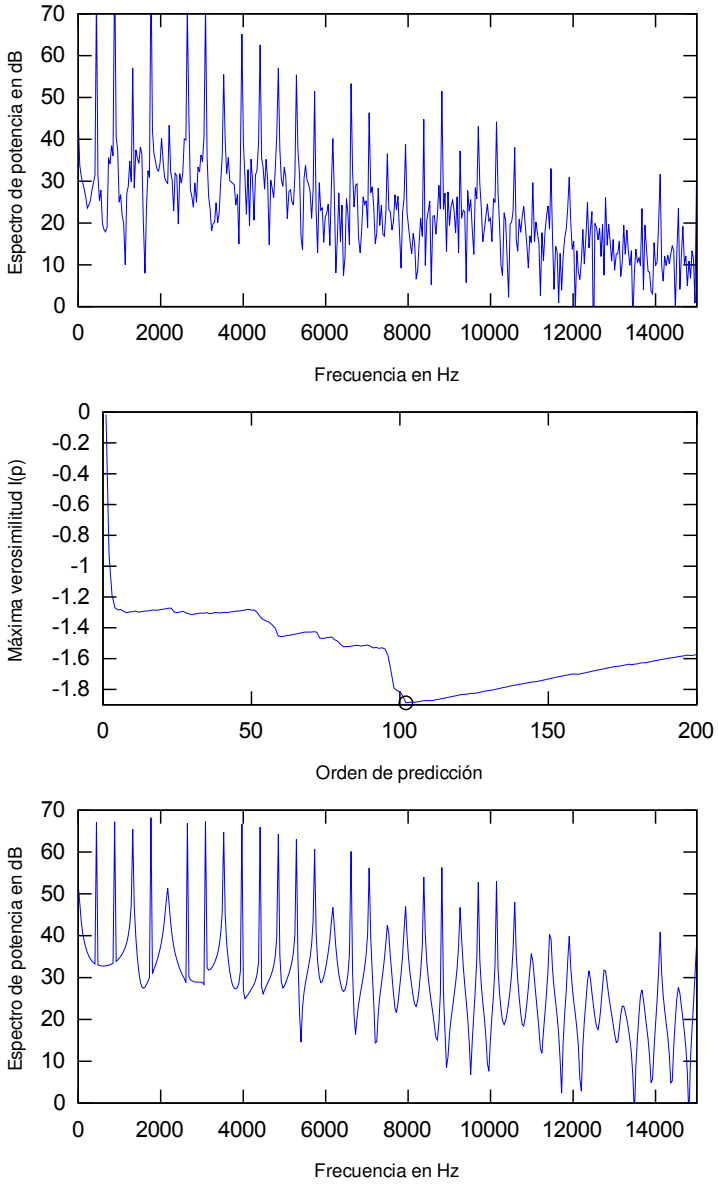


Figura 3.14: Obtención del orden óptimo de predicción de un tono de saxofón. La gráfica de arriba representa el espectro de potencia del tono de saxofón. La gráfica de en medio muestra el valor mínimo $I(p)$, marcado con un círculo. La gráfica de abajo es la respuesta en frecuencia del filtro predictor de orden $p = 103$.

Extracción de la envolvente espectral

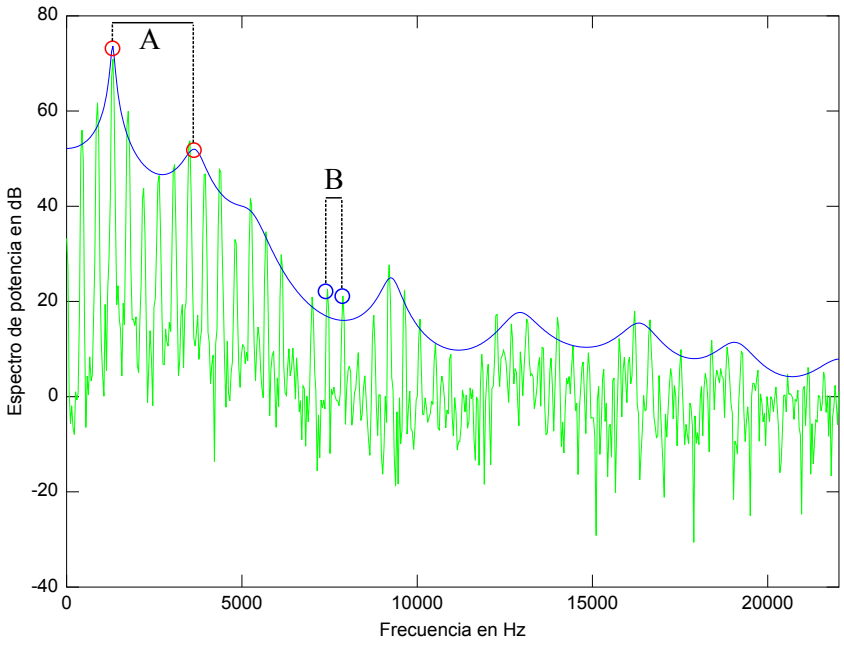


Figura 3.15: La envolvente espectral comparada con las líneas espectrales. La distancia A entre los máximos locales de la envolvente espectral, es más grande que la distancia B que hay entre las líneas espectrales de la señal.

Hemos visto que una consecuencia de minimizar el error cuadrático total, es que se obtiene una señal de error con espectro plano y un filtro que caracteriza la envolvente espectral. La envolvente se representa gráficamente como una curva que pasa por los picos del espectro. De acuerdo con esta representación, hay más picos en el espectro, que subidas y bajadas en la envolvente espectral. Esto es porque las distancias entre los picos del espectro, son más pequeñas que las distancias entre los máximos locales de la envolvente. Para ilustrar esta idea, se presenta la figura 3.15. Compare la distancia A, que hay entre los máximos locales (marcados con rojo) de la envolvente espectral, con la distancia B, que hay entre los picos (marcados con verde) del espectro. Es evidente que A es más grande que B.

Podemos establecer como un criterio intuitivo, que para extraer la envolvente espectral de un tono armónico, necesitamos captar fluctuaciones en el

espectro que sean más grandes que la frecuencia fundamental del tono, para que se cumpla que $A > B$. Esto significa, en la práctica, que el orden del predictor p debe ser menor al periodo en muestras T de la señal.

Podemos decir que la envolvente espectral corresponde a la estructura burda del espectro. Esta estructura gruesa corresponde de forma aproximada, a las formantes en el caso de la voz, o a las resonancias, en el caso de los instrumentos musicales. El problema es que cuando se utilizan órdenes de predicción muy altos, se empieza a captar la estructura fina del espectro. Observe la figura 3.16. En cada gráfica se muestra la envolvente espectral (en azul) obtenida por medio de Predicción Lineal. El orden de predicción está representado por la variable p. Note que mientras mayor es el orden de predicción, mayor es el número de crestas que contiene la envolvente espectral. En este caso, al usar un orden $p = 100$, el filtro captura la estructura armónica del tono, pero lo que buscamos es obtener la envolvente, como en el caso del orden $p = 40$. Es importante notar, que por medio de Predicción Lineal, los valles no se aproximan del todo bien. Esto se debe a que el filtro predictor es un filtro todos-polos y no tiene ceros. Un ejemplo de esto, es la falta de ajuste de la envolvente, alrededor de los armónicos 5 y 11 de la figura 3.15. El tono empleado en estos análisis, es un tono de corno de 440 Hz con una frecuencia de muestreo de 44100 Hz. El orden de $p = 100$ corresponde al periodo $T = 100$ del tono, por esta razón el filtro capturó su estructura armónica.

Para establecer manualmente el orden "óptimo" para extraer la envolvente espectral, presentamos el siguiente procedimiento:

1. Calcule la frecuencia fundamental del tono, utilizando el algoritmo de Boersma o algún otro método de determinación de altura [16].

2. Escoja un segmento de señal de aproximadamente 4 a 6 ciclos de forma de onda.

3. Grafique el espectro de potencia.

4. Visualmente, determine la distancia promedio en Hz entre los máximos locales de la envolvente. Por ejemplo, la distancia A de la figura 3.15.

5. Determine el orden óptimo de predicción por medio de la siguiente expresión $p \approx fs/A$, donde fs es la frecuencia de muestreo y A la distancia en Hz entre los picos de la envolvente.

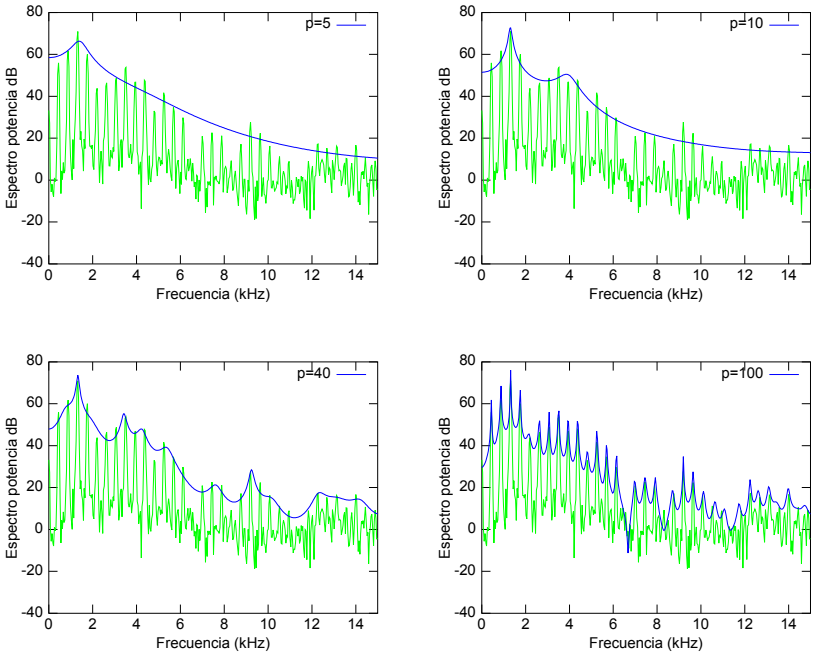


Figura 3.16: Envolvente temporal con diferentes órdenes de predicción.

El procedimiento anterior, aunque no es muy preciso, nos da una primera aproximación que podemos ir mejorando por medio de ensayo y error.

Enseguida presentamos un método automático de determinación del orden óptimo de predicción para extraer la envolvente temporal de un tono armónico. Este método se basa en la curva de máxima verosimilitud $I(p)$ propuesta por Akaike. Es importante mencionar que este método no siempre es exitoso, pero es el método por omisión en nuestra implementación de Pd. Como mencionamos al principio, en caso de fallo, se pueden modificar los parámetros de forma manual.

1. Calcule la frecuencia fundamental del tono, utilizando el algoritmo de Boersma o algún otro método de determinación de altura [16].

2. Escoja un segmento de señal de aproximadamente 4 a 6 ciclos de forma de onda.

3. Aplique la recursión de Levinson-Durbin para obtener el mínimo error cuadrático normalizado $V(p)$ para cada iteración p, hasta el orden $p = T$ que corresponde al periodo del tono, expresado en muestras.

4. Utilizando la ecuación 3.22, obtenga la curva de máxima verosimilitud $I(p)$.

5. El orden óptimo de predicción sería el menor valor de la curva $I(p)$ en el intervalo $p_a = T/15 < p < p_b = T/2$. Los valores p_a y p_b se establecen típicamente en $T/15$ y $T/2$ respectivamente, pero se pueden ajustar estos parámetros de acuerdo con las características de una señal en particular.

Debido a que el cálculo de la máxima verosimilitud 3.22 es muy sensible al término lineal, se puede substituir la curva $I(p)$ por la primera diferencia del logaritmo del error normalizado $\log(V(p))$. El algoritmo es como sigue:

1. Calcule la frecuencia fundamental del tono, utilizando el algoritmo de Boersma o algún otro método de determinación de altura [16].

2. Escoja un segmento de señal de aproximadamente 4 a 6 ciclos de forma de onda.

3. Aplique la recursión de Levinson-Durbin para obtener el mínimo error cuadrático normalizado V_p para cada iteración p, hasta el orden $p = T$ que corresponde al periodo del tono, expresado en muestras.

4. Calcule la primera diferencia del logaritmo del error cuadrático normalizado $dE(p) = DIFF(\log(V(p)))$

5. La primera aproximación al orden óptimo de predicción, sería el mínimo valor de la curva $dE(p)$ en el intervalo $p_a = T/15 < p < p_b = T/2$. Los valores p_a y p_b se establecen típicamente en $T/15$ y $T/2$ respectivamente, pero se pueden ajustar estos parámetros de acuerdo con las características de una señal en particular.

6. El orden del máximo local más cercano hacia adelante se tomará como el valor del orden óptimo de predicción.

Estos dos métodos para encontrar el orden óptimo de predicción se ilustran en la figura 3.17. En este ejemplo, se analiza un tono de saxofón de 440 Hz. Usando el método de Akaike [2] se obtiene un orden $p = 30$. Cuando se usa el método de primeras diferencias, se obtiene un orden $p = 25$. Si comparamos los espectros, la diferencia entre ellos no es muy grande. En este caso, conviene escoger el orden más pequeño para ahorrar espacio de almacenamiento y mejorar la estabilidad numérica.

El listado C.9 en Octave, es un ejemplo de como se implementan los algoritmos anteriores para extraer la envolvente temporal.

3.3.5. Formato de almacenamiento

En la sección 2.3.5 se dio una breve explicación del formato RIFF. Para guardar los parámetros de la predicción lineal, se han diseñado dos archivos con formato RIFF. El primero tiene un identificador LATT y sirve para guardar los parámetros de un filtro estático; su estructura se muestra en la tabla 3.1. El segundo tiene un identificador LPC y contiene los parámetros de un filtro dinámico. La tabla 3.2 muestra la estructura del formato LPC. En seguida se explican los diferentes bloques de estos formatos:

head Este bloque es el encabezado y contiene información básica sobre los filtros LPC como el orden del predictor, número de filtros, tamaño de la ventana, resolución, etc. Este bloque es obligatorio.

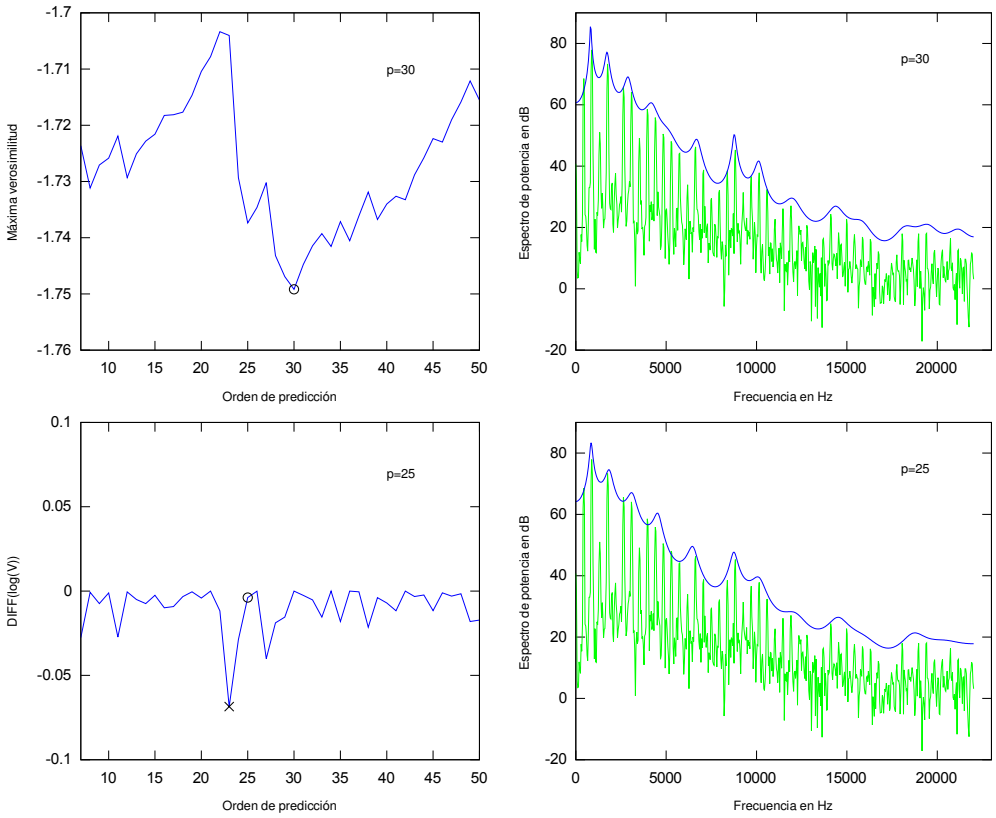


Figura 3.17: Orden óptimo de predicción para la extracción de la envolvente temporal. En las gráficas de arriba, el orden está calculado por medio del método de Akaike. En las gráficas de abajo, el cálculo se basa en la primera diferencia del logaritmo del error cuadrático.

loop Contiene información sobre el segmento de reproducción cíclica. Este bloque es opcional. La ausencia de este bloque le indica al reproductor que el sonido es impulsivo y carece de segmento de reproducción cíclica.

gain Este bloque contiene la información sobre el factor de ganancia[9]. Este bloque es opcional. La ausencia de este bloque le indica al reproductor que la señal de entrada no sería normalizada.

refc Este bloque contiene los valores de los coeficientes de reflexión en diferentes instantes de tiempo. Este bloque es obligatorio.

Tipo	Bytes	Contenido
char[4]	4	Identificador "FORM" o "RIFF" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "RIFF" o "FORM"
char[4]	4	Identificador "LATT" en formato ASCII
char[4]	4	Identificador "head" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "head"
unsigned int	4	Orden del predictor
unsigned short	2	Resolución de los coeficientes de reflexión en bits (16 bits por omisión)*
char[4]	4	Identificador "refc" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "refc"
signed short	2*	Coeficiente de reflexión 0
signed short	2*	Coeficiente de reflexión 1
signed short	2*	Coeficiente de reflexión 2
signed short	2*	Coeficiente de reflexión 3
...	...	...

Tabla 3.1: Estructura del archivo de almacenamiento LATT

3.4. Síntesis

La figuras 3.18 muestra la fase de síntesis. Una señal de excitación u_n es pasada por un filtro lattice. Los coeficientes de reflexión k_n y los valores de ganancia s_{rms}, que están guardados en el archivo, se van actualizando en distintos instantes de tiempo. Esto está representado por un reloj que controla la salida de los parámetros. La señal de salida del filtro lattice $\hat{s}_n$, se almacena en un arreglo temporal para calcular el valor RMS o la amplitud

[9]Está pendiente substituir este bloque por dos bloques: *rms* y *amp*.

Tipo	Bytes	Contenido
char[4]	4	Identificador "FORM" o "RIFF" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "RIFF" o "FORM"
char[4]	4	Identificador "LPC " en formato ASCII
char[4]	4	Identificador "head" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "head"
unsigned int	4	Orden del predictor
unsigned int	4	Número de filtros
unsigned int	4	Distancia entre filtros, en muestras(stepSize)
unsigned int	4	Tamaño de la ventana de análisis (winSize)
unsigned short	2	Resolución de los factores de ganancia (16 bits por omisión)*
unsigned short	2	Resolución de los coeficientes de reflexión en bits (16 bits por omisión)*
char[4]	4	Identificador "loop" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "loop"
unsigned int	4	Inicio del segmento de reproducción cíclica (loopA en muestras)
unsigned int	4	Final del segmento de reproducción cíclica (loopB en muestras)
char[4]	4	Identificador "gain" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "gain"
signed short	2*	Factor de ganancia al tiempo 0
signed short	2*	Factor de ganancia al tiempo 1
signed short	2*	Factor de ganancia al tiempo 2
signed short	2*	Factor de ganancia al tiempo 3
...	...	...
char[4]	4	Identificador "refc" en formato ASCII
unsigned int	4	Tamaño del segmento de datos del bloque "refc"
signed short	2*	Coeficiente de reflexión 0 del filtro 0
signed short	2*	Coeficiente de reflexión 1 del filtro 0
signed short	2*	Coeficiente de reflexión 2 del filtro 0
signed short	2*	Coeficiente de reflexión 3 del filtro 0
...	...	...
signed short	2*	Coeficiente de reflexión 0 del filtro 1
signed short	2*	Coeficiente de reflexión 1 del filtro 1
signed short	2*	Coeficiente de reflexión 2 del filtro 1
signed short	2*	Coeficiente de reflexión 3 del filtro 1
...	...	...

Tabla 3.2: Estructura del archivo de almacenamiento LPC

máxima. Este valor $\hat{s}_{rms}$ se compara con los valores guardados en el archivo s_{rms} para obtener el factor de ganancia G, con el que se normaliza la señal. El resultado final es la señal $\tilde{s}_n$. Todo este proceso se realiza en tiempo real. Como se puede ver, la fase de síntesis tiene un esquema sencillo. En las siguiente sección se explica la implementación del filtro *lattice*.

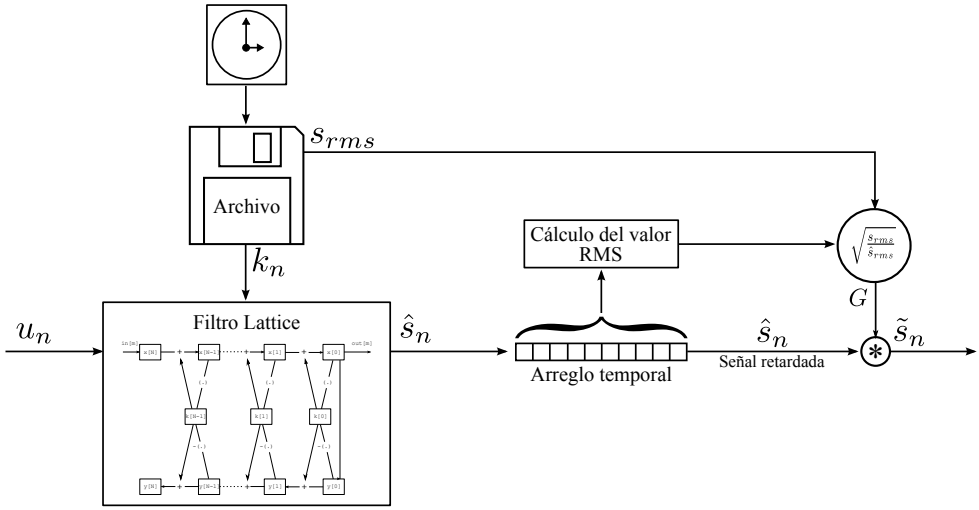


Figura 3.18: Diagrama de flujo de la síntesis por medio de predicción lineal.

3.4.1. Filtros lattice

Las estructuras *lattice* tienen una gran importancia en el procesamiento de señales. Un filtro LPC puede ser representado por una estructura *lattice*. Estas estructuras se derivan de la recursión de Levinson [45]. Se les llama *lattice* porque se representan gráficamente como una malla o enrejado de líneas cruzadas. Las estructuras lattice presentan varias ventajas:

- Se puede obtener el filtrado para varios ordenes más bajos, al mismo tiempo y sin modificar los coeficientes de reflexión k_n. Esto se logra tomado la muestra de salida, en diferentes secciones de la estructura *lattice*.

- El filtro *lattice* es muy fácil de implementar en el dominio del tiempo, funciona en tiempo real y la velocidad de cálculo es relativamente rápida para órdenes bajos.

- Finalmente, el filtro recursivo representado por la estructura lattice permanecerá estable, siempre y cuando los coeficientes de reflexión k_n estén cuantizados de manera que $|k_n| < 1$.

En este trabajo, no elaboraremos más en la teoría de estas estructuras, sólo daremos un ejemplo de su implementación. Para mayor información consulte [65] y [45].

