



UNIVERSIDAD
DE GRANADA

Facultad de Ciencias
Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicaciones

GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS

TRABAJO DE FIN DE GRADO

Análisis de Fourier y aplicación práctica en el tratamiento de señales de audio

Presentado por:
Javier Granados López

Tutor:
José Luis Gámez Ruiz
Departamento de Análisis Matemático
Diego Salas González
Departamento de Teoría de la Señal, Telemática y Telecomunicaciones

Curso académico 2023-2024

Análisis de Fourier y aplicación práctica en el tratamiento de señales de audio

Javier Granados López

Javier Granados López. *Análisis de Fourier y aplicación práctica en el tratamiento de señales de audio.*

Trabajo de Fin de Grado. Curso académico 2023-2024.

**Responsables de
tutorización**

José Luis Gámez Ruiz
Departamento de Análisis Matemático
Diego Salas González
*Departamento de Teoría de la Señal,
Telemática y Telecomunicaciones*

Grado en Ingeniería
Informática y Matemáticas
Facultad de Ciencias
Escuela Técnica Superior
de Ingenierías Informática
y de Telecomunicaciones
Universidad de Granada

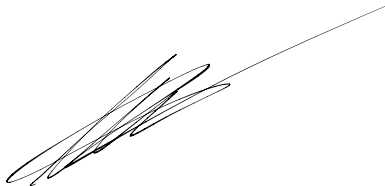
DECLARACIÓN DE ORIGINALIDAD

D. Javier Granados López

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2023-2024, es original, entendida esta, en el sentido de que no ha utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 18 de junio de 2024

Fdo: Javier Granados López

A handwritten signature in black ink, consisting of several overlapping loops and a long horizontal stroke extending to the right.

*A mi familia y amigos,
a mis profesores y tutores,
y a quienes confían plenamente en mí.*

Agradecimientos

Desearía expresar, si bien la palabra queda corta, la inmensa gratitud que siento ante mi familia, quienes sin cuya labor la conjunción de mis estudios no sería más que una vaga quimera; mis amigos, cuya sola presencia alivia mi cansancio; a Maxim y a Pedro, que de una forma u otra inspiraron maquinaciones y genialidades; los por mí considerados maestros de veras, que desde mi más tierna infancia fraguaron con vehemencia el estandarte que hoy alzo con orgullo; y a mis tutores José Luis y Diego, grandes atalayas que guiaron el proyecto en su rumbo a buen puerto.

Sépase que su inmarcesible apoyo vertebró de pies a cabeza el desarrollo de tan esmerado proyecto.

Muchas gracias, de corazón.

Índice general

Agradecimientos	IX
Summary	XIII
Introducción	XV
1. Fundamento	1
1.1. Espacios normados, Hilbertianos y bases de Hilbert	1
1.2. Espacios de Lebesgue	10
1.2.1. El espacio de Lebesgue $L_2(\Omega)$	17
1.3. Series de Fourier en $L_2([0, T])$	17
2. Discretización	21
2.1. Planteamiento	21
2.2. Aproximaciones	22
2.3. Coeficientes de Fourier aproximados y convergencia	23
2.4. Paso a \mathbb{C}^N . Ortogonalidad y transformaciones	26
2.5. La transformada discreta de Fourier	29
3. Implementación del conversor WAV-MIDI	35
3.1. Introducción. De la nota musical, el timbre y medidas de distancia	35
3.1.1. El oído humano: funcionamiento. Intervalos	37
3.2. Esbozo del algoritmo	38
3.2.1. Ventanas y solapamiento	39
3.2.2. Esquema general	40
3.3. El algoritmo de conversión	41
3.3.1. Lectura del fichero WAV y cálculo de parámetros	41
3.3.2. Estimación de la frecuencia fundamental	43
3.3.3. Corrección de tonos. Asignación de silencios	66
3.3.4. Estudio comparativo	69
3.3.5. Cálculo de duraciones	71
3.3.6. Paso a partitura	73
3.3.7. Otros ejemplos de ejecución	74
3.3.8. Eficiencia	75
3.3.9. Estado del arte en la detección de tonos	77
4. Transformaciones digitales	79
4.1. Filtrado de señales digitales	79
4.1.1. Ecualizador	82
4.1.2. Filtro de paso bajo	82
4.1.3. Filtro de paso alto	83
4.1.4. Filtro de paso banda	84
4.1.5. Filtro de banda rechazada o eliminada	85