La figura 3.19 muestra un esquema de la estructura *lattice*. Los rectángulos con los nombres de variables, representan las celdas de memoria. Para la implementación, se necesitan tres arreglos: x[i], k[i] y y[i]. El orden del filtro está representado por la variable N. Los arreglos x[i] y y[i] deben tener un tamaño de N+1 elementos. El arreglo k[i] tiene un tamaño de N elementos. Por cada muestra de entrada in[m]=x[N] se genera una muestra de salida out[m]=x[0]. La implementación final en pseudocódigo se presenta en el listado C.10.

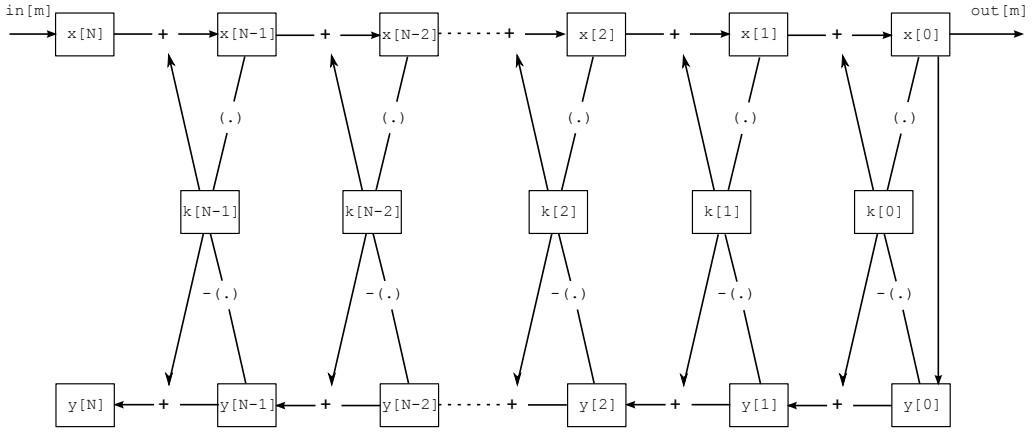


Figura 3.19: Filtro lattice.

Capítulo 4

Modelo de síntesis híbrido

La contribución original de este trabajo consiste en un modelo de síntesis híbrido que combina la síntesis por segmentación de onda y la predicción lineal. Como se planteó en los capítulos 2 y 3, ambos métodos de síntesis pueden ser beneficiados si los combinamos. Por un lado, en la síntesis por segmentación de onda, se pueden utilizar los filtros obtenidos por medio de predicción lineal para eliminar las distorsiones producidas por las funciones interpoladoras. En la predicción lineal, por otra parte, se puede hacer uso de la segmentación de onda como una señal de excitación, ampliando así su aplicación a la síntesis de instrumentos musicales.

4.1. Filtrando armónicos no controlables, por medio de predicción lineal

Utilizaremos como ejemplo un tono de flauta de 440 Hz. Tomaremos seis ciclos de forma de onda para realizar el análisis y tomaremos en cuenta los 10 primeros armónicos. Vamos a comparar el espectro de la síntesis de Mitsuhashi, por sí sola, con su versión filtrada con predicción lineal. En la figura 4.1, se muestra el espectro de la síntesis de Mitsuhashi en color verde y el espectro de la señal original en azul. Note que los primeros 10 armónicos coinciden; pero después, los armónicos de la síntesis de Mitsuhashi sobrepasan, en general, el nivel del resto de los armónicos de la señal original. Estos son producidos por la función interpoladora, que en este caso, es lineal. Observe ahora la figura 4.2. En ella se muestra el espectro de la síntesis de Mitsuhashi, pero esta vez, está filtrada por medio de un filtro obtenido con

predicción lineal. Nótese que en este caso los armónicos mas allá del décimo van descendiendo incluso por debajo del umbral de audición (representado por una curva roja). Aunque este método de síntesis híbrido tampoco es de banda limitada, como en el caso de la síntesis aditiva, el resultado auditivo es muy similar[1]. Una ventaja de este método, es que el orden del filtro no necesita ser muy grande. En este ejemplo se utilizó un orden $p = 4$. Un filtro lattice de orden bajo es muy rápido y no compromete el desempeño de la segmentación de forma de onda, en cuanto al tiempo de cómputo. Basta con escoger un orden tal, que su respuesta en frecuencia coincida, de manera burda, con la envolvente espectral del tono original, y que sea decreciente en la parte aguda del espectro. Evidentemente, si se utilizan ordenes bajos, la señal de error no tendrá un espectro plano, pero esto no nos afecta en este caso.

4.2. Usando la segmentación de forma de onda como señal de excitación

La predicción lineal da buenos resultados en la síntesis de la voz humana. Esto se debe a que el modelo LPC se ajusta muy bien a las características del tracto vocal. Esto no sucede con algunos instrumentos musicales cuya envolvente espectral tiene valles muy pronunciados. En este caso, el filtro todos polos, no puede aproximar muy bien los valles. Por esta razón, el espectro de la señal de error, extraído con la predicción lineal, no es completamente plano. Por consiguiente, al substituir la señal de error, con un tren de impulsos, (como se hace tradicionalmente), no se logran los resultados deseados. La opción es utilizar otras señales de excitación como substitutos de la señal de error, a la entrada del filtro predictor. En este trabajo, proponemos utilizar la síntesis por segmentación de forma de onda, como señal de excitación. La idea es aproximar la señal de error, por medio de la síntesis de Mitsuhashi, para luego pasarla por el filtro predictor. La señal de salida, sería nuestra aproximación a la señal original.

Para ilustrar esta idea, tomaremos un segmento de un tono de violín de 440 Hz. El segmento de análisis contiene 6 ciclos de forma de onda. La figura

[1]En teoría, porque cuando utilizamos filtros dinámicos, a veces se producen discontinuidades o inestabilidades en el filtro que pueden producir ruido. Este ruido, según nuestras pruebas subjetivas de audición, no es del agrado del oyente.

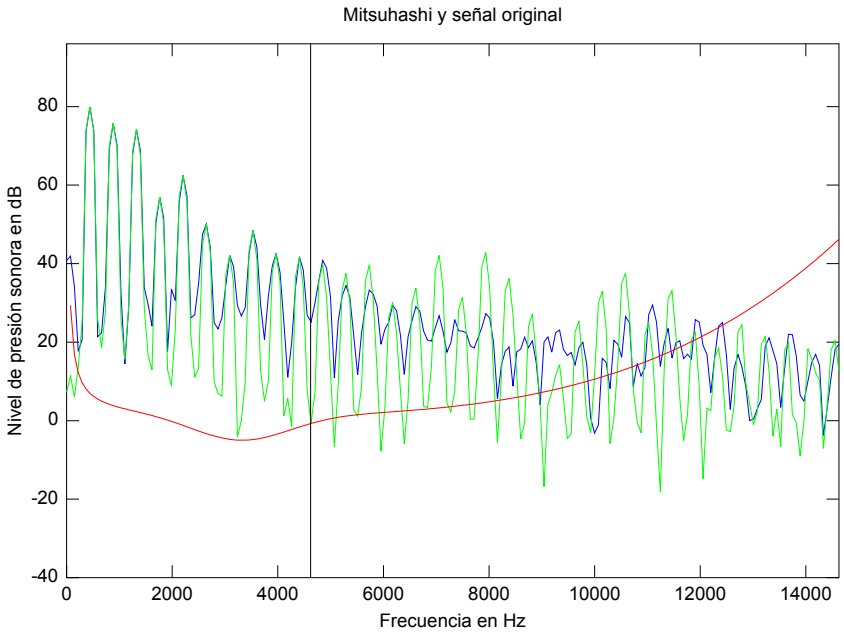


Figura 4.1: Tono de flauta de $440Hz$. Comparación de los espectros de la señal original (azul) y de su correspondiente síntesis de Mitsuhashi (verde) con 10 armónicos. La curva roja representa el umbral de audición. La línea vertical negra marca el lugar donde está el décimo armónico.

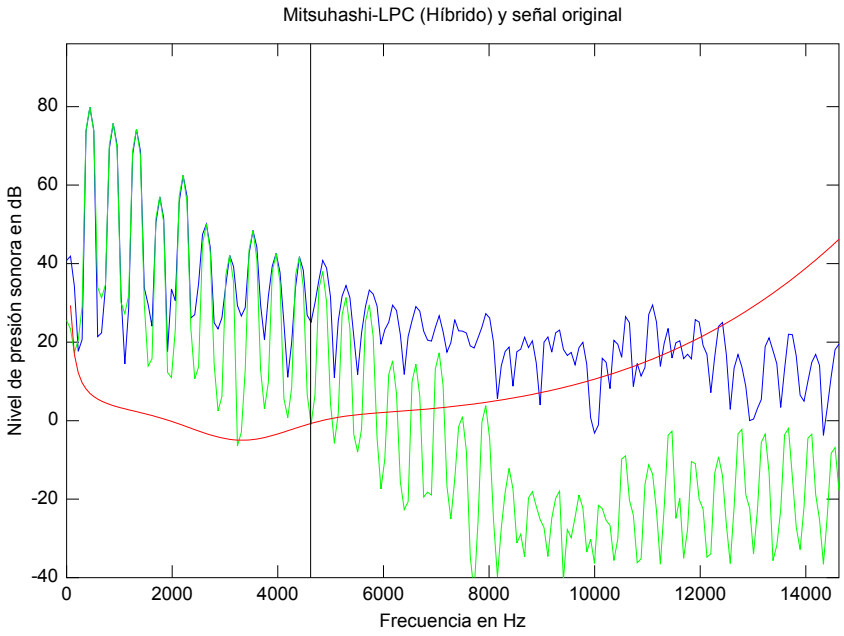


Figura 4.2: Tono de flauta de $440Hz$. Comparación de los espectros de la señal original (azul) y de su correspondiente síntesis Mitsuhashi (verde) con 10 armónicos, pero filtrado con LPC. La curva roja representa el umbral de audición. La línea vertical negra marca el lugar donde está el décimo armónico.

4.3 muestra el espectro del tono de violín en color azul. La curva de color verde, representa la envolvente espectral obtenida con un predictor de orden $p = 7$. El orden de predicción se calculó de acuerdo a los procedimientos detallados en la sección 3.3.4. Note que la envolvente espectral no pasa por el armónico 6. En otras palabras, la forma de la envolvente espectral es demasiado burda. Si quisiéramos ajustar mejor la curva de la envolvente, tendríamos que escoger un orden mas grande. Sin embargo, órdenes mas grandes comenzarían a captar la estructura fina del espectro (armónicos). Para reconstruir la señal, utilizamos un tren de impulsos de banda limitada, que incluye los primeros 20 armónicos. Este tren de impulsos se muestra en la figura 4.4. Al filtrar el tren de impulsos se obtiene una aproximación a la señal original. La figura 4.5 muestra la aproximación de la señal por medio de LPC en color verde. La curva de color azul representa el espectro de la señal original. La curva de color rojo muestra el umbral de audición, y la línea vertical en color negro, marca el lugar del armónico 20. Como puede observar, algunos armónicos no se ajustan muy bien, como el armónico 6 que discutimos en la figura 4.3. Ahora en lugar de utilizar un tren de impulsos, vamos a utilizar la segmentación de forma de onda como señal de excitación. En la figura 4.6 se muestra el resultado. En color azul se muestra el espectro de la señal original. En color verde se muestra la reconstrucción por medio del modelo híbrido. Como en la figura anterior, la curva roja representa el umbral de audición y la línea vertical de color negro marca el lugar del veinteavo armónico. Note que los primeros 20 armónicos se ajustan muy bien. Los demás, constituyen armónicos no controlables que están, en su mayor parte, por debajo del umbral de audición.

4.3. Tipos de modelos híbridos

En los ejemplos anteriores, se realizó la síntesis de un segmento de audio solamente. Hay que considerar que para analizar y sintetizar una señal de audio, es necesario hacerlo en pequeños segmentos de tiempo, normalmente traslapados. Esto implica que los parámetros de la síntesis van cambiando en función del tiempo. Utilizaremos el término *dinámico*, cuando nos refiramos a los parámetros de un modelo de síntesis que van cambiando de un segmento a otro. Utilizaremos el término *estático*, cuando dichos parámetros permanezcan constantes a lo largo de la duración de la señal en su totalidad; es decir, que los parámetros no cambian de un segmento a otro. Por ejemplo,

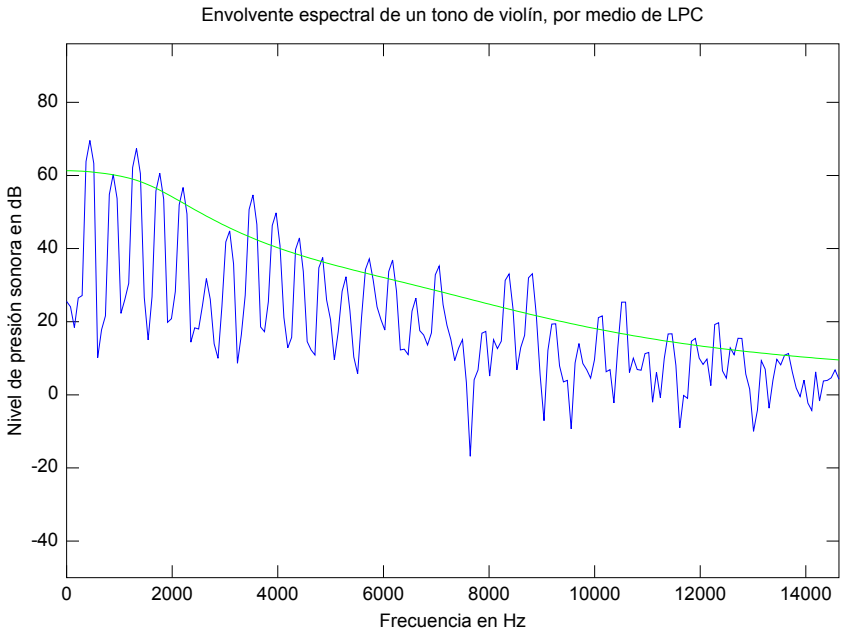


Figura 4.3: Envolvente espectral de un tono de violín, obtenido por medio de predicción lineal. La curva azul representa el espectro del tono de violín. La curva en color verde representa la envolvente espectral calculada por medio de LPC con predictor de orden $p = 7$.

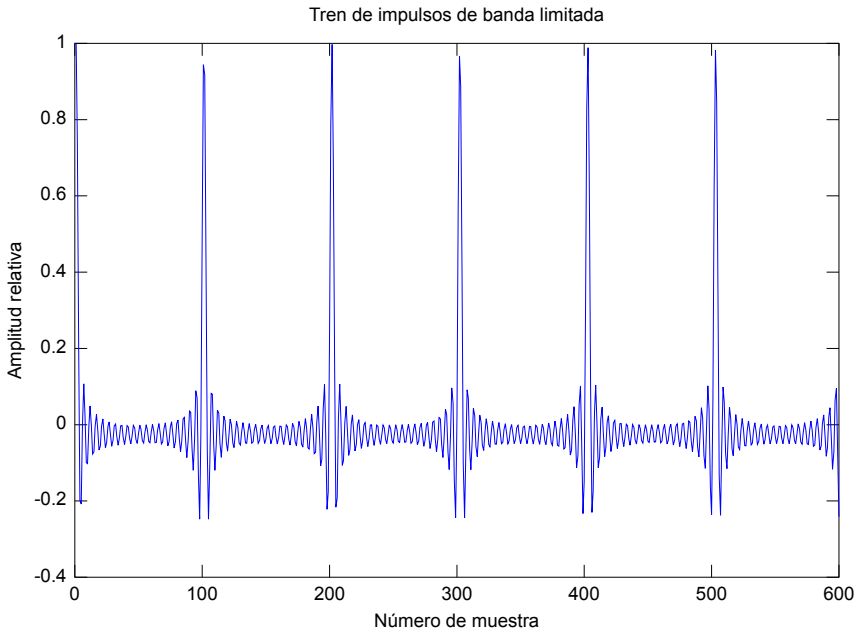


Figura 4.4: Tren de impulsos de banda limitada con 20 armónicos.

un modelo *LPC estático* se refiere a un solo filtro que será aplicado a toda la señal. Por el contrario, un modelo *LPC dinámico* se refiere a un filtro (o varios filtros) cuyos coeficientes cambian de un segmento a otro. Los mismos términos aplican para la síntesis de Mitsuhashi. De acuerdo con estas ideas, podemos agrupar los modelos híbridos en cuatro categorías:

Mitsuhashi-estático, LPC-estático En este modelo se obtiene solamente una forma de onda, y un filtro para toda la señal de audio. En este caso, se tendría que agregar una envolvente temporal (por ejemplo ADRS) para aproximar la señal de audio original.

Mitsuhashi-dinámico, LPC-estático Este modelo consiste en obtener un sólo filtro predictor para toda la señal. La señal de error se aproxima por medio de la síntesis de Mitsuhashi. Los nodos de la forma de onda cambian en función del tiempo.

Mitsuhashi-estático, LPC-dinámico En este caso, el carácter dinámico de la señal, se logra variando los parámetros del filtro LPC. La señal de excitación por medio de la síntesis de Mitsuhashi permanece constante,

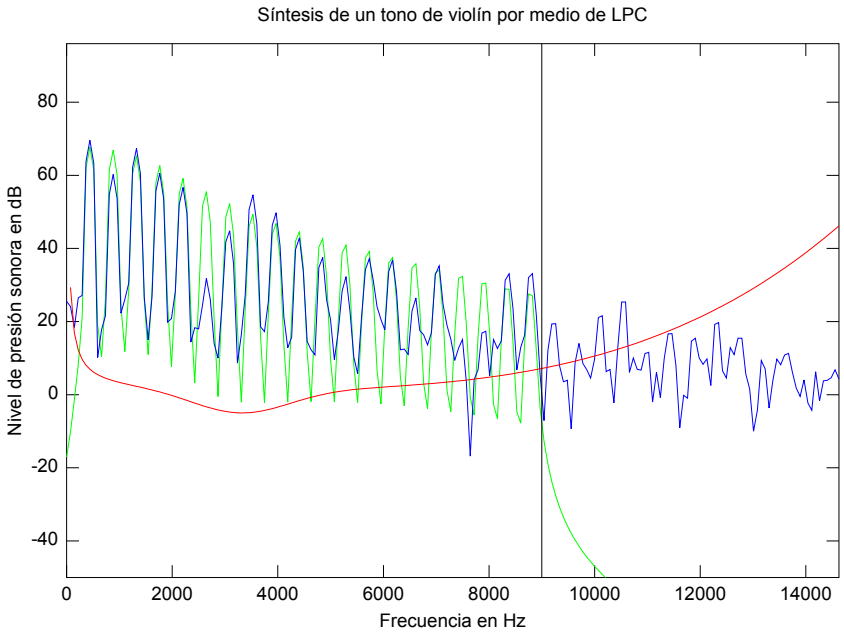


Figura 4.5: Síntesis de un tono de violín por medio de LPC. La curva en color azul representa el espectro de la señal original. La curva en color verde es la aproximación por medio de predicción lineal, utilizando un tren de impulsos de banda limitada. El tren de impulsos contiene 20 armónicos. La curva roja muestra el umbral de audición. Finalmente, la línea vertical en color negro, marca el lugar del armónico número 20.

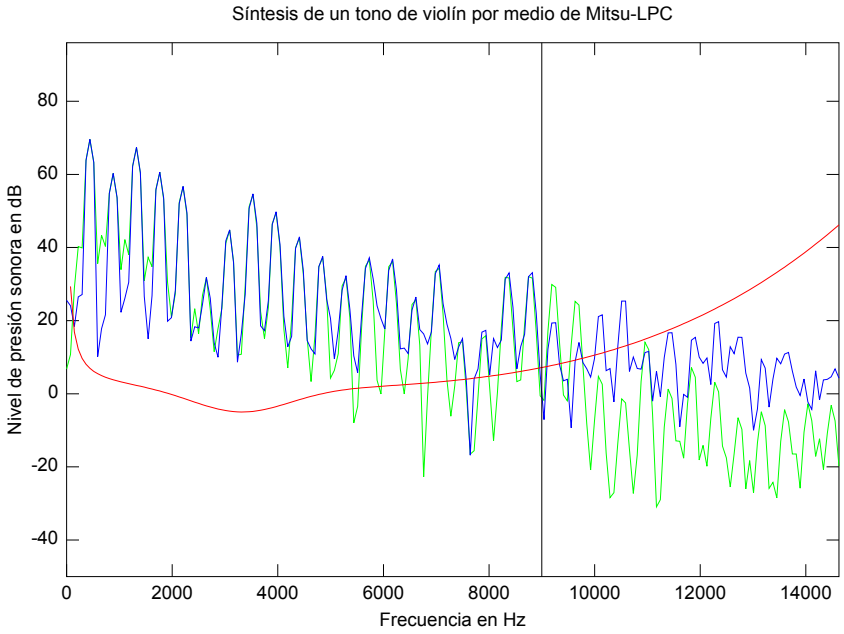


Figura 4.6: Síntesis de un tono de violín por medio del modelo híbrido Mitsu-LPC. La curva en color azul representa el espectro de la señal original. La curva en color verde representa la señal sintetizada con Mitsu-LPC. La curva roja muestra el umbral de audición. Finalmente, la línea vertical en color negro, marca el lugar del armónico número 20.

mientras que los filtros modificaran el espectro de la señal en cada segmento.

Mitsuhashi-dinámico, LPC-dinámico Este modelo consiste en que tanto los parámetros de la síntesis por segmentación de onda, como los parámetros del filtro LPC, cambian con respecto al tiempo. Aunque, podría pensarse que este caso puede aproximar mejor la señal original, es más susceptible a introducir discontinuidades e inestabilidades. Además, se tendrían que controlar un número mayor de parámetros.

Por ahora, recomiendo la utilización del segundo y tercer modelos. Es decir, Mitsuhashi-dinámico con LPC-estático o Mitsuhashi-estático con LPC-dinámico. La razón es que si utilizamos los dos modelos estáticos, el sonido no es muy realista. Por otro lado, cuando utilizamos los dos modelos dinámicos aumentan las probabilidades de producir discontinuidades e inestabilidades en la señal resultante. Además, es necesario controlar un mayor número de parámetros. Queda pendiente, para un trabajo posterior, un estudio a fondo de las características y posibilidades prácticas de cada uno de estos modelos.

Capítulo 5

Implementación en Pure Data

Se ha presentado detalladamente el funcionamiento de la síntesis por segmentación de forma de onda, la predicción lineal, y el modelo de síntesis híbrido Mitsu-LPC. Estos algoritmos de síntesis sonora han sido implementados en Pd. En este capítulo, damos una breve descripción de los objetos creados para este entorno de programación y su funcionamiento.

Pd (Pure Data) [3] es un entorno de programación gráfico para desarrollar aplicaciones de audio y vídeo. Pd pertenece a la misma familia de lenguajes de programación que Max [66], basados en conexiones o *patches*. Max fue originalmente desarrollado por Miller Puckette en el IRCAM. Pd ha sido creado con la idea de refinar Max y extenderlo a otras aplicaciones, además de audio y MIDI, como son las aplicaciones de gráficos y vídeo. La parte central de Pd es desarrollado y mantenido por Miller Puckette, pero incluye el trabajo de muchos otros programadores. Esto hace del paquete entero de Pd, un verdadero esfuerzo comunitario. Se ha escogido este entorno de programación porque tanto Pd como Max son muy utilizados en el ámbito de la computación musical. Se prefirió Pd sobre Max, porque tiene una licencia libre y funciona en diversas plataformas (Windows, MacOS y Linux). Como el código fuente de Max y Pd es muy similar, se pueden portar los objetos a Max con ligeras modificaciones.

Los algoritmos de síntesis de este trabajo han sido implementados en forma de objetos que pueden ser utilizados en Pd. Estos objetos han sido desarrollados en lenguaje C [67] y utilizando las bibliotecas *FFTW* [68] para análisis de Fourier y *libsndfile* [69] para lectura y escritura de archivos de audio.

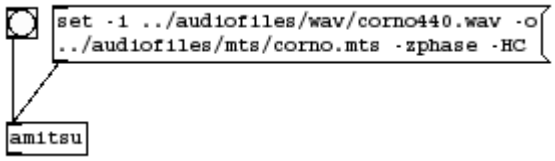
Antes de continuar, se asume que el lector está familiarizado con Ma-

x/MSP de Cycling o Pd de Miller Puckette. Si necesita mayor información acerca de la utilización de Pd, consulte el libro de Andy Farnell «Designing Sound» [70] o el excelente documento de Miller Puckette, «Theory and Techniques of Electronic Music» [10]. También puede consultar el manual de Max/MSP, ya que ambos programas funcionan de manera muy similar. Por otro lado, siendo que en esta tesis no se pretende incluir un manual de cómo desarrollar objetos externos para Pd, no presentaremos los detalles de la interfaz de programación; pero puede consultar el código fuente que viene en el CD que acompaña a este trabajo. Para compilar e instalar estos objetos, refiérase al archivo `INSTALL.txt`. Si está interesado en desarrollar objetos para Pd, el tutorial de Johannes M. Zmölnig «HOWTO Write an External for Pd» [71], es una buena opción; sin embargo, tendrá que estudiar los archivos fuente de Pd para poder desarrollar objetos más complejos. La interfaz de programación de Max/MSP es muy similar a la de Pd, y se tiene la ventaja de que Cycling proporciona de manera gratuita sus herramientas de programación y amplia documentación [66]. Otra herramienta para desarrollar objetos externos en Max y Pd, es la interfaz de programación Flext [72]. La ventaja de Flext es su portabilidad, ya que se utiliza el mismo código para crear objetos para Max y Pd en diferentes plataformas; pero es importante considerar que tiene una licencia pública GNU.

5.1. Descripción de los objetos

amitsu El objeto `amitsu` analiza una señal para obtener los parámetros de la síntesis por segmentación de forma de onda, utilizando el modelo de Mitsuhashi. Este objeto calcula las ordenadas de los nodos y guarda la información en un archivo con formato MITS. Se pueden pasar varios parámetros a este objeto, pero es un requisito, que el nombre del archivo sea pasado como el primer argumento, a menos que se use la opción `-i`. El análisis no se realizará hasta que un mensaje *bang*, sea recibido por el objeto `amitsu`. El objeto envía un mensaje *bang* a su outlet cuando el análisis ha terminado.

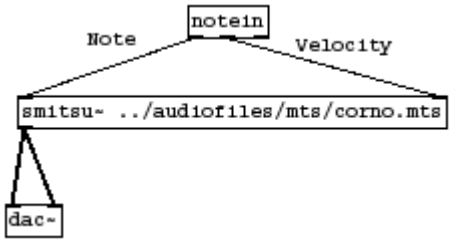
Los parámetros también pueden ser pasados al objeto `amitsu` enviando un mensaje con el selector `set`.



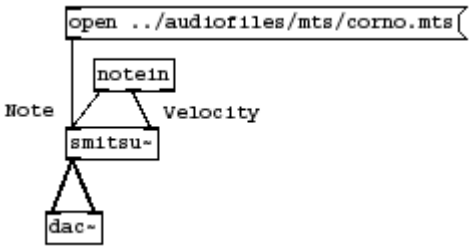
Enseguida se muestran las opciones disponibles:

```
-beg <muestra>       Principio de la señal
-end <muestra>       Final de la señal
-dBlim <límite>      Limite en dB para reducción de información
-PC, PL, HC          Función de interpolación: constante, lineal,
                     medio-coseno
-zphase              Ignorar la información de fase
-loopA <muestra>     Inicio del segmento de reproducción cíclica
-loopB <muestra>     Final del segmento de reproducción cíclica
-autoloop            El programa escoge los puntos de
                     reproducción cíclica
-noLoop              El programa no establece puntos de
                     reproducción cíclica
-flatLoop            Empareja la envolvente temporal del
                     segmento de reproducción cíclica
-nPoints <entero>    Número de nodos en la forma de onda
-o <nombreArchivo>   Nombre del archivo de salida (.mts)
-i <nombreArchivo>   Nombre del archivo que se va a analizar (.wav)
-f0 <númeroDecimal>  Establece la frecuencia fundamental. Si no se
                     especifica, f0 será calculada con análisis
                     cepstral
```

El objeto `smitsu~` carga el archivo MITS que fue previamente obtenido con el objeto `amitsu`. Se puede incluir el nombre del archivo como argumento o enviar un mensaje con el selector `open` o `set`. Una vez que el archivo es cargado, el objeto está listo para comenzar la síntesis. El primer *inlet* recibe el número de nota MIDI. El segundo, recibe el valor de velocidad.



Se pueden obtener los mismos resultados con el siguiente *patch*. Se necesita enviar el mensaje `open` antes de comenzar el flujo de audio.



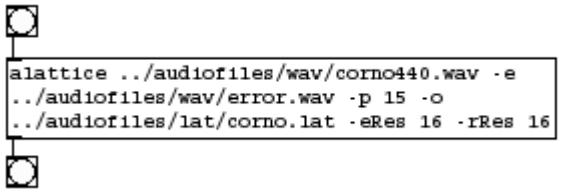
Los parámetros disponibles son:

```
-beg <muestra>      Principio de la señal
-end <muestra>      Final de la señal
-p <orden>          Orden del filtro predictor
-i <nobreArchivo>   Archivo de audio que será analizado
-o <nobreArchivo>   Archivo de salida en formato LATT
-e <nobreArchivo>   Señal de error en formato WAV
-eRes <resolución>  Resolución de la señal de error
                    (16 o 32 bits)
-rRes <resolución>  Resolución de los coeficientes de
                    reflexión
```

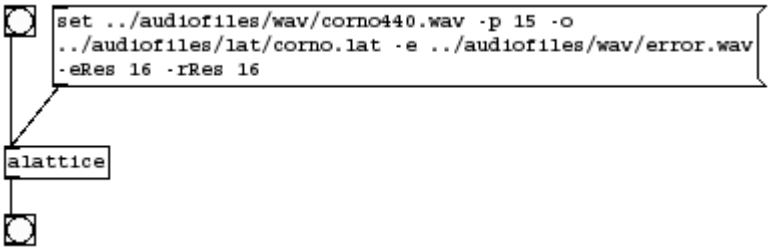
`alattice` Realiza el análisis para obtener un filtro *lattice* estático basado en Predicción Lineal. Este objeto calcula los coeficientes de reflexión utilizando la recursión de Levinson-Durbin. Los coeficientes son guardados en un archivo LATT. Estos datos serán usados, posteriormente, por el objeto `slatice~`, que es un filtro todos polos con una estructura *lattice*.

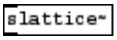
El primer argumento del objeto `alattice` es el nombre del archivo de la señal de audio que se quiere analizar. Los demás parámetros son opcionales. El análisis no se realizará hasta que un mensaje `bang` sea

recibido por el objeto `alattice`. Un mensaje `bang` será enviado por el outlet izquierdo, cuando haya concluido el análisis.

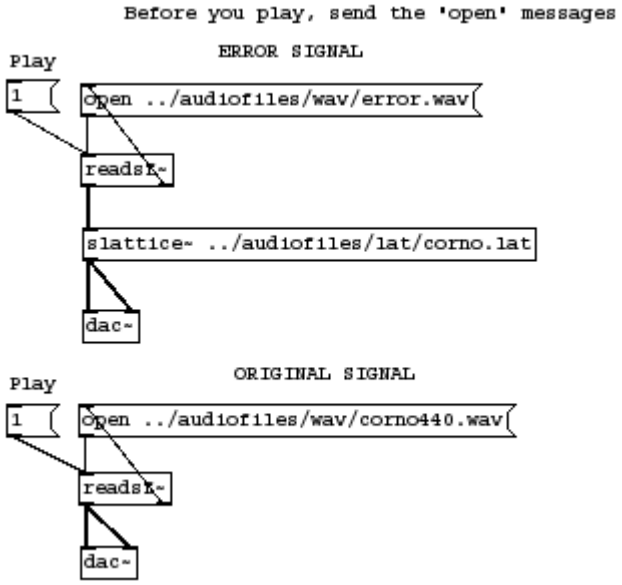


Los parámetros también pueden ser pasados al objeto enviando un mensaje con selector `set`. El nombre del archivo de audio puede ser especificado con la opción `-i` o el selector `open`.



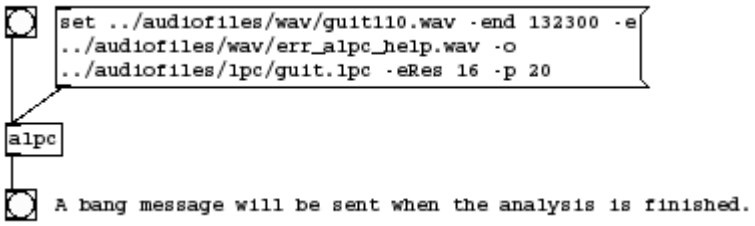


Filtra una señal de audio por medio de un filtro lattice estático basado en Predicción Lineal. Este filtro *lattice* usa los coeficientes de reflexión previamente guardados en un archivo LATT, y que fueron obtenidos por `alattice`. El primer argumento de `slatice~` debe ser el nombre del archivo LATT. Otra opción es utilizar la opción `-i` seguida del nombre del archivo. El resto de los parámetros es opcional. Si se filtra la señal de error obtenida por `alattice`, la salida será idéntica a la señal original.



`alpc` Analizador LPC dinámico. Este objeto calcula los coeficientes de reflexión para diferentes instantes de tiempo. Los coeficientes se obtienen por medio de la recursión de Levinson-Durbin. El filtro dinámico obtenido, puede ser guardado en un archivo LPC para después utilizarlo con el objeto `slpc~`.

El primer argumento debe ser el nombre del archivo de audio que se quiere analizar. Alternativamente se puede especificar el nombre del archivo por medio de la opción `-i`. Los demás argumentos son opcionales. El análisis será realizado al momento que el objeto reciba un mensaje `bang`.

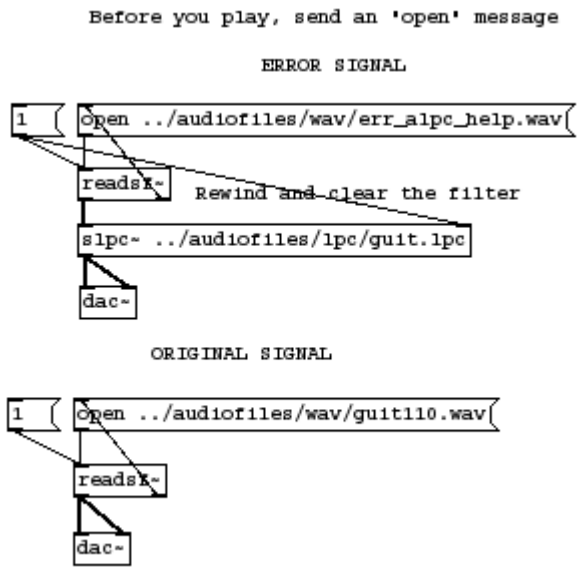


Estas son las opciones disponibles:

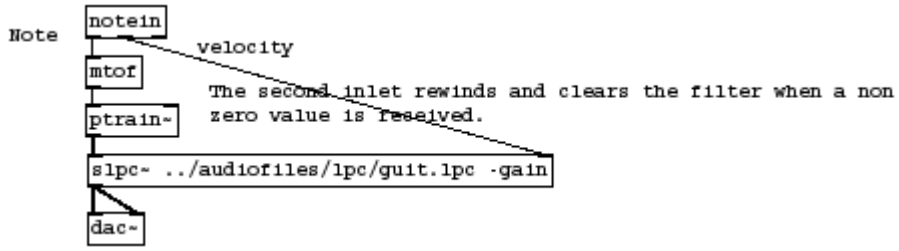
```
-beg <muestra>      Principio de la señal
-end <muestra>      Final de la señal
```

```
-p <orden>          Orden del filtro predictor
-i <nobreArchivo>   Archivo de audio que será analizado
-o <nobreArchivo>   Archivo de salida en formato LATT
-e <nobreArchivo>   Señal de error en formato WAV
-eRes <resolución>  Resolución de la señal de error
                    (16 o 32 bits)
-rRes <resolución>  Resolución de los coeficientes de
                    reflexión (16 o 32 bits)
-gRes <resolución>  Resolución del factor de ganancia
                    (16 o 32 bits)
-wSize              Tamaño de la ventana de análisis
-sSize              Salto de ventana a ventana
-f0                 Frecuencia fundamental en Hz
```

slpc~ Filtro todos polos dinámico con estructura lattice. Filtra una señal de audio por medio de un filtro lattice dinámico basado en Predicción Lineal. Los parámetros del filtro fueron previamente obtenidos durante la fase de análisis. El filtrado se realiza en tiempo real. El primer parámetro del objeto `slpc~` es el archivo LPC. Como en otros objetos, se puede utilizar también la opción `-i` para especificar el nombre del archivo. Los demás parámetros son opcionales. Si se filtra la señal de error obtenida por `alpc`, la salida de audio será una reconstrucción perfecta de la señal original.



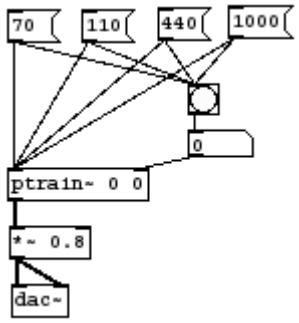
Se puede usar cualquier señal como señal de entrada. Típicamente se utiliza un tren de impulsos o ruido en la síntesis de voz por medio de LPC. Aquí hay un ejemplo de una guitarra modelada por medio de predicción lineal.



Estos son los parámetros disponibles:

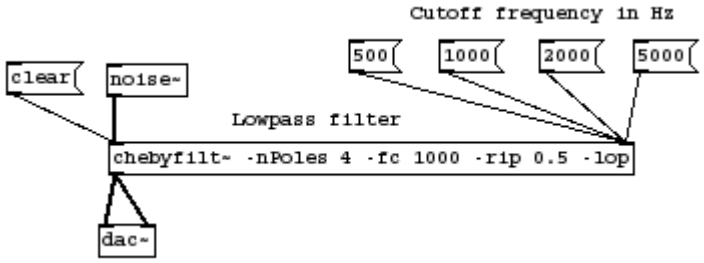
```
-i <nobreArchivo>   Archivo de audio que será analizado
-gain               Aplicar factores de ganancia
-noGain             No aplicar factores de ganancia
```

ptrain~ Genera un tren de impulsos de banda limitada. El primer inlet establece la frecuencia fundamental. El segundo inlet especifica el número de armónicos. Si el número de armónicos no es especificado, el objeto utilizara el máximo número de armónicos posibles que esta dado por $f_0/2f_s$, donde f_0 es la frecuencia fundamental y fs es la frecuencia de muestreo.



chebyfilt~ Este objeto es un filtro Chebyshev recursivo. Este filtro fue creado solamente para comparar la síntesis de Mitsuhashi filtrando y sin filtrar las componentes no controlables. Se escogió este tipo de filtro por su pronunciada caída en de la respuesta en frecuencia, pero se puede usar

cualquier otro filtro. El primer inlet acepta la señal de entrada. Cuando un mensaje `clear` es recibido en el outlet izquierdo, la memoria o estado del filtro será borrada. En otras palabras, los valores de los arreglos internos serán establecidos en cero. El inlet derecho recibe la frecuencia de corte. Se puede crear un filtro pasa banda combinando un filtro pasa bajos con un filtro pasa altos. Si el porcentaje de *ripple* se establece en 0 %, entonces se obtiene un filtro Butterworth. El número de polos debe ser un número par.



Las opciones disponibles son:

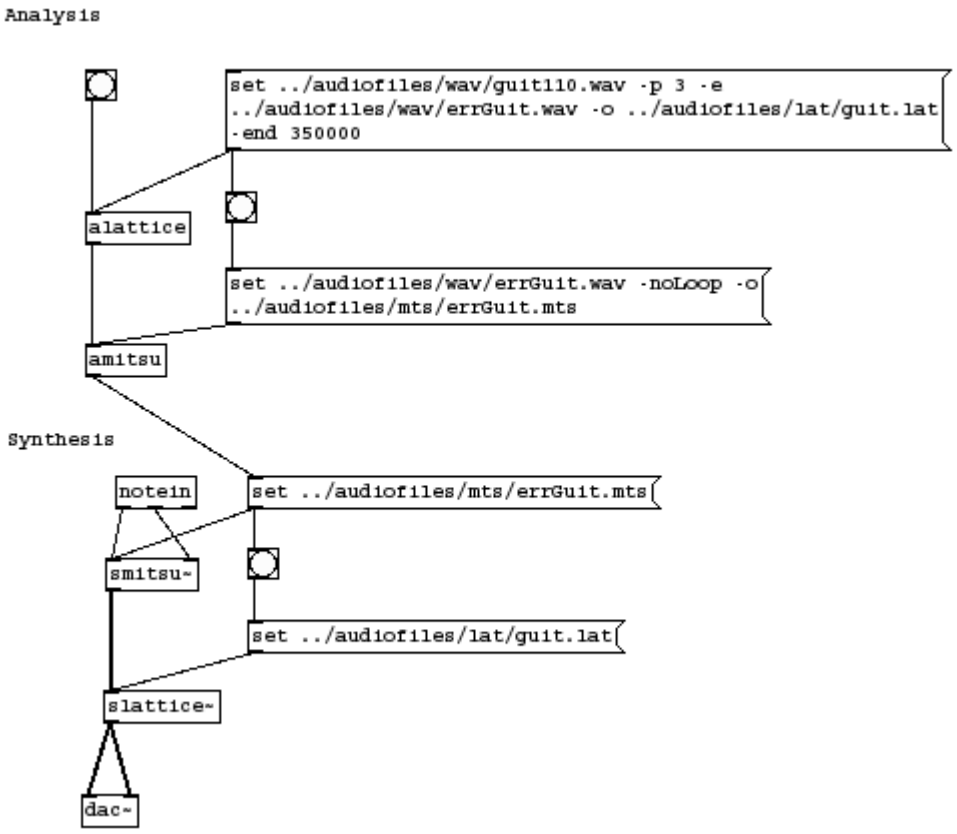
```
-nPoles <entero>   Número de polos. Debe ser un número par
-fc <frecuencia>   Frecuencia de corte en Hz
                   Valor por omisión: 100 Hz
-rip <porcentaje>  Porcentaje de ripple
                   Valor por omisión: 0.5%
-lop               Filtro pasa bajos (por omisión)
-hip               Filtro pasa altos
```

5.2. Un ejemplo de síntesis híbrida

Para concluir este capítulo, se presenta un *patch* que realiza la síntesis de un tono de guitarra, por medio de segmentación de forma de onda y predicción lineal (modelo híbrido). El *patch*, está dividido en dos partes. La parte de arriba, realiza el análisis, y la parte de abajo del *patch*, realiza la síntesis. Note que los objetos están conectados de tal manera, que cuando termine el análisis, los objetos que realizan la síntesis, estarán listos para producir el audio en tiempo real. En este ejemplo, el tono de guitarra es analizado por el objeto `alattice`, con un orden $p = 3$. Se obtiene un filtro *lattice* estático cuyos coeficientes se guardan en el archivo `guit.lat`. La señal de error es guardada en el archivo `errGuit.wav`. Posteriormente, la señal de

error resultante, es analizada por el objeto `amitsu`. La información para la síntesis de Mitsuhashi es guardada en el archivo `errGuit.mts`.

Una vez que termina el análisis, se envía un mensaje `bang` a los objetos que realizarán la síntesis. De esta manera, quedarán listos para ser utilizados. Los objetos `smitsu~` y `smitsu~`, solamente cargan los archivos `guit.lat` y `errGuit.mts`. Para escuchar el sonido resultante, será necesario utilizar un controlador MIDI. Si tiene una tarjeta de audio con puertos MIDI, puede conectar un teclado; de lo contrario, se puede utilizar algún teclado virtual y accionarlo por medio del mouse.



Capítulo 6

Pruebas subjetivas de audición

Como se mencionó en el capítulo 1, con la implementación del sintetizador híbrido en Pd se demostrará que esta técnica se puede implementar en tiempo real en una computadora personal. También se realizarán pruebas subjetivas de audición para evaluar su calidad sonora.

Se considera que la manera más confiable para medir la calidad de un sistema de audio es por medio de pruebas subjetivas de audición [4]. Existen diferentes metodologías para medir la calidad de un sistema de audio, estas van desde las que miden muy bajas calidades de audio hasta las que miden sistemas de alta fidelidad. Estos métodos son confiables, están bien definidos y han sido extensivamente probados. Muchos de estos métodos de medición se utilizan para evaluar codificadores perceptuales, como por ejemplo los codificadores MP3. En este trabajo adaptaremos la metodología MUSHRA para evaluar la calidad sonora de los algoritmos de síntesis sonora.

6.1. ¿Por qué MUSHRA?

Para realizar las pruebas subjetivas de audición se ha escogido utilizar la metodología MUSHRA[1] que está definida en la recomendación ITU-R BS.1534-1 [4].[2] Esta metodología se utiliza para evaluar subjetivamente sistemas de audio y voz de «calidad media». La reproducción de audio a través de internet, sistemas satelitales y aplicaciones multimedia portátiles son algunos ejemplos de sistemas que operan en «calidad media». En este contexto,

[1] Multiple Stimuli with Hidden Reference and Anchor

[2] International Telecommunication Union.

«calidad media» debe entenderse como un deterioro audible sufrido por la señal al ser codificada por un determinado sistema de audio.

Algunos sistemas de audio, como los codificadores perceptuales, producen un cierto deterioro[3] en la señal de audio original después de ser codificada. Este deterioro es revelado al oyente cuando se comparan auditivamente ambas señales. El deterioro es más o menos evidente dependiendo del tipo de codificación utilizado. Cuando el deterioro es muy pequeño, casi imperceptible, la recomendación ITU-R BS.1116-1 [73] resulta más apropiada para evaluar la calidad subjetiva. Esta recomendación se usa para sistemas de audio de «alta calidad». Entendiendo como «alta calidad» a que el deterioro sufrido por la señal al ser codificada es muy leve, casi imperceptible.

Siendo que en este trabajo no se va a evaluar la calidad subjetiva de codificadores perceptuales de audio, sino algoritmos de síntesis sonora, se espera que las diferencias entre las diferentes técnicas de síntesis de sonido sean más grandes de las que presentan los codificadores perceptuales de "alta calidad". Esta es una de las razones por las que se ha escogido la metodología MUSHRA. Otra razón es que en la recomendación ITU-R BS.1116-1 [73] la pruebas subjetivas se realizan en pares de estímulos, es decir que se comparan sólo dos señales a la vez. Esto hace que las pruebas sean más tardadas y se requiera un número mayor de sujetos para la experimentación. En MUSHRA, por otro lado, se comparan varias señales al mismo tiempo.