Índice general

4.1.6. Implementación de un filtro	86
4.2. Compresión de audio	89
5. Planificación	93
5.1. Marco de desarrollo	93
5.2. Programación temporal	94
6. Implementación	97
6.1. Entorno de desarrollo	97
7. Tutorial de uso	99
7.1. WAV2MIDIConverter	99
7.2. 2MANYFILTERS	101
A. Conversor	103
A.1. WAV2MIDIConverter	103
B. Filtrado	115
B.1. 2MANYFILTERS	115
Bibliografía	121

Summary

The present final degree project aims to provide a comprehensive and organized exposition of the theoretical industry required for digital signal processing and its practical application in a range of useful apps for solving some specific and certainly complex problems in the world of digital audio.

Primarily focused on the field of music and, of course, audio, this work explores utilities as widely used and attractive as filtering, compression, or the conversion of *WAV* audio files to *MIDI* format, which is ultimately equivalent to conventional sheet music.

Common to all these challenges, both in mathematical and computational contexts, is frequency analysis of signals, a problem for which Fourier series provide an invaluable tool; indeed, they are considered a revolutionary 19th-century discovery worthy of being said to have changed the world as it was known.

In accordance with these objectives, Fourier series will be constructed in $L_2([0, T])$ for $T > 0$, which benefits from the properties of a Hilbert space, using the tools provided by functional analysis and the extension of linear algebra to infinite-dimensional vector spaces. The first goal, in this regard, will be to prove the existence of a Hilbert basis in the aforementioned scenario, with the reminder and demonstration of relevant results for this purpose. Intuitively, it will provide a decomposition of functions into pure signals (sine and cosine), whose associated Fourier coefficients will account for the relative prevalence of their frequencies in the original signal.

Once the theoretical and continuous frequency analysis is achieved, we proceed to its modeling in a computational environment, where we work with a finite and discrete set of data, now in finite-dimensional spaces. It will be shown how the model satisfactorily preserves the most relevant properties of its continuous counterpart, also allowing a reversible process that enables the transformation of a given signal into approximate Fourier coefficients (its frequency analysis) and vice versa.

With all this in hand, we will proceed to the raising of a specific discretization model that represents the epitome, in terms of efficiency, of Fourier analysis in computation: the Fast Fourier Transform (FFT) and its inverse counterpart (Inverse Fast Fourier Transform or IFFT).

And so, finally, we will embark on the implementation of algorithms that will address the proposed challenges. First and foremost, and by far the most complex of them all, will be the creation of an audio-to-score converter. Today, conciliating digital audio with common musical notation remains an immense challenge even for professional music editing programs like Sibelius or Dorico, in conjunction with ad hoc software, and DAWs like Reaper, as post-editing adjustments by the composer or arranger are mostly required to adapt the final format of the musical writing. Using unsupervised learning techniques, an own design orig-

Summary

inal algorithm is proposed to solve the problem, initially for monophonic (or single-voice) music, with a considerable level of precision, for which statistics and examples of performance are provided.

Lastly, in regards to the most well-known digital transformations that can be applied to audio files, we will program an app for running a wide variety of filters (low-pass, high-pass, band-pass, and band-stop), which are the daily bread of composers and audio engineers during mixing and mastering, but are also indispensable in telecommunications and many other engineering fields where the digital signal is the raw material. Additionally, the principles of *MP3* audio compression, one of the most well-known and widely used today, will be addressed.

During the completion of this project, the complexity of the major challenge, the conversion to sheet music, was repeatedly pointed out, especially when it was intended to operate in an independent and innovative manner. Ultimately, the conclusions that can be drawn, in light of the state of the art not only in this field but in many others still in development, are that Fourier analysis has a reach and range of utility so indescribably vast that, in conjunction with new technologies based on artificial intelligence, the limits that can be reached are unimaginable, even for the author of this humble work, not only in terms of the conversion problem but in signal processing as we know it today.

There is still much work to be done, but the countless possibilities for improvement are quite encouraging to move forwards.

Introducción

El presente trabajo de fin de grado encuentra su meta en la exposición completa y organizada de la industria teórica que precisa el tratamiento de señales digitales y su aplicación práctica en un compendio de aplicaciones útiles para la resolución de algunos problemas concretos y ciertamente complejos del mundo del audio digital.

Se trata, cimentado el interés de manera preferente sobre el campo de la música y, por supuesto, el audio, de utilidades de tal profusión de uso u atractivo como son el filtrado, la compresión, o la conversión de archivos de audio *WAV* a formato *MIDI*, el cual es, en última instancia, equivalente a la partitura convencional.

Y lo que resulta común a todos estos retos tanto en contexto matemático como computacional reside ineludiblemente en el análisis frecuencial de señales, problema para el cual las series de Fourier otorgan una herramienta de incalculable valor; tanto es así que se consideran un descubrimiento revolucionario del siglo XIX digno de decirse que ha cambiado el mundo tal y como se conocía.

De acuerdo a estos objetivos, se erigirán las series de Fourier en $L_2([0, T])$ para $T > 0$, que goza de la comodidad que surte de las propiedades de espacio de Hilbert, a partir de los instrumentos provistos por el análisis funcional y extensión del álgebra lineal a espacios vectoriales de dimensión infinita. La primera de las metas, a este respecto, será probar la existencia de una base de Hilbert en el escenario previamente mencionado, con el recordatorio y la demostración de resultados relevantes para ello; intuitivamente, proporcionará una descomposición de funciones en señales puras (seno y coseno), cuyos coeficientes de Fourier asociados darán cuenta de la prevalencia relativa de sus frecuencias en la señal de partida.

Una vez alcanzado el análisis de frecuencias teórico y continuo, procedemos a su modelización en ambiente computacional, donde se trabaja con un conjunto finito y discreto de datos, ahora sí, en espacios de dimensión finita. Se verá cómo el modelo conserva satisfactoriamente las propiedades más relevantes de su homólogo continuo dando cabida, además, a un proceso reversible que otorga la capacidad de transformar una señal dada en unos coeficientes de Fourier aproximados (su análisis de frecuencias) y viceversa.

Con todo este equipo, se procederá a la construcción de un modelo de discretización particular que responde a la sublimación, en términos de eficiencia, del análisis de Fourier en computación, la transformada rápida de Fourier (Fast Fourier Transform, o FFT) y su contraparte inversa (Inverse Fast Fourier Transform, o IFFT).

Y así, finalmente, podremos embarcarnos en la implementación de los algoritmos que darán respuesta a los retos planteados. En primer lugar y, con diferencia, el más complejo de todos ellos será dar nacimiento a un conversor de audio a partitura. A día de hoy, la conciliación del audio digital y la notación musical común sigue suponiendo un inmenso

desafío incluso para programas profesionales de edición musical como Sibelius o Dorico, en conjunción con software ad hoc, y DAWs como Reaper, de manera que casi siempre se va a precisar de retoques posteriores por parte del compositor o arreglista a fin de adecuar el formato último de la escritura musical. Haciendo uso de técnicas de aprendizaje no supervisado, se propone un algoritmo original de confección propia que resuelve el problema, a priori monódico (o a una voz), con un considerable nivel de precisión, y se aportan estadísticas y ejemplos de su desempeño.