6.2. Breve explicación de la metodología MUSHRA

En la metodología MUSHRA, un sujeto escucha varias muestras de audio. Cada muestra de audio corresponde a uno de los sistemas de audio que se quiere evaluar. La tarea del sujeto es comparar estas muestras de entre sí. Al mismo tiempo, se debe comparar cada estímulo, con la muestra de audio original (referencia). El sujeto debe calificar cada uno de los estímulos en una escala de 0 a 100. Entre las muestras de audio que se deben calificar, se incluye una copia de la referencia (referencia escondida). También se incluye una o más versiones modificadas de la referencia (anclas). Las anclas sirven para ayudar al sujeto a establecer subjetivamente una escala absoluta. De esta manera, el sujeto no juzgará estímulos de muy baja calidad de manera

[3]En inglés es llamado «impairment».

muy exigente. Es importante mencionar que ni el sujeto, ni el experimentador saben en que orden se presentan los estímulos, pues la computadora los escoge al azar.

6.3. Selección de los sujetos

Según la recomendación ITU-R BS.1534-1 [4], es es preferible que se usen oyentes que tengan experiencia en escuchar sonidos de manera crítica. Así los resultados serán más confiables que si se utilizaran oyentes sin experiencia. Los sujetos deben participar en una preselección para evaluar si (1) los sujetos escuchan de manera crítica y si (2) poseen una audición normal. La preselección se realiza durante una fase de entrenamiento en la que participan todos los sujetos. Al final de las pruebas de audición se conduce una postselección. Para ser tomados en cuenta, (1) los participantes deberán mostrar calificaciones consistentes para pruebas repetidas. (2) Se pueden rechazar los resultados de los sujetos cuyos datos muestren inconsistencias en comparación con el resultado promedio de todos los sujetos para cada prueba individual. Finalmente, (3) si hay pocos sujetos que utilicen los extremos de la escala, mientras que los demás se concentran en otros valores, entonces los resultados de de estos sujetos también pueden ser rechazados. En nuestras pruebas de audición participaron, en su mayor parte, alumnos y maestros del CCADET así como músicos.

6.4. Número de participantes

En experiencias pasadas, se ha visto que no más de 20 sujetos son necesarios para obtener resultados confiables [4]. Si durante el experimento se ha logrado ya el nivel estadístico necesario para obtener conclusiones, entonces no se necesitarán más sujetos. Por el contrario, si se se ha logrado el control estadístico necesario, un número mayor de sujetos será necesario para lograr la resolución deseada.

6.5. Equipo y software

A continuación se lista el equipo y el software que se utilizó para realizar las pruebas de audición. El sistema de audio fue calibrado de acuerdo con la

recomendación ITU-R BS.1534-1. Consulte [4] si necesita información más detallada.

- Una computadora DELL Inspiron 1501 con OpenSUSE Linux 11.3.
- Audífonos Sony MDR-SA1000. Se calibró el equipo de reproducción para reproducir las señales aproximadamente a 75 dBA. Todos los participantes utilizaron los mismos audífonos con la misma configuración de volumen.
- RateIt (ver figura 6.1), una interfaz gráfica para realizar pruebas subjetivas de audición basado en la metodología MUSHRA.
- Audiómetro Bruel & Kjaer 1800

Figura 6.1: RateIt. Interfaz gráfica para la metodología MUSHRA.

6.6. Material de audio

Todas las muestras originales de audio fueron tomadas de MUMS [4] [74]. Para las pruebas se utilizaron tonos de instrumentos cuyas componentes son

[4]McGill University Master Samples

casi armónicas. Se escogieron los siguientes instrumentos musicales para las pruebas de audición:

- Instrumentos continuos: violín, saxofón, clarinete, flauta, corno y oboe
- Instrumentos impulsivos: guitarra, piano

Por cada instrumento se utilizaron nueve muestras de audio. Estas se describen a continuación con la abreviatura que se utiliza en las tablas y gráficas de resultados.

Orig La señal original.

Mit Op Síntesis de Mitsuhashi con un número óptimo $narm_{op}$ de armónicos. En la sección 2.3.4 se explica cómo se calcula el número óptimo de armónicos para la síntesis de Mitsuhashi.

Mit 50 % Síntesis de Mitsuhashi con un número de armónicos que es la mitad del óptimo $narm = narm_{op}/2$.

Mit-LPC Op Síntesis de Mitsuhashi *Mit Op* combinado con un filtro lattice estático de orden bajo.

Mit-LPC 50 % Síntesis de Mitsuhashi *Mit 50 %* combinado con un filtro lattice estático de orden bajo.

LPC Arm Síntesis por medio de LPC con filtro lattice dinámico y un tren de impulsos de banda limitada como señal excitadora. El orden del predictor está determinado por $narm_{op}$.

LPC Op Síntesis por medio de LPC con filtro lattice dinámico y un tren de impulsos de banda limitada como señal excitadora. El orden del predictor está determinado por medio del algoritmo presentado en la sección 3.3.4.

Anc 1 Primera ancla. La señal original filtrada con un filtro pasa bajos con un ancho de banda de 3.5 kHZ según la recomendación ITU-R BS.1534-1 [4]

Anc 2 Segunda ancla. La señal original filtrada con un filtro pasa bajos de tal manera que su contenido armónico sea el mismo que *Mit 50 %*. 2.3.4.

En una sesión se presentan un total de 72 muestras sin contar la fase de entrenamiento donde se utilizan 18 muestras más. Las muestras de audio tienen una duración aproximada de 5 segundos. La duración de la sesión es aproximadamente de 40 minutos. La sesión puede durar más o menos dependiendo de cada sujeto.

6.7. Entorno

Las pruebas de audición se realizaron en las instalaciones del CCADET. Se utilizó la cámara anecoica para realizar las pruebas subjetivas y las audiometrías.

6.8. Las pruebas subjetivas de audición

Las pruebas subjetivas de audición constan de dos fases. La primera es la fase de entrenamiento que tiene como propósito entrenar a los sujetos para las pruebas de audición y hacer una preselección de sujetos. En esta fase se determina si los participantes tienen una audición normal y se verificará su capacidad de escuchar sonidos de manera crítica. Durante la fase de entrenamiento se realiza una audiometría, se les da una explicación de las pruebas subjetivas de audición y se familiarizan con la interfaz gráfica. Consulte el apéndice D para mayor información sobre las instrucciones dadas durante la sesión de entrenamiento. Después de la fase de entrenamiento sigue la prueba en sí misma que consiste en calificar la calidad sonora de diferentes tonos que se presentan al sujeto de manera aleatoria.

6.9. Análisis estadístico

Por medio de las pruebas subjetivas de audición, se quiere demostrar que la calidad sonora del modelo híbrido *Mitsuhashi-LPC* es mejor que la calidad de la síntesis de *Mitsuhashi* y *LPC* por separado. En términos estadísticos, queremos comprobar que no hay diferencia de calidad de audio entre los distintos algoritmos de síntesis, o que la calidad de audio del modelo híbrido es mayor que la calidad de audio de los otros algoritmos de síntesis. Podemos entonces, expresar nuestra hipótesis nula (H_0) y nuestra hipótesis alternativa (H_1) de la siguiente manera:

H_0**:** Todos los algoritmos de síntesis tienen la misma calidad sonora[5].
$Orig = Mit\,Op = Mit\,50\,\% = Mit\text{-}LPC\,Op = Mit\text{-}LPC\,50\,\% = Lpc\,Arm = LPC\,Op = Anc1 = Anc2$

H_a**:** Al menos un algoritmo de síntesis tiene una calidad sonora diferente.

Basta con que un algoritmo de síntesis, tenga una calidad sonora diferente, para rechazar la hipótesis nula H_0. Rechazar esta hipótesis nula general, nos proporciona muy poca información, ya que lo único que se puede decir es que *por lo menos una calidad sonora es diferente a las demás*. Se pueden realizar pruebas más específicas y obtener más información, si comparamos la calidad sonora de todos los algoritmos entre sí, por parejas. Con base en la hipótesis nula H_0, podemos realizar 36 comparaciones. Cada comparación constituye, en sí misma, una hipótesis nula que declara que *la calidad sonora de dos algoritmos de síntesis es igual*. Por consiguiente tendremos 36 hipótesis individuales que probar. Para el análisis de los resultados, se pondrá especial atención en 4 grupos de comparaciones.

- Comparación entre la máxima calidad sonora y el tono original (Max y $Orig$). Esta comparación nos sirve para saber, si el algoritmo de síntesis con mayor calidad sonora, puede o no, igualar la calidad sonora del tono original.

- Comparación entre algoritmos con un número óptimo de parámetros ($Mit\,Op$, $Mit\text{-}LPC\,Op$ y $LPC\,Arm$). Esta comparación es útil para saber si el modelo híbrido (Mitsuhashi-LPC) tiene mejor calidad sonora que los algoritmos de Mitsuhashi y LPC por separado.

- Comparación entre algoritmos con un número reducido de parámetros ($Mit\,50\,\%$, $Mit\text{-}LPC\,50\,\%$ y $LPC\,Op$). Al igual que el grupo anterior, esta comparación es útil para saber si el modelo híbrido (Mitsuhashi-LPC) tiene mejor calidad sonora que los algoritmos de Mitsuhashi y LPC por separado. La diferencia con el grupo anterior, es que la síntesis se realizó con un menor número de parámetros. Nos interesa comparar algoritmos con un número aproximadamente igual de parámetros, ya

[5]En realidad, nos referimos a las *medias de calidad sonora*, pero para mejorar la claridad del texto, escribiremos *calidad sonora* solamente. La hipótesis debería formularse como sigue: H_0: Las medias de la calidad sonora de cada algoritmo de síntesis son iguales. H_1: Al menos una media de calidad sonora es diferente a las demás.

que la calidad sonora depende, en gran parte, del número de parámetros utilizados durante la síntesis. Se esperan diferencias más marcadas de calidad sonora entre los algoritmos de síntesis, cuando se utiliza un número menor de parámetros.

- Comparación entre el ancla 2 y otros modelos, en especial, los algoritmos con reducción de 50 %. El ancla 2 se obtiene filtrando el tono original por medio de la transformada de Fourier. Esta comparación nos sirve para saber si las distorsiones producidas por los métodos de síntesis influyen en el juicio de calidad sonora. En otras palabras, se espera que el algoritmo híbrido *Mit-LPC* 50 % tenga la misma calidad sonora que *Ancla*2.

La recomendación ITU-R BS.1534-1 [4] sugiere reportar las medias de cada presentación junto con su correspondiente intervalo de confianza, calculado de la siguiente manera:

$$\bar{u}_{jk} = \frac{1}{N} \sum_{i=1}^{N} u_{ijk} \tag{6.1}$$

donde u_i es la calificación del observador i para un algoritmo de síntesis k aplicado al instrumento k. N es el número de observadores.

Se establece un intervalo de confianza de 95 % que está dado por:

$$[\bar{u}_{jk} - \delta_{jk}, \bar{u}_{jk} + \delta_{jk}] \tag{6.2}$$

$$\delta_{jk} = t_{0.05} \frac{S_{jk}}{\sqrt{N}} \tag{6.3}$$

donde $t_{0.05}$ es el valor t para un nivel de significancia del 95 %.

La desviación estándar de cada presentación esta dada por:

$$S_{jk} = \sqrt{\sum_{i=1}^{N} \frac{(\bar{u}_{jk} - u_{ijk})^2}{N-1}} \tag{6.4}$$

Además de la presentación de las medias con intervalos de confianza, resulta oportuno en el presente estudio, hacer una comparación de promedios por parejas con corrección de Holm [75]. Estas comparaciones se presentan en la sección 6.10. Los cálculos estadísticos están basados en los escritos de

Lane [76], Wright [75] y Simens [77]. El código para calcular los intervalos de confianza y las tablas de comparación de promedios por parejas, se encuentra en los listados C.11 y C.12.

6.10. Reporte de resultados

Como se mencionó en la sección 6.9, la recomendación ITU-R BS.1534-1 [4] sugiere reportar las medias de calidad sonora junto con su correspondiente intervalo de confianza. Estos resultados se presentan con gráficos de barras con barras de error. A estos gráficos se les ha añadido unas tablas de comparación de promedios por parejas en la mitad inferior de la figura. Estas tablas sirven para complementar la información proporcionada gráficamente. La figura 6.2 es un ejemplo de este tipo de gráfico. Además de las medias con intervalos de confianza que manda el estándar, se han incluido, en el apéndice A, gráficos de cajas[6]. A continuación damos una breve explicación de cómo se interpretan los gráficos y las tablas de comparación.

6.10.1. Gráficos de cajas

Los gráficos de cajas son muy útiles para mostrar la dispersión de los datos y la presencia de valores atípicos. Estos gráficos se presentan en el apéndice A. Observemos, por ejemplo, la figura A.1. La media (promedio) está marcada con el símbolo "+". El borde inferior de la caja representa el primer cuartil (percentil 25). El borde superior, marca el tercer cuartil (percentil 75). La línea interior de la caja muestra la mediana. Los extremos de los bigotes marcan los límites dentro de los cuales se considera que los valores son típicos. Más allá de estos límites, los valores son considerados atípicos y están marcados con asteriscos. Los círculos, representan valores aún más atípicos.

6.10.2. Gráficos de barras con intervalos de confianza

Un ejemplo de estos gráficos se muestra en la figura 6.2. La altura de las columnas verticales, representa la calidad sonora. Note que las barras están representadas con dos colores diferentes. El grupo de la izquierda, en color azul claro, corresponde a los algoritmos con un número «óptimos» de

[6]En inglés: Box-Whisker-Plots

parámetros. El grupo de la derecha, en color azul marino, corresponde a los algoritmos de síntesis con un número de parámetros reducido en un 50 %. Los bigotes, o barras de error, representan los intervalos de confianza, calculados por medio de las ecuaciones 6.1, 6.2 y 6.3, a un nivel de 95 %. Un intervalo de confianza a un nivel de 95 % se puede interpretar de la siguiente manera: si se realizaran las pruebas auditivas varias veces, el verdadero valor de calidad sonora, estaría contenido dentro del intervalo de confianza 95 % de las veces. Podemos interpretar este intervalo, como una medida de incertidumbre, es decir que el valor de calidad sonora podría ser cualquier valor dentro de este intervalo. Las barras de error son muy útiles para comparar a simple vista dos valores de calidad sonora. Cuando los intervalos de confianza no se traslapan, podemos decir que la diferencia de calidad sonora entre dos modelos de síntesis es *significativa*. Debemos tener cuidado con el término *significativo* porque puede causar confusiones. En nuestro contexto, *significativo* quiere decir, que hay suficiente evidencia para dudar de nuestra hipótesis nula: *las medias de calidad sonora son iguales*. Se considera *significativa*, a una diferencia que es de un tamaño tal que sería improbable haberla obtenido por casualidad. No es considerada *significativa*, una diferencia de valores, que es tan pequeña, que podría atribuirse a la variabilidad de los resultados. En este caso, se dice que no hay suficiente evidencia para dudar de la hipótesis nula. El hecho de no tener suficiente evidencia para rechazar la hipótesis nula, no quiere decir que pueda ser aceptada. Sin embargo, en la práctica, podemos asumir que los tonos de los dos algoritmos de síntesis, son auditivamente indistinguibles, cuando la diferencia de calidad sonora es muy pequeña.

6.10.3. Tablas de comparaciones por parejas

Se pueden utilizar los intervalos de confianza para determinar si la diferencia de calidad sonora, entre dos algoritmos de síntesis, es significativa. Sin embargo, resulta más apropiado utilizar valores de probabilidad (*valor P*). Se han agregado los valores de probabilidad a los gráficos de barras. De esta manera, resulta muy cómodo comparar la calidad sonora e intervalos de confianza, con los valores de probabilidad. La figura 6.2 es un ejemplo. Note que la mitad superior, contiene los valores de calidad sonora y la mitad inferior, las tablas de comparaciones. Los valores de la tabla, nos muestran la probabilidad de obtener un resultado como el que hemos obtenido, *suponiendo que la hipótesis nula es cierta*. Otra manera de interpretar el *valor P* es que mientras menor es el valor de probabilidad, mayor es la evidencia

de que la hipótesis nula es falsa. En otras palabras, un *valor P* pequeño, nos indica que es muy improbable que nuestros resultados se deban al azar.

Veamos un ejemplo. En la tabla 6.2 se muestra el valor $p = 0.03$ entre los modelos de síntesis $Lpc\,Arm$ y $LPC\,Op$. Si la hipótesis nula ($Lpc\,Arm = LPC\,Op$) fuese verdad, habría tan solo un 3 % de probabilidad de haber obtenido una diferencia igual o menor que $|Lpc\,Arm - Lpc\,Op|$[7]. Como la probabilidad es muy baja, resulta dudoso que la hipótesis nula ($Lpc\,Arm = LPC\,Op$) sea cierta. En este caso, decimos que la diferencia de calidad sonora es *significativa*. Aunque el *valor P* es una variable continua, normalmente se establece un *nivel de significancia*, como criterio para decidir si se rechaza la hipótesis nula. Típicamente, si el *valor P* es menor que 0.05, se rechaza la hipótesis nula. Valores mayores a 0.05 indican que no hay suficiente evidencia para rechazarla.

Veamos otro ejemplo donde el valor de probabilidad es mayor a 0.05. En la tabla 6.2 se muestra el valor $p = 0.43$ entre los modelos de síntesis *Mit Op* y *Mit-LPC Op*. Este valor nos indica que hay un 43 % de probabilidades de haber obtenido una diferencia igual o menor que $|Mit\,Op - Mit\text{-}LPC\ Op|$, suponiendo que la hipótesis nula ($Mit\,Op = Mit\text{-}LPC\,Op$) fuese verdad. Como la probabilidad es muy alta, no hay suficiente evidencia para rechazar la hipótesis nula y probablemente esta diferencia se deba meramente al azar. En las tablas de comparaciones, las diferencias *no significativas* están marcadas en color rojo.

6.10.4. Resultados generales

En nuestro experimento participaron 20 sujetos. Se realizaron audiometrías a todos los participantes. Sólo participaron en las pruebas auditivas los sujetos que mostraron audición normal. Siguiendo los criterios de postselección, se rechazaron 5 sujetos. Como no hubo variaciones importantes en los resultados, se decidió presentar los resultados tomando en cuenta a todos los participantes.

Como se puede observar en la figura A.1, los datos están muy dispersos. La razón es que cada sujeto es diferente en cuanto a la exigencia con la que juzgan la calidad sonora de las muestras de audio. Algunos participantes son muy exigentes y otros son muy benevolentes. Mas que un resultado absoluto de la calidad sonora, nos interesa comparar los modelos de síntesis entre

[7]Por causa del azar.

sí. En el gráfico se muestran muchos valores atípicos marcados con círculos y asteriscos. Muchos de estos valores atípicos se encuentran en la columna correspondiente a la muestra original. Los valores de la columna *Orig* que están por abajo de 80 (*excelente*) podrían ser un indicativo de que el sujeto no es capaz de escuchar los sonidos de manera crítica, de tal forma que los resultados de estos sujetos podrían ser eliminados del análisis estadístico. Otra posible interpretación sería que el sujeto sí es capaz de escuchar los sonidos de manera crítica, pero no comprendió que su tarea era basar su juicio de calidad sonora en la comparación de cada muestra con la señal original. Puede ser que al sujeto no le gustó el sonido original y por eso le dio una calificación baja. El sujeto, en este caso, todavía es capaz de dar un juicio comparativo entre las diferentes muestras de audio, pero sus evaluación será en general más exigente. Este problema podría solucionarse en el futuro rediseñando la fase de entrenamiento para asegurase de que el sujeto comprende la tarea que se le ha asignado. La interfaz gráfica de *Rateit* también influye en la dispersión de los datos. Algunos sujetos se concentran más en los valores numéricos que en las etiquetas de calidad sonora (excelente, bien, regular, pobre, mal). Una mejora de la interfaz gráfica, sería la eliminación de los valores numéricos.

La figura 6.2 muestra los promedios generales de calidad sonora. El eje horizontal representa los modelos de síntesis y en el eje vertical la calidad sonora. Las barras de error representan los intervalos de confianza a un nivel de 95 %. La mitad inferior muestra la tabla de comparaciones por parejas.

Como se puede ver en la figura 6.2, el modelo de síntesis sonora de mayor calidad es *Mit-Lpc Op*. A pesar de que se utilizaron los parámetros óptimos para la síntesis, la calidad sonora de *Mit-Lpc Op*, resultó inferior a la calidad del sonido original (*Orig*). Se esperaba que el modelo híbrido con parámetros óptimos fuera auditivamente indistinguible del tono original, pero como se muestra en la figura, este no es el caso.

Se esperaba que el modelo híbrido *Mit-Lpc Op* fuera mejor que los demás modelos de síntesis, y en efecto, la calidad del modelo híbrido *Mit-Lpc Op* resultó superior a la calidad de *Mit Op* y *LPC Arm*. Sin embargo, la diferencia entre *Mit-Lpc Op* y *Mit Op* es muy pequeña. Según la tabla de comparación, hay un 43 % de probabilidad, de que esta diferencia se haya dado por casualidad. Quizás, con un mayor número de pruebas de audición y de participantes, se podría obtener una diferencia significativa. Podemos concluir que, en general, el modelo híbrido *Mit-Lpc Op* tienen una calidad sonora mayor a *Mit Op* y *LPC Arm*, con la reserva de que, la diferencia entre *Mit-Lpc Op* y *Mit Op*

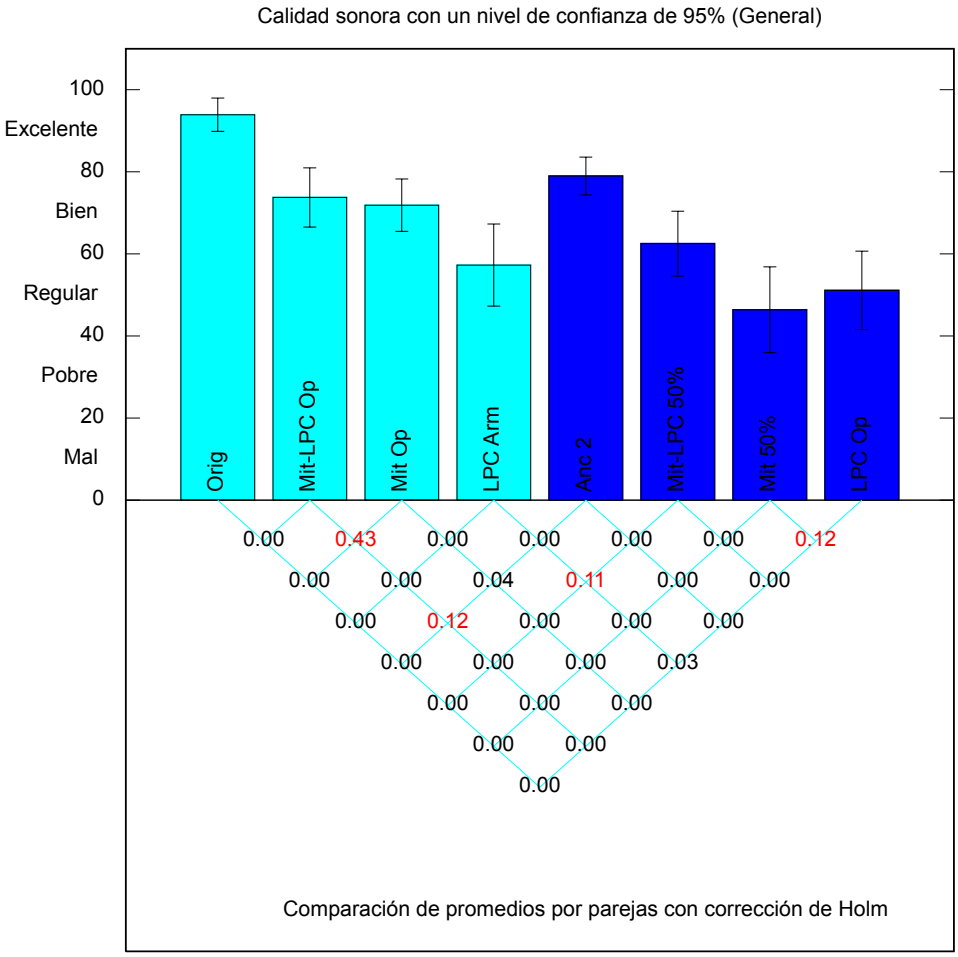


Figura 6.2: Medias de calidad sonora con intervalos de confianza y comparación de promedios por pareja (General)

puede haberse dado por casualidad.

Comparemos ahora los algoritmos que se sintetizaron con un menor número de parámetros. Según la figura 6.2, la calidad de $Mit\text{-}Lpc\,50$ es mayor que $Mit\,50$. Los intervalos de confianza no se traslapan y el valor de probabilidad es $p = 0.00$. Esto quiere decir que, en general, la calidad sonora de la síntesis por segmentación de onda mejora cuando se combina con la predicción lineal, especialmente cuando se utiliza un menor número de parámetros (reducción del 50 %). La calidad sonora de $Mit\text{-}Lpc\,50$ es también superior a $LPC\,Op$. Aunque los intervalos de confianza se traslapan, su valor de probabilidad es $p = 0.00$. Esta es una evidencia de que el modelo híbrido, tienen una calidad sonora mayor que los modelos LPC cuando se reduce el número de parámetros.

Ahora comparemos $Anc2$ con $Mit\text{-}LPC\,50$ y $Mit\,50$. La calidad sonora de $Anc2$ es mayor a la calidad de $Mit\text{-}LPC\,50$ y $Mit\,50$. Esta afirmación está justificada por el valor de probabilidad $p = 0.00$ y porque los intervalos de confianza no se traslapan. Este hecho, nos hace pensar, que el ruido (o la ausencia de él) juega un papel importante en el juicio de calidad sonora. Un modelo de síntesis de banda limitada, como la síntesis aditiva, ausente de distorsiones, es más probable que sea juzgado por el oyente con una mayor calidad sonora.

Se han presentado los resultados generales, pero se debe tener cuidado con su interpretación, porque los juicios de calidad sonora dependen mucho del tipo de instrumento que se está evaluando. Enseguida discutimos los resultados de cada instrumento. Los instrumentos están clasificados en tres grupos de acuerdo a su viabilidad de ser sintetizados por medio del modelo híbrido propuesto en este trabajo. El primer grupo (corno, clarinete, oboe y saxofón) comprende los instrumentos que pueden ser sintetizados con segmentación de forma de onda y predicción lineal, obteniendo los mejores resultados. El segundo grupo (flauta y violín) muestra una calidad sonora menor, quizás porque sus mecanismos de excitación (el viento y el frote del arco) presentan ruido. El último grupo (guitarra y piano) obtuvo la calidad sonora más baja. Se obtienen mejores resultados utilizando LPC para síntetizar los tonos de estos instrumentos.

6.10.5. Instrumentos continuos no ruidosos: corno, clarinete, oboe y saxofón

La calidad sonora de estos instrumentos fue, en general, buena según la escala de calidad sonora. Los métodos de síntesis híbridos (*Mit-LPC OP* y *Mit-LPC* 50 %) presentan una calidad sonora mayor que sus correspondientes métodos de síntesis individuales (*Mit OP*, *Mit* 50 %, *LPC Arm* y *LPC Op*), como se puede verificar en las figuras 6.3, 6.4, 6.5, 6.6. Posiblemente los ruidos producidos por la interpolación, las inestabilidades de los filtros y la imprecisión numérica, son enmascarados por el contenido armónico de los tonos de estos instrumentos. Estos instrumentos, son los mejores candidatos para ser sintetizados por medio de los algoritmos de síntesis híbridos propuestos en este trabajo. Enseguida se discuten, los resultados de cada instrumento.

Corno

Como se puede observar en la figura 6.3, el algoritmo de síntesis juzgado con mejor calidad sonora fue *Mit-LPC Op*. No hay mucha diferencia entre la calidad sonora de *Mit-LPC Op* y la calidad del tono original (*Orig*). Los intervalos de confianza se traslapan y el valor de probabilidad es $p = 0.19$. No se cuenta con suficiente evidencia para rechazar la hipótesis de que *la calidad sonora de Mit-LPC Op y la calidad sonora del tono original Orig son iguales*. Una interpretación informal sería que *Mit-LPC Op* es auditivamente indistinguible del tono original. Ambos tonos fueron juzgados con una calidad *excelente* según la escala de calidad sonora.

La calidad sonora de *Mit-LPC Op* es mayor que la calidad de *Mit Op* y *LPC Arm*, como se puede corroborar en la figura 6.3. Los intervalos de confianza no se traslapan y los valores de probabilidad son $p = 0.00$. Esto quiere decir que el uso del filtro LPC, en efecto, ayudó a mejorar la calidad sonora del algoritmo de Mitsuhashi.

La calidad de *Mit-LPC* 50 % es mayor que la calidad de *Mit* 50 %. Quiere decir que la síntesis de Mitsuhashi mejoró con la utilización del filtro LPC. La calidad de *Mit-LPC* 50 % también es mayor que *LPC Op*. Sin embargo, la diferencia entre *Mit-LPC* 50 % y *LPC Op* no es significativa, puesto que su valor de probabilidad es $p = 0.25$. Podemos decir que la calidad del modelo híbrido es mejor que la calidad de la síntesis de Mitsuhashi, pero no hay suficiente evidencia para justificar que la calidad del modelo híbrido sea

mejor que la calidad de la predicción lineal. Quizás con un mayor número de muestras se podría obtener una diferencia significativa.

La calidad sonora de los algoritmos de predicción lineal (*LPC Arm* y *LPC Op*) es muy similar. Tienen un valor de probabilidad de $p = 1.00$. Hay una probabilidad muy grande de que la poca diferencia de calidad que hay entre estos dos algoritmos, se deba al azar. En la práctica, significa que estos dos tonos son indistinguibles auditivamente. Por consiguiente, si se quisiera sintetizar un corno por medio de LPC, se escogería el orden más pequeño entre *LPC Arm* y *LPC Op* para ahorrar parámetros y tiempo de cómputo. En este caso *LPC Arm* contenía 56 coeficientes y *LPC Op* 29. El uso de *LPC Op* representaría un ahorro de casi 50 %.

La calidad sonora de *Ancla*2 es mayor que *Mit-LPC* 50 % con un valor de probabilidad de $p = 0.02$. Como se mencionó anteriormente, esto es un indicativo de que el ruido es un aspecto importante en el juicio de la calidad sonora y que un sistema de banda limitada, como la síntesis aditiva, es juzgado con una mayor calidad sonora.

Clarinete

El algoritmo juzgado con la mayor calidad sonora fue *Mit Op*, según la figura 6.4. La diferencia de calidad sonora entre *Mit Op* y el tono original (*Orig*) es muy pequeña; de hecho, el valor de probabilidad es $p = 1.00$. No podemos aceptar la hipótesis de que *la calidad sonora del tono original (Orig) es igual a la de Mit Op*. Sin embargo, el valor P es tan grande que podemos afirmar, de manera informal, que estos tonos fueron juzgados con la misma calidad sonora o que son auditivamente indistinguibles.

La calidad sonora de *Mit Op* tampoco es significativamente diferente a *Mit-LPC Op* ya que la tabla de la figura 6.4 muestra un valor de probabilidad $p = 0.49$. En el caso del clarinete, la combinación de la síntesis de Mitsuhashi con la predicción lineal no mejoró la calidad sonora. Entonces, es mejor sintetizar este tono utilizando solamente la síntesis por segmentación de onda.

La calidad sonora de *Mit-LPC* 50 % es mayor que la calidad de *Mit* 50 %. La diferencia de calidad sonora no es significativa, ya que el valor de probabilidad entre estos dos algoritmos es $p = 0.08$. No hay suficiente evidencia para rechazar la hipótesis de que *la calidad de Mit* 50 % *es igual a la calidad de Mit-LPC* 50 %. Cómo el valor de probabilidad no es muy grande, podemos pensar que la calidad de *Mit-LPC* 50 % sí es mayor que la cali-

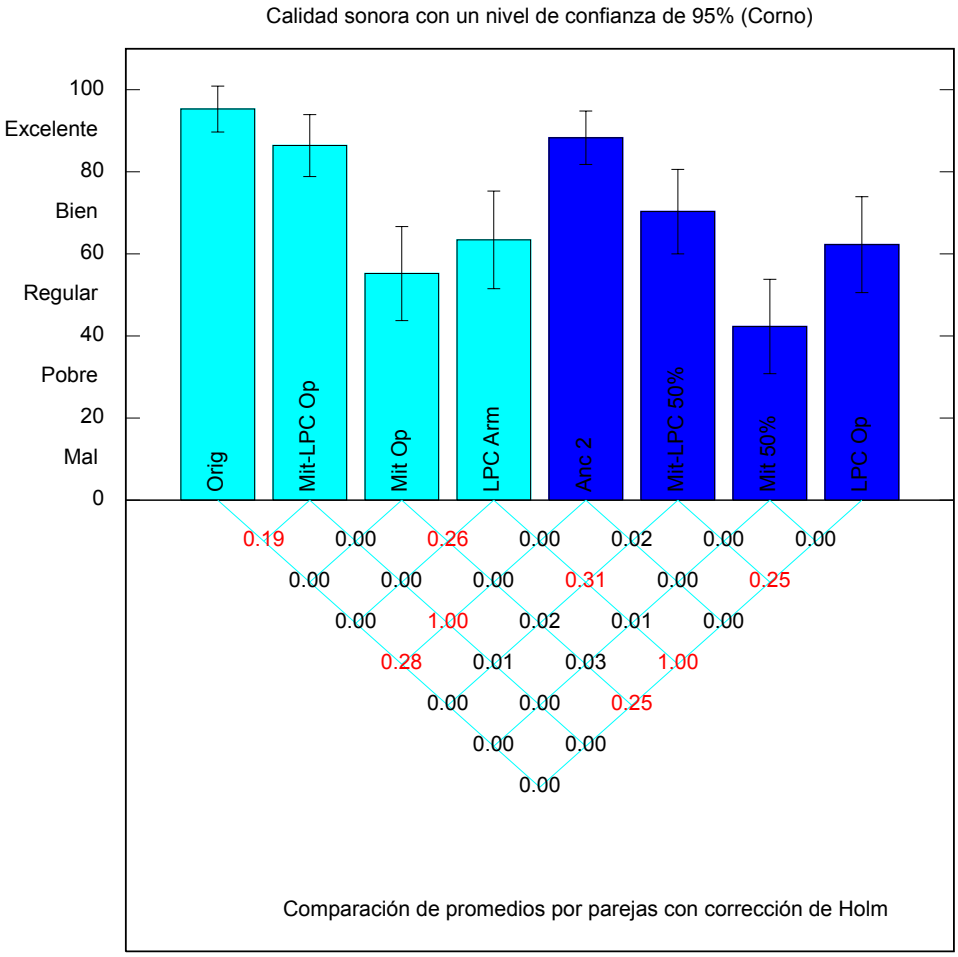


Figura 6.3: Medias de calidad sonora con intervalos de confianza y comparación de promedios por pareja (Corno)

dad de $Mit\,50\,\%$, pero se necesitan más pruebas auditivas para obtener una diferencia significativa.

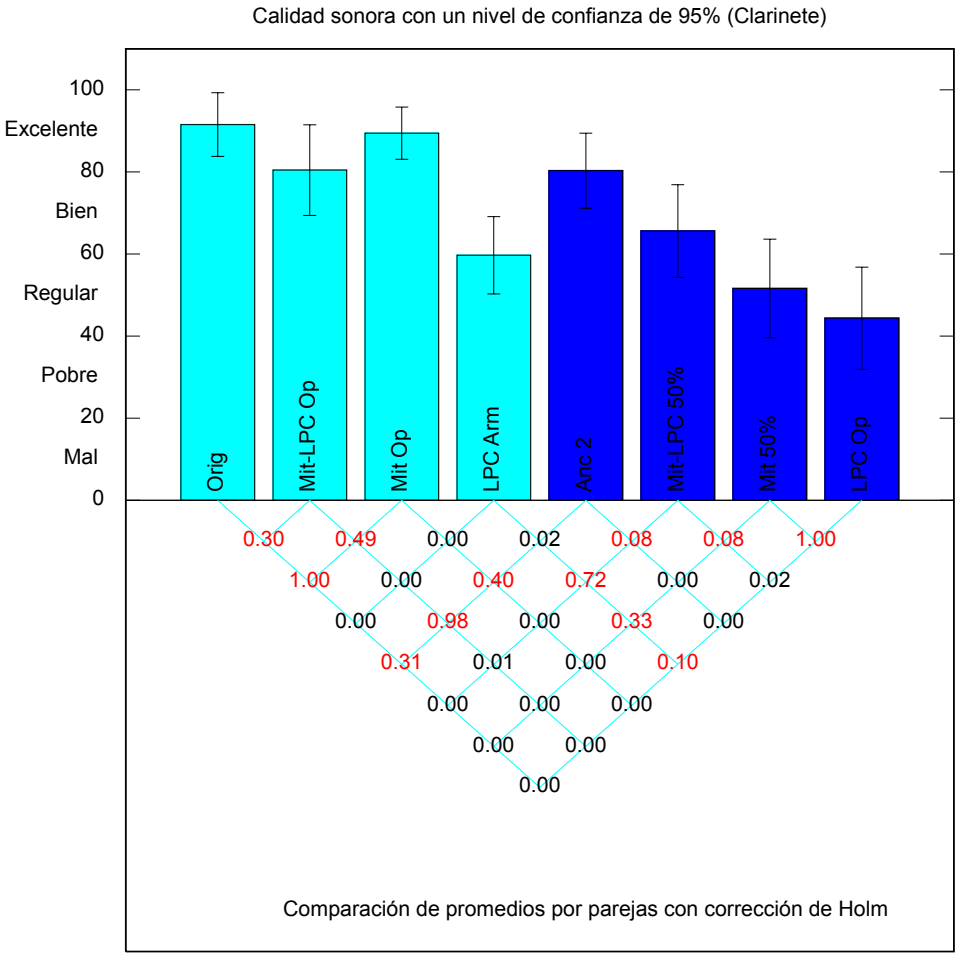


Figura 6.4: Medias de calidad sonora con intervalos de confianza y comparación de promedios por pareja (Clarinete)

Oboe

El algoritmo con mayor calidad sonora es *Mit-LPC Op*, como se muestra en la figura 6.5. La diferencia de calidad sonora entre *Mit-LPC Op* y el tono original *Orig*, es muy pequeña. Los intervalos de confianza se traslapan y

el valor de probabilidad es $p = 1.00$. Tenemos evidencia para pensar que *Mit-LPC Op* y el tono original fueron indistinguibles auditivamente.

La calidad sonora de *Mit-LPC Op* es mayor que *Mit Op* y *LPC Arm*, pero la diferencia entre *Mit-LPC Op* y *Mit Op* es muy pequeña. El valor de probabilidad de $p = 0.50$ indica que no se tiene evidencia suficiente para rechazar la hipótesis de que *la calidad sonora de Mit-LPC Op y Mit Op es igual*.

La calidad de *Mit-LPC* 50 % es mayor que *Mit* 50 % y *LPC Op*. Las diferencias son lo suficientemente grandes para concluir que no son causa de la suerte. La prueba de ello es que los intervalos de confianza no se traslapan y los valores de probabilidad son $p = 1.00$. En el caso del oboe, se cumple que la calidad mejora cuando se combina la síntesis de Mitsuhashi con la predicción lineal. Una observación muy interesante es que *Mit-LPC* 50 %, *Mit-LPC Op* y *Orig* tienen casi la misma calidad sonora, con un valor de probabilidad $p = 1.00$. En consecuencia, podríamos utilizar *Mit-LPC* 50 % para ahorrar información, logrando una calidad sonora muy similar al tono original. En el caso del tono de oboe, tanto la predicción lineal, como la síntesis de Mitsuhashi son superados por el modelo híbrido propuesto en este trabajo.

Saxofón

Los resultados del saxofón son muy similares a los del oboe. El algoritmo juzgado con mayor calidad sonora fue *Mit-LPC Op*, como se muestra en la figura 6.6. Sin embargo, *Mit-LPC Op* no es significativamente diferente al tono original (*Orig*), ni tampoco a los algoritmos *Mit Op* y *Mit-LPC* 50 %. La figura 6.6 muestra los valores de probabilidad. Al igual que el oboe, en el tono de saxofón, se puede lograr un ahorro importante de información, ya que podemos utilizar *Mit-LPC* 50 % logrando una calidad sonora muy similar al tono original. En el tono de saxofón, la calidad sonora del modelo híbrido supera a la síntesis de Mitsuhashi y la predicción lineal.

6.10.6. Instrumentos continuos con ruido: flauta y violín

La calidad sonora de los tonos sintetizados de estos instrumentos resultó regular, de acuerdo con la escala de calidad sonora. En el caso del violín y la flauta, los métodos de síntesis con mayor calidad fueron *Mit Op*

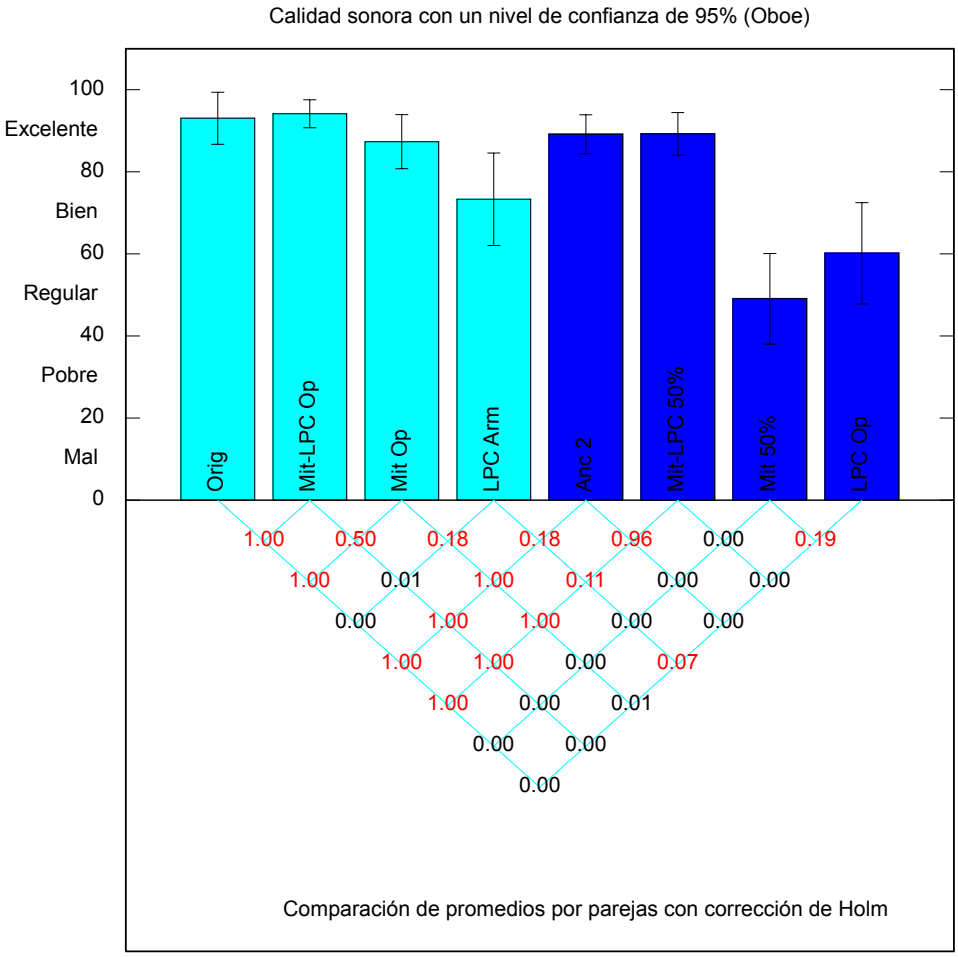


Figura 6.5: Medias de calidad sonora con intervalos de confianza y comparación de promedios por pareja (Oboe)

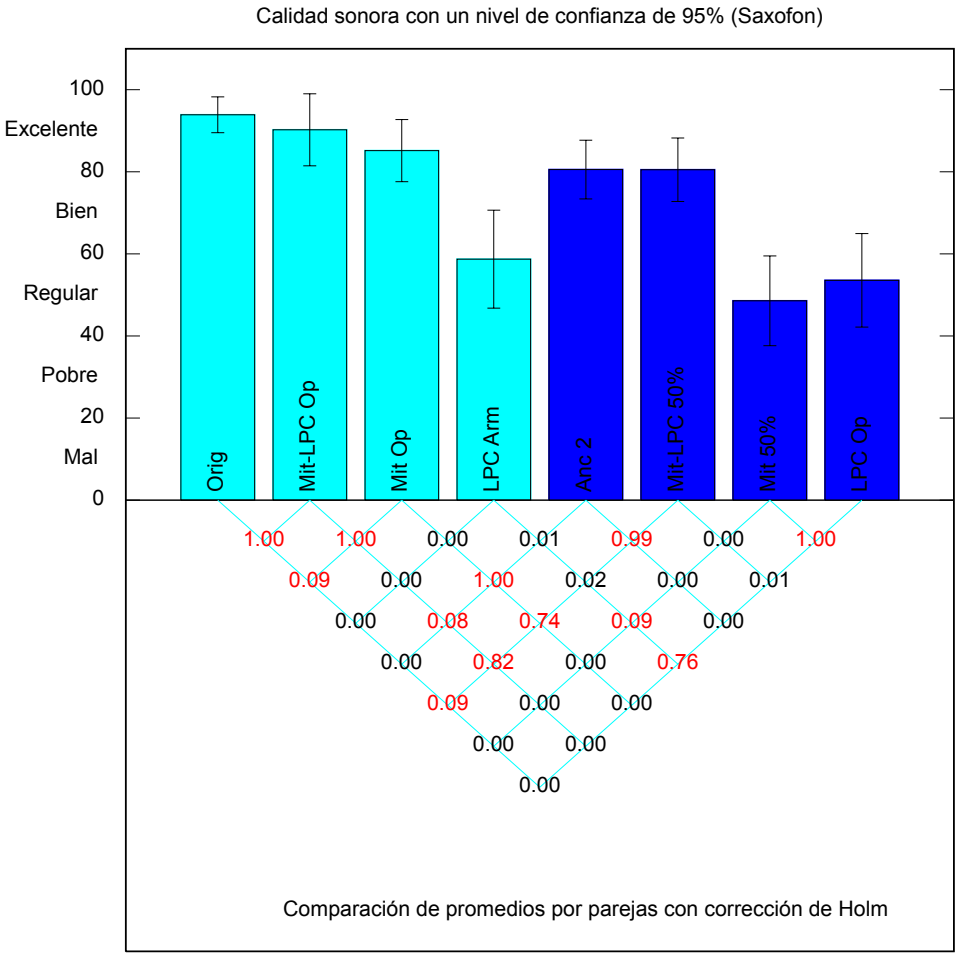


Figura 6.6: Medias de calidad sonora con intervalos de confianza y comparación de promedios por pareja (Saxofon)

y *Mit-LPC Op*. La calidad sonora de los demás modelos de síntesis es muy similar y no presentaron diferencias significativas. La utilización de la predicción lineal, no mejoró la calidad sonora. El tono de estos dos instrumentos contiene mucho ruido producido por su mecanismo de excitación. El ruido del violín proviene del frote del arco con las cuerdas y el ruido de la flauta proviene del exceso de aire que se escapa de la boquilla. Estos ruidos afectan el análisis *LPC*. El resultado sonoro, a la hora de la reproducción, es un ruido de fondo parecido al que producen los discos antiguos de acetato. Una posible solución es separar el ruido de las componentes armónicas y realizar la síntesis de estos componentes por separado. Esto se puede lograr usando las técnicas que se usan en la síntesis paramétrica [54, 78–84]. También se puede separar la parte armónica y el ruido de una señal utilizando la predicción lineal. La separación de un tono en ruido y componentes armónicas no es un problema trivial. No ahondaremos más en este tema, pero aquí se menciona para tomarlo en cuenta en un trabajo de investigación posterior. No discutiremos los resultados individuales de estos instrumentos pero puede consultar las figuras 6.7 y 6.8 para mayor información.