Por último, y en relación a las más conocidas transformaciones digitales que pueden aplicarse a ficheros de audio, construiremos una utilidad de aplicación de filtros de todo tipo (paso bajo, alto, banda y banda eliminada), que suponen el pan de cada día de compositores e ingenieros de audio durante la mezcla y la masterización, pero que son también indispensables en telecomunicaciones y otras tantas ingenierías cuya materia prima es la señal digital. Asimismo, se abordarán los principios de la compresión de audio *MP3*, una de las más conocidas y utilizadas en la actualidad.

Durante la realización de este proyecto se puso de manifiesto en múltiples ocasiones la complejidad del desafío mayor que se propuso resolver, la conversión a partitura; máxime cuando se pretendía operar de manera independiente e inédita. Y, en definitiva, las conclusiones a las que se pueden llegar, a la vista del estado del arte no solo del propio campo, sino de tantos otros aún en desarrollo, es que el análisis de Fourier presenta un alcance y un rango de utilidad tan indeciblemente vastos que, en conjunción con las nuevas tecnologías basadas en inteligencia artificial, inimaginables quedan, aún para el redactor de este humilde trabajo, las cotas que se puedan alcanzar ya no solo en cuanto al problema de conversión, sino al tratamiento de señales tal y como hoy lo conocemos.

Aún queda mucho trabajo por delante, pero alentadoras son las innumerables posibilidades de mejora.

1. Fundamento

Se presentan, a continuación, tanto los elementos como los resultados teóricos que de ellos emergen, imprescindibles para alcanzar los objetivos propuestos en la práctica. Estos se encuadran en el marco del Análisis Funcional; más específicamente, ahondaremos en el Análisis de Fourier.

1.1. Espacios normados, Hilbertianos y bases de Hilbert

Sean K un cuerpo y E un K -espacio vectorial. Comenzamos definiendo una serie de conceptos generales:

Definición 1.1. Llamaremos **norma** a una aplicación $\|\cdot\| : E \rightarrow \mathbb{R}_0^+$ que verifica las siguientes propiedades:

- $\|x\| \geq 0 \quad \forall x \in E$; además, $\|x\| = 0$ si, y solo si, $x = 0$. (No negatividad)
- $\|\lambda x\| = |\lambda| \|x\| \quad \forall x \in E, \quad \forall \lambda \in K$. (Homogeneidad)
- $\|x + y\| \leq \|x\| + \|y\| \quad \forall x, y \in E$. (Desigualdad triangular)

Observación 1.1. Puede probarse fácilmente la desigualdad

$$|\|x\| - \|y\|| \leq \|x - y\| \quad \forall x, y \in E$$

lo cual implica, directamente, la continuidad de la norma.

Observación 1.2. Se dice **seminorma** a una aplicación $x \rightarrow \varphi(x) \in \mathbb{R}_0^+$ verificando las propiedades segunda y tercera anteriores.

En estas condiciones,

Definición 1.2. Diremos que un espacio vectorial es un **espacio normado** si está dotado de una norma.

Recordemos, además, y en virtud de la distancia inducida por la norma, que

Definición 1.3. Un espacio normado se dice **completo** si toda sucesión de Cauchy de elementos suyos es convergente en dicho espacio.

Así,

Definición 1.4. Diremos que un espacio vectorial es un **espacio de Banach** si es normado y completo.