6.10.7. Instrumentos impulsivos: guitarra y piano

Los tonos sintetizados de guitarra y piano mostraron una calidad sonora pobre y regular, según la escala de calidad sonora. En el caso del piano, la calidad sonora de los métodos de síntesis es muy similar. Por consiguiente, las diferencias de calidad no son significativas. Esto se puede corroborar en la figura 6.10. Observe que en la región central se muestran puros números en color rojo, que indican probabilidades mayores a 0.05. Es importante mencionar que los resultados del piano no son muy representativos, porque hubo un error en el detector de altura a la hora del análisis. Los tonos sintetizados tenían variaciones pequeñas pero perceptibles de altura. Los sujetos reportaban que los tonos se escuchaban desafinados. Esto es una evidencia de que la diferencia de altura es un factor importante en el juicio de calidad sonora. Un sujeto, incluso, reportó su desagrado por el tono original. Para el lector curioso, la muestra utilizada fue tomada del archivo `MPP PIANO LOUD_C5.wav` de MUMS [74].

En el caso de la guitarra, la predicción lineal presenta la mayor calidad sonora, como se muestra en la figura 6.9. La calidad sonora de los modelos de predicción lineal (*LPC Arm* y *LPC Op*) es mayor que la calidad de los modelos de Mitsuhashi (*Mit Op* y *Mit* 50 %). Sus valores de probabilidad

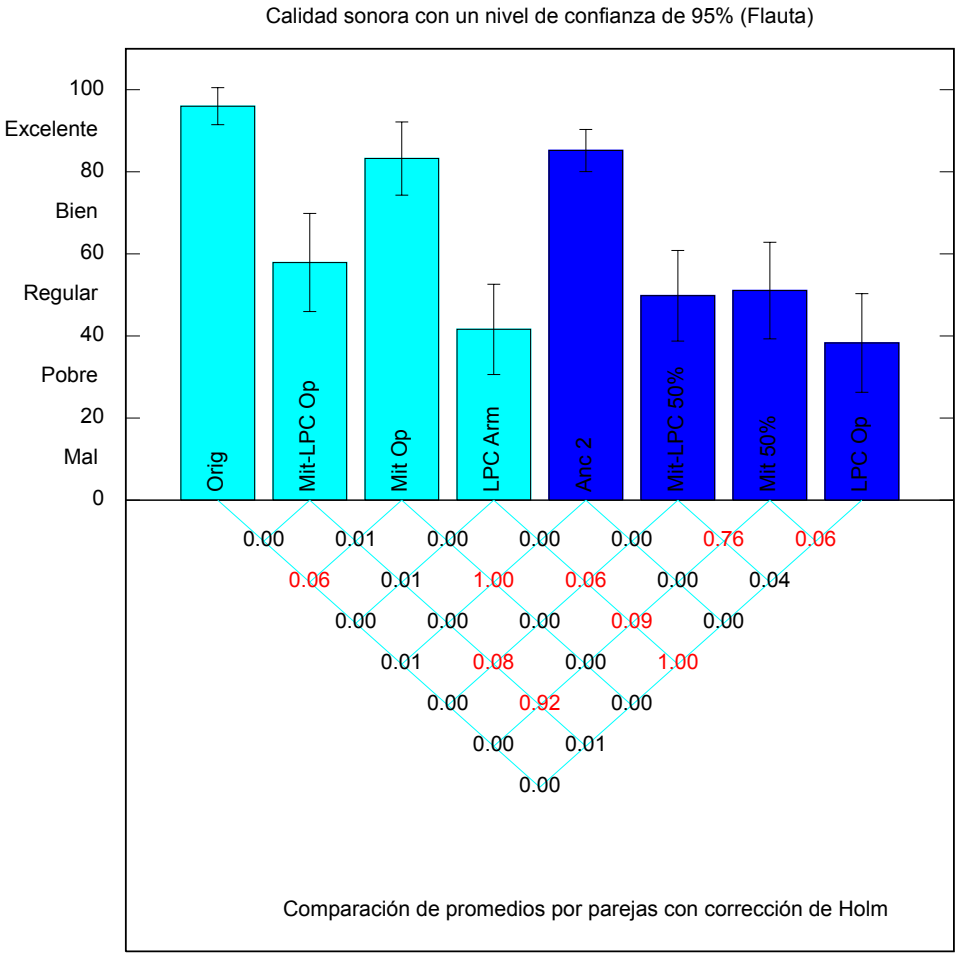


Figura 6.7: Medias de calidad sonora con intervalos de confianza y comparación de promedios por pareja (Flauta)

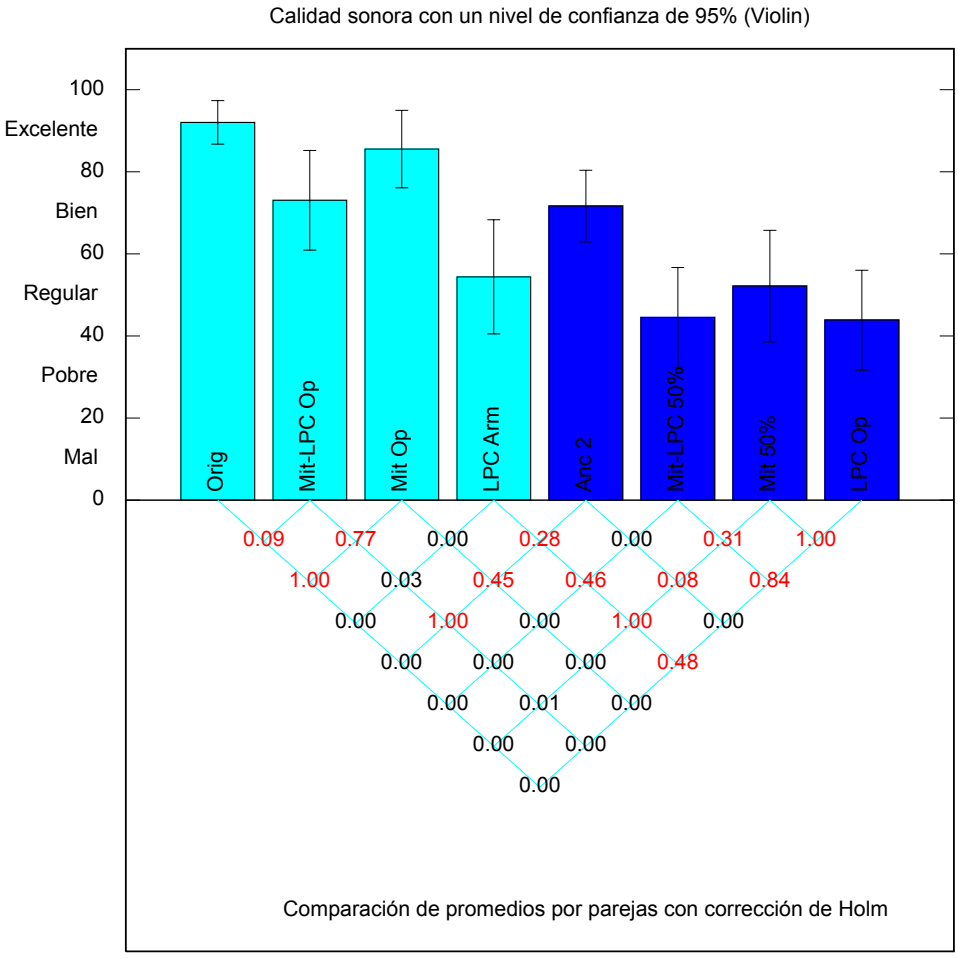


Figura 6.8: Medias de calidad sonora con intervalos de confianza y comparación de promedios por pareja (Violin)

son $p = 0.01$. En cuanto al ahorro de información, los modelos de predicción lineal son preferibles a los modelos híbridos porque requieren, en general, de un menor número de parámetros.

El problema con el piano y la guitarra, es que sus armónicos agudos tienen poca amplitud. Las distorsiones y los ruidos producidos por los filtros LPC contienen armónicos agudos, que por su amplitud, no llegan a ser enmascarados por los armónicos del tono original. Esto hace que las distorsiones sean más notorias. La solución sería mejorar la estabilidad y precisión numérica de los filtros LPC. Una posibilidad es utilizar la descomposición de Cholesky en lugar de la recursión de Levinson a la hora del análisis [2, 45]. Otra opción es la utilización de la predicción lineal pura(PLP), propuesta por den Brinker [54].

Otra característica importante de estos tonos, a diferencia de los tonos continuos, es que su amplitud decae rápidamente. La grabación de estos tonos tiene una duración aproximada de 5 segundos. Sin embargo, después de aproximadamente 2 segundos, se escucha casi únicamente el ruido de fondo. Los tonos sintetizados suenan bien en la primera parte, después de la mitad, se escuchan ruidos que son producidos por el algoritmo de síntesis. Este ruido influye en el juicio de calidad sonora de los oyentes. En otras palabras, una vez que el tono impulsivo ha decaído lo suficiente, el oyente compara dos tipos de ruido de fondo, donde el sintetizado, es más desagradable. Es probable, que la calidad sonora reportada por los sujetos habría resultado mayor, si los tonos utilizados en las pruebas auditivas hubieran tenido una duración menor a 2 segundos. Es importante considerar este hecho para posteriores pruebas auditivas con instrumentos impulsivos.

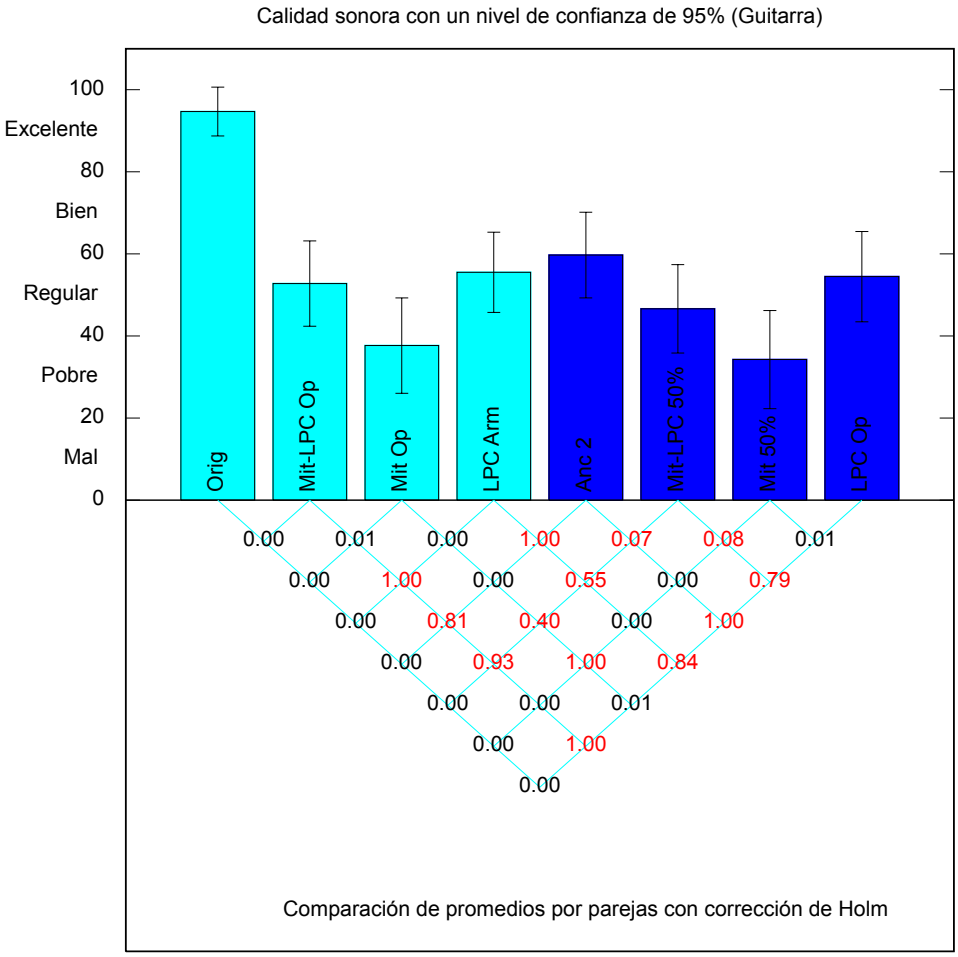


Figura 6.9: Medias de calidad sonora con intervalos de confianza y comparación de promedios por pareja (Guitarra)

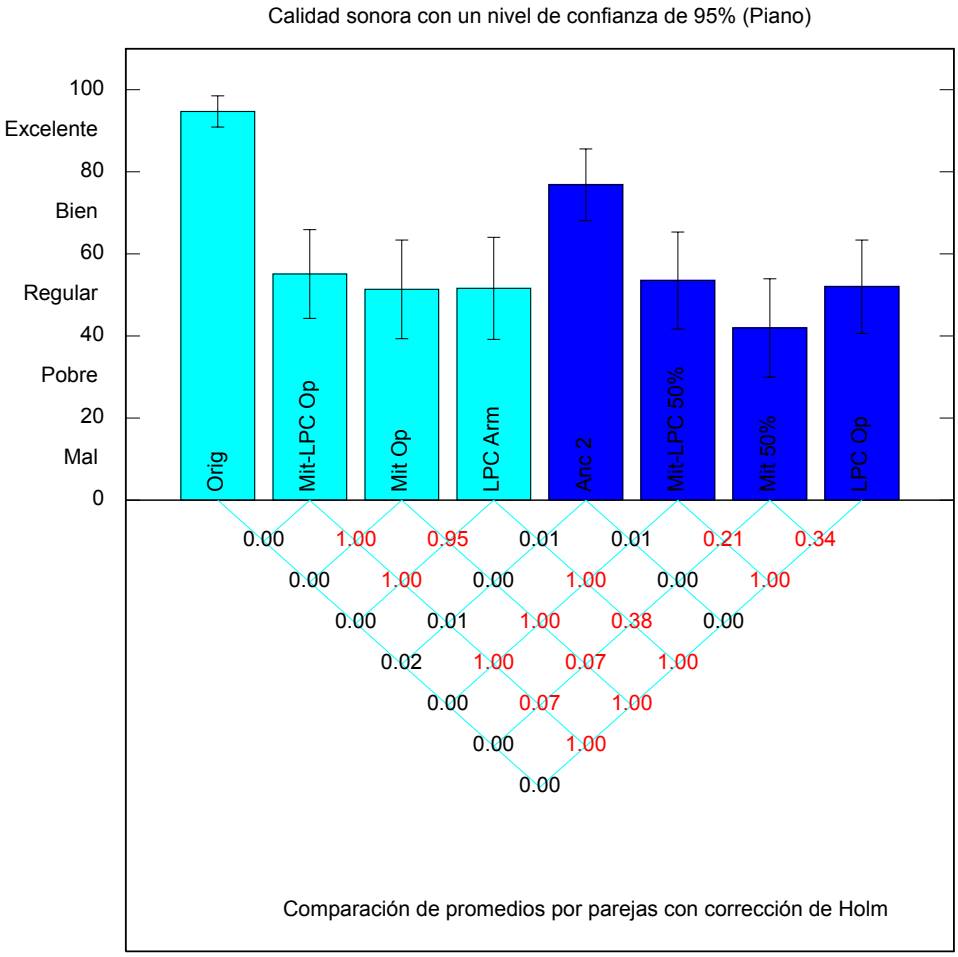


Figura 6.10: Medias de calidad sonora con intervalos de confianza y comparación de promedios por pareja (Piano)

Capítulo 7

Conclusiones

En este trabajo se ha propuesto un nuevo modelo de síntesis de sonido que combina la síntesis por segmentación de onda de Mitsuhashi y la predicción lineal. Se ha demostrado que este método puede ser implementado en una computadora personal y puede ser utilizado en tiempo real durante su fase de síntesis. Algunas conclusiones importantes se han obtenido a partir de las pruebas auditivas de audición. La síntesis de Mitsuhashi introduce distorsiones audibles. Estas distorsiones influyen de manera importante en el juicio de calidad sonora dado por una persona. Quizás, esta sea la razón por la que la síntesis de Mitsuhashi no se popularizó. A pesar de que Mitsuhashi compara su modelo de síntesis con la síntesis aditiva, la síntesis aditiva supera a la de Mitsuhashi en cuanto a su calidad sonora. Con el modelo híbrido *Mitsuhasi-LPC* que se propone de manera original en este trabajo, se puede mejorar la calidad sonora de la síntesis de Mitsuhashi para sintetizar un cierto tipo de instrumentos. El éxito en el modelado de un instrumento musical por medio de un determinado algoritmo de síntesis depende, en gran parte, de las características propias de cada instrumento.

Instrumentos continuos no ruidosos

Los instrumentos continuos como el corno, clarinete, oboe y el saxofón, comparten ciertas características, entre ellas: son instrumentos de aliento con un mecanismo de caña, parciales armónicas, alto contenido de armónicos agudos, y con poco ruido producido por su mecanismo de excitación. Estos instrumentos pueden ser sintetizados utilizando el sintetizador híbrido *Mitsuhasi-LPC* propuesto en este trabajo, logrando la mayor calidad sonora

en comparación con los otros métodos.

Instrumentos continuos ruidosos

La síntesis de instrumentos continuos como la flauta y el violín, cuyo sistema de excitación produce ruido, no se benefician tanto por el modelo híbrido. Esto se debe a que el ruido producido por el mecanismo de excitación dificulta el cálculo de los parámetros en la fase de análisis. Una posible solución a este problema consiste en extraer el ruido y las componentes tonales de la señal de audio y realizar el análisis y la síntesis por separado. Para ello se pueden utilizar las técnicas que se usan en la síntesis paramétrica [54, 78–84].

Instrumentos impulsivos

Los instrumentos impulsivos como la guitarra y el piano no son buenos candidatos para ser sintetizados con el modelo híbrido *Mitsuhasi-LPC*. Esto se debe a que las distorsiones producidas por el algoritmo de síntesis son más notorias y a que la amplitud del tono producido por estos instrumentos decae rápidamente. Para reducir las distorsiones se propone mejorar la estabilidad y precisión numérica de los filtros LPC mediante del uso de la descomposición de Cholesky en lugar de la recursión de Levinson. Para reducir la presencia de ruido de fondo después del ataque, se propone utilizar señales de corta duración y extrapolar los parámetros para sintetizar la última parte del tono. Según los resultados de las pruebas auditivas resulta conveniente utilizar la predicción lineal para sintetizar sonidos como la guitarra y el piano, pero se necesita mejorar la estabilidad y precisión numérica de los filtros LPC.

7.1. Aplicaciones

Los tres modelos de síntesis presentados en este trabajo pueden ser utilizados en la creación de instrumentos para su utilización en la música por computadora. Estos algoritmos son rápidos y ocupan pocos recursos de cómputo. Esto los hace viables para su implementación en ambientes o dispositivos con pocos recursos. En el ámbito de la música electrónica, estos algoritmos se pueden utilizar para realizar metamorfosis de instrumentos musicales por medio de la interpolación de sus diferentes parámetros. Por su representación gráfica, la síntesis de Mitsuhashi se puede utilizar para editar instrumentos por medio de una interfaz gráfica interactiva. También puede servir

para hacer modificaciones tímbricas de un sonido en tiempo real mediante el uso de controladores gráficos.

7.2. Trabajo futuro

- Los algoritmos de síntesis presentados en este trabajo están desarrollados de forma modular, es decir que se pueden conectar unos con otros de diferentes maneras. Esto nos da muchísimas posibilidades. En un futuro, será necesario experimentar más con los parámetros y combinaciones de estos modelos. Una vez clasificados, se necesitará realizar más pruebas de audición. En este trabajo, se utilizaron solamente las combinaciones más simples.
- Mejorar la estabilidad y precisión numérica de los filtros LPC.
- Desarrollar la interfaz gráfica de edición de la síntesis de Mitsuhashi.
- Implementar objetos en Pd para realizar metamorfosis de instrumentos musicales.
- Realizar más pruebas de audición y adaptar la metodología MUSHRA para lograr resultados menos dispersos.
- Implementar algoritmos de separación de ruido y componentes tonales.

Apéndice A

Gráficos de cajas

Los resultados de las pruebas auditivas de audición se presentan en forma de gráficos de cajas. La línea inferior de la caja representa el percentil 25 (primer cuartil), la línea superior de la caja representa el percentil 75 (tercer cuartil) y la línea al interior de la caja representa el percentil 50 (segundo cuartil). El símbolo + representa la media. El rango intercuartílico (RIC) es la distancia entre el primer y el tercer cuartil, es decir que el 50 % de los datos se encuentra dentro de la caja. Los bigotes representan los valores mínimos y máximos de la variable. Los asteriscos representan valores atípicos y los círculos representan valores más atípicos que se extienden 1.5 RIC y 3 RIC más allá del primer y tercer cuartiles respectivamente.

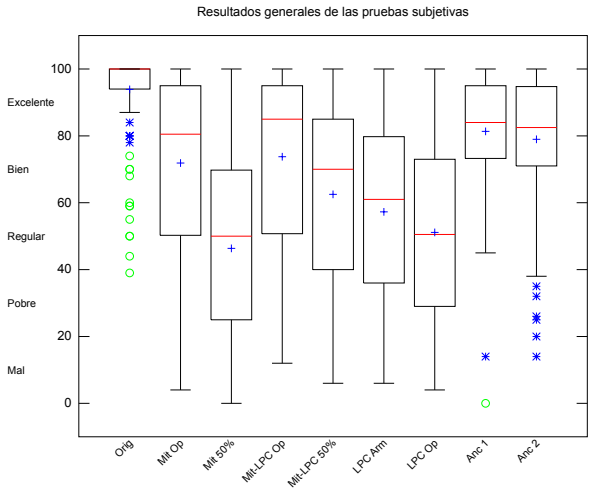


Figura A.1: Resultados generales de las pruebas subjetivas de audición.

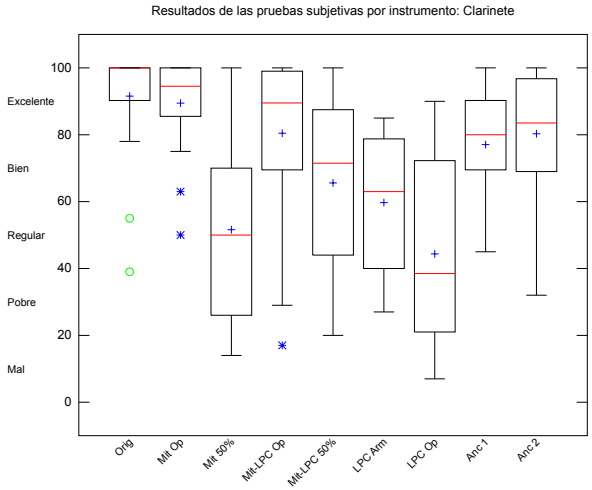


Figura A.2: Resultados de las pruebas subjetivas de audición para el tono de clarinete.

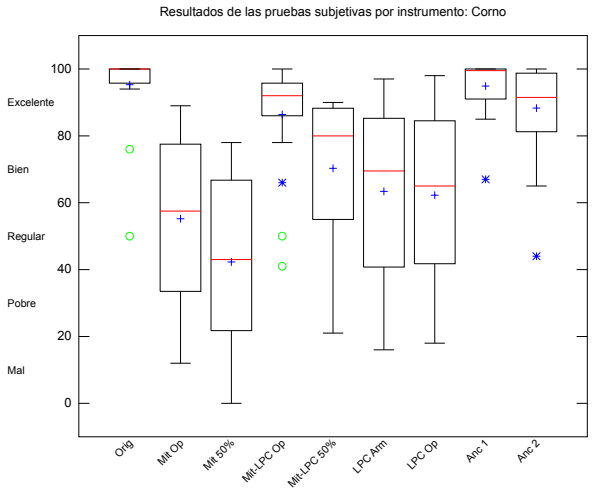


Figura A.3: Resultados de las pruebas subjetivas de audición para el tono de corno.

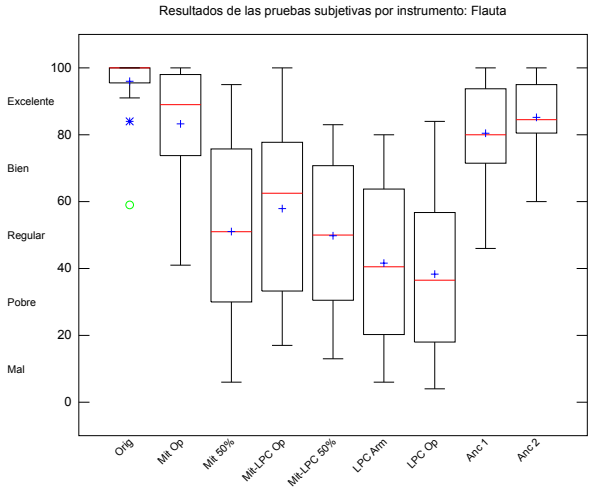


Figura A.4: Resultados de las pruebas subjetivas de audición para el tono de flauta.

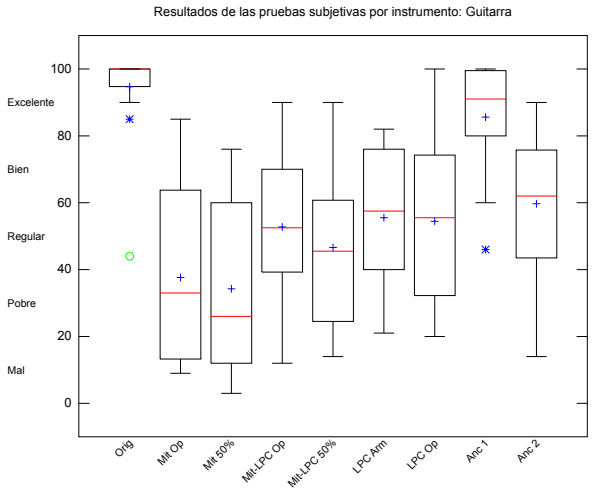


Figura A.5: Resultados de las pruebas subjetivas de audición para el tono de guitarra.

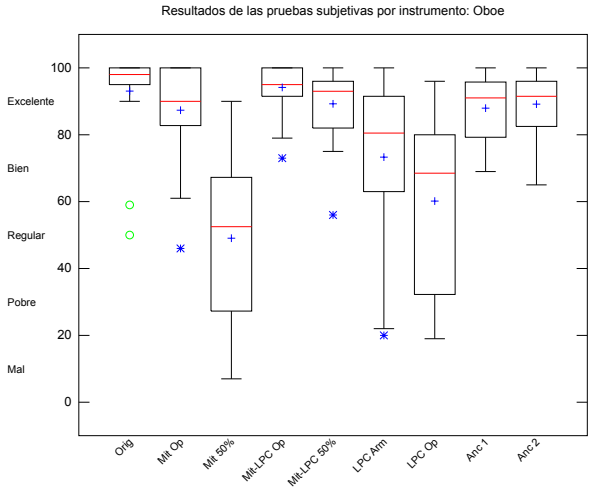


Figura A.6: Resultados de las pruebas subjetivas de audición para el tono de oboe.

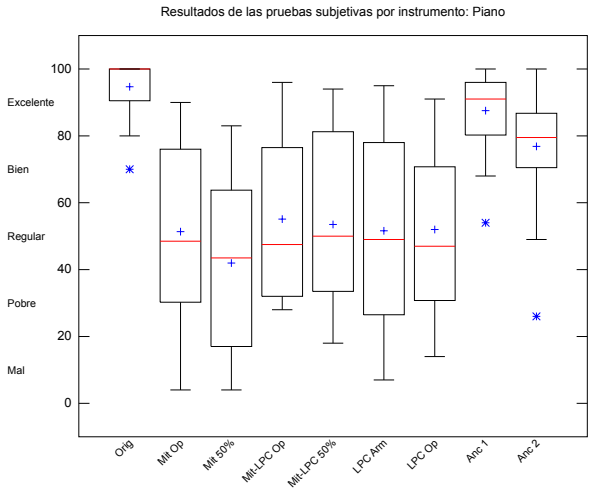


Figura A.7: Resultados de las pruebas subjetivas de audición para el tono de piano.

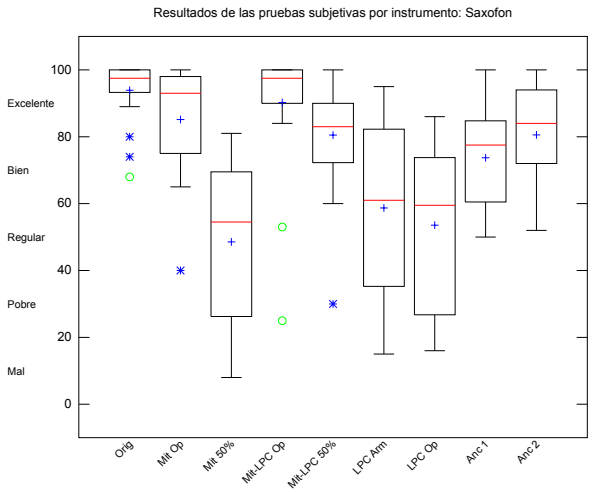


Figura A.8: Resultados de las pruebas subjetivas de audición para el tono de saxofón.

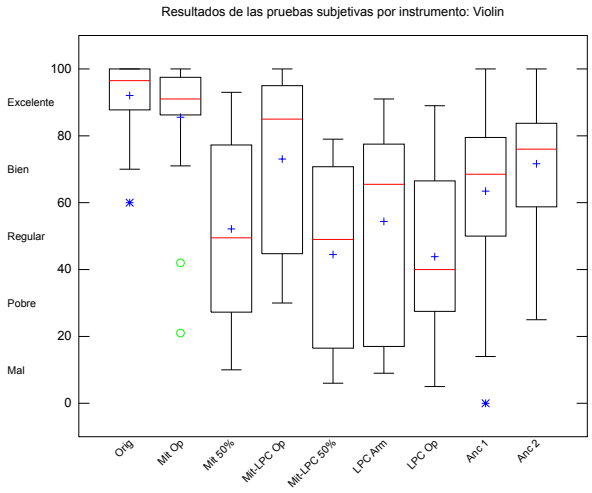


Figura A.9: Resultados de las pruebas subjetivas de audición para el tono de violín.

Apéndice B

Funciones interpoladoras de Mitsuhashi

Estas funciones interpoladoras están explicadas en el artículo de Mitsuhashi [1]. Se presentan aquí de manera simplificada. La serie de Fourier de la función periódica 2.1 está representada por

$$g(x) = \sum_{-\infty}^{\infty} \alpha_k e^{jkx} \tag{B.1}$$

Los coeficientes de Fourier α_k de la función segmentada $g(x)$ están representados por:

$$\alpha_k = b_k \sum_{n=0}^{N-1} y_n e^{-j(2\pi/n)nk} \tag{B.2}$$

Para obtener los coeficientes α_k de Fourier para cada tipo de interpolación, hay que substituir la variable b_k de la ecuación B.2 por la expresión correspondiente según la función de interpolación.

Interpolación constante

- Función interpoladora:

$$f(x) = \begin{cases} 0 & \text{para } 0 \leq x \leq 0.5 \\ 1 & \text{para } 0.5 < x \leq 1 \end{cases} \tag{B.3}$$

- Variable b_k:

$$b_k = \begin{cases} \frac{1}{N} & \text{para } k = 0 \\ \frac{\sin(\pi k/N)}{\pi k} & \text{para } k \neq 0 \end{cases} \tag{B.4}$$

.

Interpolación lineal

- Función interpoladora:

$$f(x) = x \tag{B.5}$$

- Variable b_k:

$$b_k = \begin{cases} \frac{1}{N} & \text{para } k = 0 \\ \frac{4N \sin^2(\pi k/N)}{(2\pi k)^2} & \text{para } k \neq 0 \end{cases} \tag{B.6}$$

Interpolación medio coseno

- Función interpoladora:

$$f(x) = \frac{1 - \cos(\pi x)}{2} \tag{B.7}$$

- Variable b_k:

$$b_k = \begin{cases} \frac{1}{N} & \text{para } k = 0 \\ \frac{\sin(2\pi k/N)}{2\pi k[1-(2k/N)^2]} & \text{para } k \neq 0, 2k \neq \pm N \\ \frac{1}{2N} & \text{para } 2k = \pm N \end{cases} \tag{B.8}$$

Interpolación polinomial

Para la interpolación polinomial se puede utilizar un polinomio $f(x)$ de grado l, siempre y cuando cumpla con la condición de que $f(0) = 0$ y $f(1) = 1$. Un caso particular es el del polinomio $f(x) = 3x^2 - 2x^3$.

- Función interpoladora:

$$f(x) = 3x^2 - 2x^3 \tag{B.9}$$

- Variable b_k:

$$b_k = \begin{cases} \frac{1}{N} & \text{para } k = 0 \\ \frac{12N^2}{(2\pi k)^3}\left[\frac{2N\sin^2(\pi k/N)}{\pi k} - \sin\left(\frac{2\pi k}{N}\right)\right] & \text{para } x \neq 0 \end{cases} \tag{B.10}$$

- Función interpoladora: cualquier polinomio $f(x)$ de grado l que cumpla con la condición $f(0) = 0$ y $f(1) = 1$.

- Variable b_k:

$$b_k = \begin{cases} \frac{1}{N} & \text{para } k = 0 \\ \frac{j}{2\pi k}\left[A_k\left(1 - e^{j(2\pi/N)k}\right) + B_k\left(1 - e^{-j(2\pi/N)k}\right)\right] & \text{para } k \neq 0 \end{cases} \tag{B.11}$$

  donde A_k y B_k están dados por

$$A_k = \sum_{m=0}^{l} \frac{f^{(m)}(0)}{(j2\pi k/N)^m} \tag{B.12}$$

$$B_k = \sum_{m=0}^{l} \frac{f^{(m)}(1)}{(j2\pi k/N)^m} \tag{B.13}$$

  y donde $f^{(m)}$ representa la derivada de orden m de la función interpoladora $f(x)$.

Apéndice C

Algoritmos y código fuente

C.1. Algoritmo de boersma

En el listado C.1 se muestra el código fuente en Octave para el algoritmo boersma. Los listados C.2 y C.3 contienen el código de la interpolación sinc y el algoritmo brent respectivamente.

Listado C.1: boersma.m

```
function [Tr, r] = boersma(x, Ta, Tb, tol)
% Finds the fundamental frequency of signal x
% using Boersma's algorithm
%
% x      Signal to be analysed
% Ta     Period of Highest candidate (in samples)
% Tb     Period of Lowest  candidtae (in samples)
% tol    Tolerance
%
% Tr     Period of the fundamental frequency (in samples)
% r      Harmonic Strength (A number between 0 an 1)

wSize = length(x);
n=0:wSize-1;

% Substract local average
x=x-mean(x);

% Apply window
w=hanning(length(x));
xw=x.*w;

% Append half a window length with zeroes
endZeros = 2^ceil(log(wSize*1.5)/log(2));
xw(wSize:endZeros)=0;
n=0:endZeros-1;
```

```
% Calculate autocorrelation using FFT
Rxw=real(ifft(abs(fft(xw)).^2));
rxw=Rxw/Rxw(1);

% Calculate autocorrelation of window
w(wSize:endZeros)=0;
Rw=real(ifft(abs(fft(w)).^2));
rw=Rw/Rw(1);

% Devide by the autocorrelation of the window
rx = rxw./rw;

% Find peak
[m im]=max(rx(Ta:Tb));
im = im + Ta - 1;

% Accurate estimation of peak using sinc
% interpolation and Brent algorithm
w2 = shift(w, -floor(wSize/2) + im);
rxw2=rx.*w2;

% Apply Brent algorithm
ax = im-1;
bx = im;
cx = im+1;
f = @(x)(-sincInterp(rxw2,x));
[xmin, ymin] = brent(ax, bx, cx, f, tol);
xmin = xmin - 1;
ymin = ymin * -1;
Tr = xmin;
r = ymin;
```

Listado C.2: sincInterp.m

```
function y = sincInterp(x,t)
% function y = sincInterp(x,t)
%
% Computes sinc interpolation using data from vector x
% the function is evaluated at time t
%
% x     vector data
% t     time

m = (1:length(x))';

for i=1:length(t)
   y(i) = sum(x.*sinc(t(i)-m));
end
```

Listado C.3: brent.m

```
function [xmin, ymin] = brent(ax, bx, cx, f, tol);
% Brent algorithm From Numerical Recipes by Press et all. 1992
% http://www.fizyka.umk.pl/nrbook/bookcpdf.html
%
% Given a function f, and given a bracketing triplet of
```

```
% abscissas ax, bx, cx (such that bx is between ax and cx,
% and f(bx) is less than both f(ax) and f(cx)),
% this routine isolates the minimum to a fractional precision
% of about tol using Brent's method. The abscissa of the minimum
% is returned as xmin, and the minimum function value is
% returned as ymin, the returned function value.

ITMAX = 100;
CGOLD = 0.3819660;
ZEPS = 1.0e-10;

d = 0;
if ax < cx
  a = ax;
else
  a = cx;
end

if ax > cx
  b = ax;
else
  b = cx;
end

x=w=v=bx;
fw=fv=fx=feval( f, x );

for iter=1:ITMAX
  xm=0.5*(a+b);
  tol2=2.0*(tol1=tol*abs(x)+ZEPS);
  if (abs(x-xm) <= (tol2-0.5*(b-a)))
    xmin=x;
    ymin=fx;
    return;
  end

  if (abs(e) > tol1)
    r=(x-w)*(fx-fv);
    q=(x-v)*(fx-fw);
    p=(x-v)*q-(x-w)*r;
    q=2.0*(q-r);

    if (q > 0.0)
      p = -p;
    end

    q=abs(q);
    etemp=e;
    e=d;

    if (abs(p) >= abs(0.5*q*etemp) || p <= q*(a-x) ||
        p >= q*(b-x))
      if x>= xm
        d=CGOLD*(e=a-x);
      else
        d=CGOLD*(e=b-x);
      end
```

```
    else
      d=p/q;
      u=x+d;
      if (u-a < tol2 || b-u < tol2)
        d=SIGN(tol1,xm-x);
      end
    end

  else
      if x >= xm
        d=CGOLD*(e=a-x);
      else
        d=CGOLD*(e=b-x);
      end
  end

  if abs(d) >= tol1
    u = x + d;
  else
    u = x+SIGN(tol1,d);
  end
  fu = feval(f,u);
  if (fu <= fx)
    if (u >= x)
      a=x;
    else
      b=x;
    end
    v=w; w=x; x=u;
    fv=fw; fw=fx; fx=fu;
  else
    if (u < x)
      a=u;
    else
      b=u;
    end

    if (fu <= fw || w == x)
       v=w;
       w=u;
       fv=fw;
       fw=fu;
    elseif (fu <= fv || v == x || v == w)
       v=u;
       fv=fu;
    end
  end

end
```

C.2. Algoritmo de reproducción de la síntesis de Mitsuhashi

El algoritmo de reproducción, sin tomar en cuenta la reproducción cíclica, es como sigue:

1. Se inicializan a cero dos contadores, sample y phase. El contador sample sirve para llevar el conteo del número de muestra actual y phase guarda la posición actual dentro de un ciclo de forma de onda definido por los nodos y[n] y la función interpoladora $f(x)$. La variable phase puede tomar valores entre 0 y 2π, es decir, $0 \leq phase < 2\pi$. La variable sample sólo toma valores enteros positivos.

2. Cuando la variable phase ha llegado o sobrepasado un ciclo entero, ($phase \geq 2\pi$), se actualizan las ordenadas y[n] de los nodos por medio de interpolación lineal y la fase vuelve a comenzar desde 0, ($phase = phase - 2\pi$).

3. Si el número de muestra actual coincide o sobrepasa el momento correspondiente a la siguiente forma de onda ($sample \geq timeStamp$), se cargarán los valores de la forma de onda siguiente. Estos nuevos valores influirán en el cálculo de la interpolación lineal durante los ciclos subsiguientes.

4. Se calcula el valor de la muestra actual por medio de la ecuación 2.1.

5. Se incrementan los contadores y comienza una nueva iteración.

Listado C.4: mitsu.c

```
typedef struct _ifunction {
    double (* f )( double x);
    double (* a )( unsigned int k, unsigned int N);
    char id;
} ifunction;

typedef struct _MitsuData {
    double * y; // ordinates
    double * breakPoints;
    unsigned int * timeStamps;
    double * periods;
    double period;
    double phaseIncrement;
    unsigned int nPoints;
    unsigned int nWaveForms;
```

```
    ifunction * f;
    unsigned int waveform;
    unsigned int sample;
    double phase;
    unsigned int loopA;
    unsigned int loopB;
    char noteOn;
    char backward;
} MitsuData;


double MitsuData_getSample( MitsuData * data ) {
    double rv;
    unsigned int n;
    if ( data -> waveform >= data -> nWaveForms) return 0;
    if ( data -> sample == 0 ) {
        data -> phaseIncrement = (2.0 * M_PI) / data -> period;
        MitsuData_updateOrdinates(data);
    }

    n = (data -> phase * data -> nPoints) / (2.0 * M_PI);
    rv = ( (data -> y[n+1]) - (data -> y[n]) ) *
         ( data -> f -> f( (data -> nPoints/(2.0 * M_PI) ) *
         (data -> phase) - n ) ) + data -> y[n];

    data -> phase += data -> phaseIncrement;
    data -> sample ++;

    if ( (data -> sample) >= (data -> timeStamps[data -> waveform + 1]) ) {
        data -> waveform ++;
    }

    if ( (data -> phase) >=  2.0 * M_PI ) {
        data -> phase = (data -> phase) - 2.0 * M_PI;
        MitsuData_updateOrdinates(data);
    }

    return rv;
}

void MitsuData_updateOrdinates( MitsuData * data ) {
    int n;
    if ( (data -> waveform + 1) >= data -> nWaveForms) {
        for ( n = 0 ; n <= (data -> nPoints) ; n++ ) {
            data -> y[n] = 0.0;
        }
        return;
    }

    for ( n = 0 ; n <= (data -> nPoints ) ; n++ ) {
       data -> y[n] = linearIntepolation(
                data -> sample, // x
                data -> timeStamps[data -> waveform], // x0
                data -> breakPoints[ (data -> waveform) *
                        (data -> nPoints) + n], // y0
                data -> timeStamps[data -> waveform + 1 ], // x1
                data -> breakPoints[( data -> waveform + 1) *
```

```
                    (data -> nPoints) + n] ); // y1
    }
}
```

En el siguiente listado se incluye la reproducción cíclica. Se agregaron algunas líneas a la función MitsuData_getSample() del listado C.4. La variable backward es una bandera que indica cuándo se está retrocediendo dentro del segmento de reproducción cíclica. La variable waveform se incrementa ++ o decrementa −− según el sentido al que se está avanzando al interior del segmento de reproducción cíclica.

Listado C.5: mitsu.c

```
double MitsuData_getSample( MitsuData * data ) {
    double rv;
    unsigned int n;
    if ( data -> waveform >= data -> nWaveForms) return 0;
    if ( data -> sample == 0 ) {
        data -> phaseIncrement = (2.0 * M_PI) / data -> period;
        MitsuData_updateOrdinates(data);
    }

    n = (data -> phase * data -> nPoints) / (2.0 * M_PI);
    rv = ( (data -> y[n+1]) - (data -> y[n]) ) *
         ( data -> f -> f( (data -> nPoints/(2.0 * M_PI) ) *
         (data -> phase) - n ) ) + data -> y[n];

    data -> phase += data -> phaseIncrement;
    ( data -> backward ? data -> sample -- : data -> sample ++);

    if ( (data -> sample) >=
         (data -> timeStamps[data -> waveform + 1]) ) {
        data -> waveform ++;
    }

    if ( (data -> sample) <=
         (data -> timeStamps[data -> waveform]) ) {
        data -> waveform --;
    }


    if ( (data -> phase) >=  2.0 * M_PI ) {
        if (data -> sample < data -> loopA ) {
            data -> backward = 0;
        }

        if ( data -> sample > data -> loopB ) {
            data -> backward = 1;
        }

        if ( data -> loopB == 0 ) {
            data -> backward = 0;
        }
```

```
        data -> phase = (data -> phase) - 2.0 * M_PI;
        MitsuData_updateOrdinates(data);
    }

    return rv;
}
```

C.3. Oscilador senoidal simple

El listado C.6 es un ejemplo sencillo, en lenguaje C, de como se implementa un oscilador senoidal. La función note2freq() calcula la frecuencia fundamental a partir del valor MIDI. La función vel2amp() calcula la amplitud a partir de la velocidad. La función getSample() calcula el valor de cada muestra, observe el uso de las variables phase y phaseIncrement.