Por otro lado,

Definición 1.5. Llamaremos **producto escalar** a una aplicación $\langle \cdot, \cdot \rangle : E \times E \rightarrow K$ satisfaciendo:

1. Fundamento

- $\langle x, x \rangle \in \mathbb{R}_0^+ \quad \forall x \in E$; además, $\langle x, x \rangle = 0$ si, y solo si, $x = 0$. (Definida positiva)
- $\langle x, y \rangle = \overline{\langle y, x \rangle} \quad \forall x, y \in E$. (Hermiticidad)
- $\langle \lambda x + \mu y, z \rangle = \lambda \langle x, z \rangle + \mu \langle y, z \rangle \quad \forall x, y, z \in E, \quad \forall \lambda, \mu \in K$. (Linealidad por la izquierda)

Observación 1.3. A propósito de esta definición cabe mencionar que de las tres propiedades expuestas (más en particular, de las dos últimas) se deduce la linealidad conjugada por la derecha; es decir, $\langle x, \lambda y + \mu z \rangle = \overline{\lambda} \langle x, y \rangle + \overline{\mu} \langle x, z \rangle \quad \forall x, y, z \in E, \quad \forall \lambda, \mu \in K$, lo que convierte al producto escalar en una aplicación sesquilineal.

Observación 1.4. En el caso de un espacio vectorial real, donde la conjugación resulta equivalente a la identidad, la hermiticidad del producto escalar se torna simetría y la sesquilinealidad antes mencionada degenera en bilinealidad.

Observación 1.5. Para $K = \mathbb{R}$ o $K = \mathbb{C}$, un espacio vectorial en que pueda definirse un producto escalar resulta inmediatamente normado, pues dicho producto escalar induce una norma dada por

$$\|x\| := \sqrt{\langle x, x \rangle} \quad \forall x \in E$$

Consideremos, en los sucesivos, $K = \mathbb{R}$ o $K = \mathbb{C}$. Observemos que:

Proposición 1.1. (Desigualdad de Cauchy-Schwartz). Sea E un K -espacio vectorial con producto escalar $\langle \cdot, \cdot \rangle : E \times E \rightarrow K$ y sea $\| \cdot \|$ la norma asociada. Se cumple que

$$|\langle x, y \rangle| \leq \|x\| \|y\| \quad \forall x, y \in E$$

Demostración. Sean $x, y \in E$ arbitrarios. Basta ver, equivalentemente (pues son ambos miembros no negativos), que

$$|\langle x, y \rangle|^2 \leq \|x\|^2 \|y\|^2 = \langle x, x \rangle \langle y, y \rangle$$

Tal desigualdad resulta trivial en el caso $x = 0$. Por lo tanto, supondremos en adelante que $x \neq 0$.

Tomemos ahora $\lambda, \mu \in K$ nuevamente arbitrarios. Sabemos que $\langle z, z \rangle \geq 0 \quad \forall z \in E$, luego

$$\begin{aligned} \langle \lambda x - \mu y, \lambda x - \mu y \rangle &= \lambda \langle x, \lambda x - \mu y \rangle - \mu \langle y, \lambda x - \mu y \rangle \\ &= \lambda \overline{\lambda} \langle x, x \rangle - \lambda \overline{\mu} \langle x, y \rangle - \mu \overline{\lambda} \langle y, x \rangle + \mu \overline{\mu} \langle y, y \rangle \\ &= |\lambda|^2 \langle x, x \rangle - \lambda \overline{\mu} \langle x, y \rangle - \mu \overline{\lambda} \langle y, x \rangle + |\mu|^2 \langle y, y \rangle \geq 0 \end{aligned}$$

Esto debe satisfacerse para cualesquiera λ y μ elegidos; en particular, para $\lambda = \langle y, x \rangle$ y $\mu = \langle x, x \rangle$. Así,

$$\begin{aligned} |\langle y, x \rangle|^2 \langle x, x \rangle - \langle y, x \rangle \overline{\langle x, x \rangle} \langle x, y \rangle - \langle x, x \rangle \overline{\langle y, x \rangle} \langle y, x \rangle + |\langle x, x \rangle|^2 \langle y, y \rangle \\ &= |\langle y, x \rangle|^2 \langle x, x \rangle - 2 \langle y, x \rangle \overline{\langle x, x \rangle} \langle x, y \rangle + |\langle x, x \rangle|^2 \langle y, y \rangle \\ &= |\langle y, x \rangle|^2 \langle x, x \rangle - 2 |\langle y, x \rangle|^2 \langle x, x \rangle + |\langle x, x \rangle|^2 \langle y, y \rangle \\ &= |\langle x, x \rangle|^2 \langle y, y \rangle - |\langle y, x \rangle|^2 \langle x, x \rangle \\ &= |\langle x, x \rangle|^2 \langle y, y \rangle - |\langle x, y \rangle|^2 \langle x, x \rangle \geq 0 \end{aligned}$$