Listado C.6: midi2sin.c

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


double fs = 44100; // Hz
double phase = 0.0;
double fq;
double amp;
double phaseIncrement;

double getSample (){
  double rv = amp * sin(phase);
  phase += phaseIncrement;
  if ( phase >= 2.0 * M_PI) {
    phase = phase - 2.0 * M_PI;
  }
  return rv;
}

double note2freq( unsigned char midiNote ) {
  return 440.0 * pow( 2.0, (midiNote - 69)/12.0 );
}

double vel2amp( unsigned char velocity ) {
  return velocity/127.0;
}

void printHelp() {
  fprintf(stderr, "usage: midi2sin note vel dur\n");
  fprintf(stderr, "ej: sine 60 127 1000\n");
}

int main(int argc, char** argv) {
```

```
  if (argc != 4 ) {printHelp(); exit(0);}
  fq = note2freq(atoi(argv[1]));
  phaseIncrement = (2.0 * M_PI * fq )/fs;
  amp = vel2amp(atoi(argv[2]));
  int nSamples = (fs * atoi(argv[3]) ) / 1000;
  int i;
  for ( i = 0; i < nSamples; i ++ ) {
    printf( "%f\n", getSample() );
  }
}
```

C.4. Algoritmo de Durbin

En el listado C.7 se presenta un pseudocódigo para el algoritmo de Durbin. El código es una traducción casi directa de las ecuaciones 3.8-3.13. Sin embargo, una pequeña modificación es necesaria a la hora de la actualización de los valores a_j de la ecuación 3.11. Las líneas 13 a 17 del listado C.7 corresponden a la ecuación 3.11. Si no existiera la variable temporal temp, el valor de la variable a[j] se borraría y no se podría usar en iteraciones posteriores. Este problema se resuelve calculando el elemento a[j] y el elemento a[i−j] que equivale a a[h], en la misma iteración j. La variable temp se utiliza para guardar el valor de a[j] de la iteración anterior. En el listado C.7, los arreglos a y k, corresponden a los coeficientes del predictor y a los coeficientes de reflexión respectivamente. La variable E representa el error mínimo total en cada iteración i.

Listado C.7: durbin.pseudo

```
1  a[0] = 1;
2  E = R[0];
3  for ( i = 1 ; i <= p ; i++ ) {
4       sum = 0.0;
5
6       for ( j = 1 ; j < i ; j++ ) {
7            sum = sum + a[j] * R[i-j];
8       }
9
10      k[i] = (-1.0 * (R[i] + sum) ) / E;
11      a[i] = k[i];
12
13      for ( j = 1, h = i - 1 ; j <= h ; j++, h-- ) {
14           temp = a[j];
15           a[j] = a[j] + k[i] * a[h];
16           a[h] = a[h] + k[i] * temp;
17      }
18
19      E = ( 1.0 - k[i-1] * k[i] ) * E;
20 }
```

C.5. Predicción Lineal en Octave

Listado C.8: lpc.m

```
function [a, g2, ref] = lpc(sn, p)
%[a, g2, e] = flpc(sn, p)
%
%Calculates Linear prediction Coefficients
%
%a      Predictor coefficients
%g2     Gain factor g^2
%
%sn     Signal to be analysed
%p      Filter order

%N must be at least 4 cicles in length
N = length (sn);
w = hanning(N);
snw = sn .* w;

%periodic autocorrelation
R = real(ifft(abs(fft(snw)).^2));

%one-term LPC
[a, g2, ref] = levinson (R , p);
```

C.6. Extracción de la envolvente espectral por medio de LPC

Listado C.9: SpectralEnvelopeLPC.m

```
function [a v] = SpectralEnvelopeLPC( x, fc, fs, method )
%a = SpectralEnvelopeLPC( sig )
%
%a        LPC all-pole filter coeficients
%x        Signal to be analyzed
%v        Normalize minimum quadratic error
%method   i - Maximum likelihood
%         d - First differences

%Pitch detection
Tb = round(fs/(fc*(3/4)));
Ta = round(fs/(fc*(4/3)));
tol = 0.0001;
N = length(x);
[Tr, r] = boersma(x, Ta, Tb, tol);
xw = x.*hanning(N);
R = real(ifft(abs(fft(xw)).^2));
p = round(Tr);
```

```
%LPC
[a, v, ref] = flevinson (R, p);

%Maximum likelihood
pn = 1:p;
Ne = N * 0.375;
I=log(v) + (2*pn)/Ne;
pa = round(Tr/15);
pb = round(Tr/2);
[mn imn] = min(I(pa:pb));
imn = imn + pa - 1;

% First difference
logV = log(v);
diffLogV = diff(logV);
[w iw]=min(diffLogV(pa:pb));
iw = iw + pa - 1;

for i=iw:p-2
  if (abs(diffLogV(i)) < abs(diffLogV(i+1)) )
    break;
  end
end
i=i-1;

if method == 'i'
  a=a(1:imn);
else
  a=a(1:i);
end
  v=v(length(a));
```

C.7. Implementación del filtro latice

El listado C.10 contiene el pseudocódigo para la implementación de un filtro lattice.

Listado C.10: LatticeFilter.pseudo

```
// N is the order of the predictor
// x and y have N + 1 elements
// k has N reflection coefficients
// Set all elements of x and y to 0
// in[m] is the m-th sample of the input signal
// out[m] is the m-th sample of the filtered signal

x[N]=in[m];

for(i = N-1 ; i >= 0 ; i--){
    x[i] = x[i+1] - k[i]*y[i];
    y[i+1] = y[i] + k[i]*x[i];
}
```

```
y[0]=x[0];
out[m]=x[0];
```

C.8. Cálculo de los intervalos de confianza

Este es el código en Octave [9] que fue utilizado para calcular los promedios de calidad sonora y los intervalos de confianza. El libro *Online Statistics Education: A Multimedia Course of Study* [76] fue utilizado como referencia.

Listado C.11: meanConfidenceInterval.m

```
function [mdata err] = meanConfidenceInterval(data, level)
%function [mdata err] = meanConfidenceInterval(data, level)
%
%Calculates the mean m and confidence intervals err
%based on student distribution
%
%data     data, each set arranged in columns
%level    significance level eg. 95%
%
%mdata    mean
%err      confidence interval

if nargin() < 2
  level = 95;
end

[nr nc] = size(data);
N = nr;
mdata = mean(data);  %mean
vdata = var(data);   %variance
sdata = std(data);   %standard deviation

sm = sdata/sqrt(N); %standard error
df = N - 1; %degrees of freedom
t_level = abs(tinv( ( 1 - ( level / 100 ) ) / 2, df)); %Student t
err = t_level*sm;   % error +-

% plot
if nargout() == 0
  bar(mdata); hold on;
  h2 = errorbar(mdata, err);
  set(h2,"linestyle","none");
  hold off;
end
```

C.9. Cálculo de comparaciones de promedios por parejas

Este es el código en Octave [9] que fue utilizado para calcular los valores de probabilidad de las comparaciones de promedios por parejas. Las fuentes utilizadas para el cálculo fueron: el libro *Online Statistics Education: A Multimedia Course of Study* [76] y los articulos de Simens [77] y Wright [75].

Listado C.12: pairwiseComparisonsCorrelated.m

```
function [pval mdata aIdx bIdx nonSig] =
                pairwiseComparisonsCorrelated (data, correctionType)
%function [] = pairwiseComparisonsCorrelated (data, correctionType)
%
%Performs pairwise comparisons between means for correlated samples
%
%correctionType  "Holm"  (default)
%                "Bonferroni"
%                "Simens"
%                "none"
%
%pval             provability values (ordered)
%mdata            means
%aIdx             Index of the mean being compared  mdata(aIdx)
%                 vs mdata(bIdx)
%bIdx             Index of the other mean being compared
%nonSig           Index at which nonsignificant values start
%                 ej. pval(nonSig:length(pval)) are nonsignificant
%                 at the 0.05 level no matter if pval(i)<0.05
%                 This criteria applies only to the Holm correction
%
%data             data(soundQuality, synthAlgorithm)


[nr nc] = size(data);
mdata = mean(data);

if nargin() < 2
  correctionType = "Holm";
end

%Calcular valores de probabilidad sin correccion (t-test)
for a = 1:nc-1
  for b = a + 1:nc
    deltaAB = data(:,a)-data(:,b);
    [PVAL, T, DF] = t_test (deltaAB, 0);
    p(b-1,a) = PVAL;
  end
end

oneColumn_p = reshape(p',numel(p));
[sorted_p pIdx] = sort(oneColumn_p);

%Truncar ceros (Triangulo superior de la matriz)
```

```
n = nc - 2;
n = (1+n)*(n/2);
pIdx = pIdx(n+1:numel(pIdx));
sorted_p = sorted_p(n+1:numel(sorted_p)); % Truncar ceros
n = length(sorted_p);

% Correccions
if strcmp('Holm', correctionType)
  i = (1:n)';
  holm = (n - i + 1).*sorted_p;
  holm(find(holm>1))=1;   % Evitar valores mayores que cero
  pval = holm;
  nonSig = min(find(pval>=0.05));
elseif  strcmp('Bonferroni', correctionType )
  bonf = sorted_p * n;
  bonf(find(bonf>1))=1;   % Evitar valores mayores que cero
  pval = bonf;
elseif strcmp('Simens', correctionType )
  j = (1:n)';
  simens = (sorted_p * n)/j;
  pval = sorted_p;
end

% Calcular indices
np = length(p);
c = mod(pIdx-1, np)+1;
r = ceil(pIdx./np);
aIdx = c;
bIdx = r + 1;
```

Apéndice D

Instrucciones durante la sesión de entrenamiento

Fase de entrenamiento o familiarización

El primer paso para las pruebas auditivas es familiarizarse con el procedimiento de evaluación. Esta fase es llamada fase de entrenamiento y precede a la fase de evaluación formal.

El propósito de la fase de entrenamiento es procurar que usted, el evaluador, logre los siguientes objetivos:

- Parte A: Familiarizarse con el material sonoro así como sus rangos de volumen.
- Parte B: Aprender a utilizar el equipo de evaluación y la escala de calificaciones.

En la parte A de la fase de entrenamiento, usted va a escuchar todos los sonidos que han sido seleccionados para la evaluación, con el objetivo de ilustrar el rango completo de posibilidades. Los ejemplos sonoros que va a escuchar son más o menos críticos dependiendo del nivel de reducción de datos y otras condiciones utilizadas. Puede dar «click» en diferentes botones para escuchar diferentes muestras sonoras incluyendo los sonidos de referencia. De esta manera, usted podrá apreciar los diferentes niveles de calidad sonora de cada muestra sonora. Los ejemplos están ordenados de acuerdo a características comunes.

En la parte B de la fase de entrenamiento, usted aprenderá a usar el equipo de reproducción y evaluación que será utilizado para evaluar la calidad sonora de los algoritmos de síntesis.

Durante la fase de entrenamiento, usted aprenderá cómo interpretar auditivamente las características de los sonidos en función de la escala de calificaciones de forma personal. Está prohibido, durante la duración de la fase de entrenamiento, discutir o comentar con otros participantes su interpretación personal sobre la escala de calificaciones. Sin embargo, si se puede explicar a otros como escuchar las diferentes distorsiones producidas por el algoritmo de síntesis.

Las calificaciones obtenidas durante la fase de entrenamiento no serán tomadas en cuenta en las pruebas oficiales.

Fase de prueba de evaluación a ciegas

El propósito de la fase de evaluación es invitarlo a asignar calificaciones usando la escala de calidad de audio. Sus calificaciones deben reflejar su juicio subjetivo sobre el nivel de calidad sonora para cada muestra de audio presentada. Cada prueba contiene un número determinado de señales que serán calificadas. Cada señal tendrá una duración aproximada de 5 segundos. Usted debe escuchar el sonido de referencia y los demás sonidos de prueba dando «click» a los botones respectivos. Puede escuchar las señales en cualquier orden y tantas veces sea necesario. Utilice la barra desplazadora de cada señal para indicar su opinión sobre su calidad sonora. Cuando esté satisfecho con las calificaciones de todas las señales, usted deberá dar «click» al botón «register scores» de la pantalla.

Bibliografía

[1] Y. Mitsuhashi, "Piecewise interpolation technique for audio signal synthesis," *Jounal of the Audio Engineering Society*, vol. 30, no. 4, pp. 192–202, 1982.

[2] J. Makhoul, "Linear prediction: A tutorial review," in *Proceedings of the IEEE*, vol. 63, pp. 561–580, 1975.

[3] M. Puckette, "Pure data: Entorno gráfico de programación para procesamiento de audio y video," Mar. 2011. http://puredata.info/.

[4] ITURrecBS1534-1, "Method for the subjective assessment of intermediate quality level of coding systems," *ITU-R Recommendation BS.1534-1*, 2003.

[5] D. D. Gagné, "An application of linear prediction of speech techniques to music synthesis," in *AES 85th Convention*, Nov. 1988.

[6] A. D. Bernstein and E. D. Cooper, "The piecewise-linear technique of electronic music synthesis," *Jounal of the Audio Engineering Society*, vol. 24, no. 6, pp. 446–454, 1976.

[7] C. Roads, *The Computer Music Tutorial*. The MIT Press, feb 1996.

[8] J. W. Eaton, *GNU Octave Manual*. Network Theory Limited, 2002.

[9] J. W. Eaton, "Gnu octave: A high-level language, primarily intended for numerical computations," Aug. 2008. http://www.gnu.org/software/octave/.

[10] M. Puckette, *The Theory and Technique of Electronic Music*. World Scientific Publishing Co., 2007.

[11] Y. Mitsuhashi, "Musical sound synthesis by forward differences," *Jounal of the Audio Engineering Society*, vol. 30, no. 1/2, pp. 2–9, 1982.

[12] N. Collins, "Splinesynth: An interface to low-level digital audio," in *Proceedings of the Diderot Forum on Mathematics and Music, Vienna*, pp. 49–61, Research Fellow, Centre for Electronic Arts, Middlesex University, 1999.

[13] N. Collins, "Splinesynth2: Interpolating break-point sets to obtain sound transformations distinct from a cross-fade." Unpublished research report on the SplineSynth2 software, 2000.

[14] A. Spanias, T. Painter, and V. Atti, *Audio Signal Processing and Coding*. John Wiley & Sons, 2007.

[15] C. Roads, *Microsound*. The MIT Press, sep 2004.

[16] P. Boersma, "Accurate short-term analysis of the fundamental frequency and the harmonic to noise ratio of sampled sound," in *Institute of Phonetic Sciences, University of Amsterdam, Proceedings 17*, pp. 97–110, 1993.

[17] J. Wroblewski and A. Wieczorkowska, "Octave-error proof timbre-independent estimation of fundamental frequency of musical sounds," in *AES 116th Convention*, May 2004.

[18] W. J. Hess, *Pitch Determination of Speech Signals*. Springer, 1st ed., 1983.

[19] P. Boersma and D. Weenink, "Praat: doing phonetics by computer (version 5.0.30) [computer program]." Retrieved July 26, 2008, from http://www.praat.org/, 2008. http://www.praat.org/.

[20] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd ed., 1992.

[21] J. M. Clark, D. Luce, R. Abrams, H. Schlossberg, and J. Rome, "Preliminary experiments on the aural significance of parts of tones of orchestral instruments and on choral tones," *Jounal of the Audio Engineering Society*, vol. 11, pp. 45–54, Jan. 1963.

[22] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," tech. rep., IRCAM, Apr. 2004.

[23] J. Hajda, "A new model for segmenting the envelope of musical signals: The relative salience of steady state versus attack, revisited," in *AES 101st Convention*, Nov. 1996.

[24] D. Luce and J. M. Clark, "Duration of attack transients of nonpercussive orchestral instruments," *Jounal of the Audio Engineering Society*, vol. 13, pp. 194–199, July 1965.

[25] J. W. Beauchamp, "Synthesis by spectral amplitude and "brightness" matching of analyzed musical instrument tones," *Jounal of the Audio Engineering Society*, vol. 30, pp. 396–406, June 1982.

[26] M. J. Wright, *The Shape of an Instant: Measuring and Modeling Perceptual Attack Time With Probability Density Functions*. PhD thesis, Stanford University, Mar. 2008.

[27] A. S. Pena, "A theoretical approach to a generalized auditory model for audio coding and objective perceptual assessment," in *AES 94th Convention*, Jan. 1994.

[28] R. A. Kendall, ""the role of acoustic signal partitions in listener categorization of musical phrases"," *Music Perception*, vol. 4, no. 2, pp. 185–214, 1986.

[29] "ISO/IEC JTC1/SC29/WG11 MPEG, IS11172-3 (MPEG-1). information technology - coding of moving pictures and association ofr digital storage media at up to about 1.5 mbits/s, part 3: Audio," 1992.

[30] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, and M. Dietz, "ISO/IEC MPEG-2 advanced audio coding," *Jounal of the Audio Engineering Society*, vol. 45, no. 10, pp. 789–814, 1997.

[31] K. Brandenburg and M. Bosi, "ISO/IEC MPEG-2 advanced audio coding: Overview and applications," in *AES 103rd Convention*, Sept. 1997.

[32] M. Kahrs and K. Brandenburg, *Applications of digital signal processing to audio and acoustics*. Kluwer Academic Publishers, 1998.

[33] A. B. J. Horner, "Piecewise-linear approximation of additive synthesis envelopes: A comparison of various methods," *Computer Music Journal*, vol. 20, no. 2, pp. 72–95, 1996.

[34] J. Yuen and A. Horner, "Hybrid sampling-wavetable synthesis with genetic algorithms (p)," *Jounal of the Audio Engineering Society*, vol. 45, no. 5, pp. 316–330, 1997.

[35] S. Wun and A. Horner, "Evaluation of iterative methods for wavetable matching," *Jounal of the Audio Engineering Society*, vol. 53, no. 9, pp. 826–835, 2005.

[36] S. Wun and A. Horner, "A comparison between local search and genetic algorithm methods for wavetable matching," *Jounal of the Audio Engineering Society*, vol. 53, no. 4, pp. 314–325, 2005.

[37] C.-W. Wun and A. Horner, "Perceptual wavetable matching for synthesis of musical instrument tones," *Jounal of the Audio Engineering Society*, vol. 49, no. 4, pp. 250–262, 2001.

[38] A. Horner, "A simplified wavetable matching method using combinatorial basis spectra selection," *Jounal of the Audio Engineering Society*, vol. 49, no. 11, pp. 1060–1066, 2001.

[39] A. Horner, "Wavetable matching synthesis of dynamic instruments," in *AES 101st Convention*, Nov. 1996.

[40] A. Horner, J. Beauchamp, and L. Haken, "Methods for multiple wavetable synthesis of musical instrument tones," *Jounal of the Audio Engineering Society*, vol. 41, pp. 336–356, May 1993.

[41] A. Horner, "Wavetable matching synthesis of dynamic instruments with genetic algorithms," *Jounal of the Audio Engineering Society*, vol. 43, pp. 916–931, Nov. 1995.

[42] H. Fastl and E. Zwicker, *Psycho-Acoustics facts and models*. Springer, 3rd edition ed., 2007.

[43] J. G. Roederer, *The physics and pshychophysics of music*. Springer-Verlag, 1994.

[44] *Complete MIDI 1.0 Detailed Specification*, 1999/2008.

[45] P. Vaidyanathan, *The Theory of Linear Prediction*. Morgan and Claypool Publishers, 1st. ed., Feb. 2008.

[46] N. Levinson, "The wiener rms criterion in filter design and prediction," *J. Math. Phys.*, vol. 25, pp. 261–278, Jan. 1947.

[47] N. Weiner, *Extrapolation, interpolation and smoothing of stationary time series*. John Wiley & Sons, New York, 1949.

[48] N. Weiner and P. Massani, "The prediction theory of multivariate stochastic processes, part 1," *Acta Math.*, vol. 98, pp. 111–150, 1957.

[49] B. S. Atal and S. L. Hanauer, "Speech analysis and synthesis by linear prediction of the speech wave," *The Journal of the Acoustical Society of America*, vol. 50, no. 2B, 1971.

[50] F. Itakura and S. Saito, "A statistical method for estimation of speech spectral density and formant frequencies," *Trans IECE Jpn.*, vol. 53-A, pp. 36–43, 1970.

[51] L. R. Rabiner and R. W. Schafer, *Digital processing of speec signals*. Prentice-Hall, Englewood Cliffs, NJ, 1978.

[52] J. R. Deller, J. G. Proakis, and J. H. L. Hansen, *Discrete-time Processing of Speech Signals*. Macmillan, New York, 1993.

[53] B. Gold and N. Morgan, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. Wiley, first ed., jul 1999.

[54] A. C. den Brinker and F. Riera-Palou, "Pure linear prediction," in *AES 115th Convention*, Oct. 2003.

[55] J.-L. Garcia, P. Gournay, and R. Lefebvre, "Backward linear prediction for lossless coding of stereo audio," in *AES 116th Convention*, May 2004.

[56] F. Ghido, "Optimal quantized linear prediction coefficients for lossless audio compression - scalar quantization revisited," in *AES 120th Convention*, May 2006.

[57] M. R. Schroeder, *Computer Speech: Recognition, Compression, Synthesis*. Springer, 2nd ed., 2004.

[58] J. A. Moorer, "The use of linear prediction of speech in computer music applications," *Jounal of the Audio Engineering Society*, vol. 27, pp. 134–140, Mar. 1979.

[59] M. B. Sandler, "Auto regressive modelling and synthesis of acoustic instruments," in *AES 86th Convention*, Feb. 1989.

[60] M. B. Sandler, "New results in LPC synthesis of drums," in *AES 89th Convention*, Aug. 1990.

[61] P. Lansky, "Linear prediction: The hard, but interesting way to do things," in *The AES 5th International Conference: Music and Digital Technology*, Apr. 1987.

[62] U. Zölzer, ed., *DAFX, Digial Audio Efects*. John Wiley & Sons, Apr. 2005.

[63] GNU, "Octave-forge from http://octave.sourceforge.net/index.html," July 2009. Octave-Forge is a central location for the collaborative development of packages for GNU Octave.

[64] M. Athineos, "Autoregressive modeling of temporal envelopes," *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, vol. 55, Nov. 2007.

[65] J. Makhoul, "Stable and efficient lattice methods for linear prediction," *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, vol. 25, Oct. 1977.

[66] "Max from http://cycling74.com/." Entorno gráfico de programación para applicaciones de audio.

[67] B. W. Kernighan and D. M. Ritchie, *The C programming language*. Prentice Hall, 2nd ed., 1988.

[68] M. Frigo and S. G. Johnson, "Fftw from http://www.fftw.org," Aug. 2011. A C library for computing the discrete Fourier transform.

[69] E. de Castro Lopo, "libsndfile from http://www.mega-nerd.com/libsndfile," Aug. 2011. A C library for reading and writing files containing sampled sound.

[70] A. Farnell, *Designing Sound*. Applied Scientific Press Ltd., 1st ed., Sept. 2008.

[71] J. M. Zmölnig, "Howto write an external for puredata from http://iem.at/pd/externals-howto/," Aug. 2011.

[72] T. Grill, "Flext from http://puredata.info/members/thomas/flext/," Oct. 2009. A C++ layer for cross-platform development of Pd and Max/MSP objects.

[73] ITURrecBS1116-1, "Methods for the subjective assessment of small impairments in audio systems including multichannel sound systems," *ITU-R Recommendation BS.1116-1*, 1997.

[74] F. Opolko and J. Wapnick, "Mcgill university master samples," 2006.

[75] S. P. Wright, "Adjusted p-values for simultaneous inference," *Biometrics*, vol. 48, pp. 1005–1013, Dec. 1992.

[76] D. Lane, "Online statistics education: A multimedia course of study," 2010. (http://onlinestatbook.com/). Project Leader: David M. Lane, Rice University.

[77] R. J. Simens, "An improved bonferroni procedure for multiple tests of significance," *Biometrika*, vol. 73, no. 3, pp. 751–754, 1986.

[78] B. Edler, H. Purnhagen, and C. Ferekidis, "Asac-analysis/synthesis audio codec for very low-bit rates," in *AES 100 Convention*, Apr. 1996.

[79] B. Edler and H. Purnhagen, "Concepts for hybrid audio coding schemes based on parametric techniques," in *AES 105th Convention*, Sept. 1998.

[80] H. Purnhagen, B. Edler, and C. Ferekidis, "Object-based analysis/synthesis audio coder for very low bit rates," in *AES 104th Convention*, Apr. 1998.

[81] H. Purnhagen, "An overview of mpeg-4 audio version 2," in *AES 17th International Conference*, Aug. 1999.

[82] H. Purnhagen, "Advances in parametric audio coding," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Oct. 1999.

[83] H. Purnhagen, N. Meine, and B. Edler, "Speeding up HILN-MPEG-4 parametric audio encoding with reduced complexity," in *AES 109 Convention*, Aug. 2000.

[84] *Parametric Coding for High-Quality Audio*, Apr. 2002.

Índice alfabético