de donde

$$|\langle x, y \rangle|^2 \langle x, x \rangle \leq |\langle x, x \rangle|^2 \langle y, y \rangle$$

Puesto que $x \neq 0$, se tiene necesariamente $\langle x, x \rangle > 0$, lo que permite afirmar que

$$|\langle x, y \rangle|^2 \leq \frac{|\langle x, x \rangle|^2 \langle y, y \rangle}{\langle x, x \rangle} = \langle x, x \rangle \langle y, y \rangle$$

■

Proposición 1.2. (Continuidad del producto escalar). Sea E un K -espacio vectorial con producto escalar $\langle \cdot, \cdot \rangle : E \times E \rightarrow K$ y sea $\| \cdot \|$ la norma asociada. Entonces, dados $x, y \in E$ y $\{x_n\}_{n \in \mathbb{N}}, \{y_n\}_{n \in \mathbb{N}}$ sucesiones en E tales que $x = \lim_{n \rightarrow +\infty} x_n$ e $y = \lim_{n \rightarrow +\infty} y_n$, se verifica entonces que $\lim_{n \rightarrow +\infty} \langle x_n, y_n \rangle = \langle x, y \rangle$.

Demostración. Tenemos $\|x - x_n\| \rightarrow 0$ y $\|y - y_n\| \rightarrow 0$, luego

$$\begin{aligned} |\langle x, y \rangle - \langle x_n, y_n \rangle| &= |\langle x, y \rangle - \langle x_n, y \rangle + \langle x_n, y \rangle - \langle x_n, y_n \rangle| \\ &= |\langle x - x_n, y \rangle + \langle x_n, y - y_n \rangle| \\ &\leq \underbrace{|\langle x - x_n, y \rangle|}_{D.T.} + |\langle x_n, y - y_n \rangle| \\ &\leq \underbrace{\|x - x_n\| \|y\|}_{C-S} + \underbrace{\|x_n\|}_{Acotada} \|y - y_n\| \rightarrow 0 \end{aligned}$$

■

A continuación, aprovechamos para demostrar una sencilla propiedad de los espacios normados que será de utilidad más adelante.

Proposición 1.3. Sea E un K -espacio vectorial normado y sea $\| \cdot \|$ la norma asociada. Consideremos $\{x_n\}_{n \in \mathbb{N}}$ una sucesión de Cauchy en E que admite una sucesión parcial convergente $\{x_{\sigma(n)}\} \rightarrow x \in E$, donde $\sigma : \mathbb{N} \rightarrow \mathbb{N}$ es una aplicación estrictamente creciente. Entonces se cumple $\{x_n\} \rightarrow x$.

Demostración. Por ser $\{x_n\}$ de Cauchy, dado $\epsilon > 0$, $\exists N_\epsilon^1 \in \mathbb{N} / \forall n, m \geq N_\epsilon^1, \|x_m - x_n\| \leq \frac{\epsilon}{2}$. Por otro lado, por definición de convergencia de una sucesión, dado $\epsilon > 0$, $\exists N_\epsilon^2 \in \mathbb{N} / \forall n \geq N_\epsilon^2, \|x - x_{\sigma(n)}\| \leq \frac{\epsilon}{2}$.

En consecuencia, poniendo $N_\epsilon^3 = \max\{N_\epsilon^1, N_\epsilon^2\}$ se tiene, $\forall n \geq N_\epsilon^3$ y por ser $\sigma(n) \geq n \quad \forall n \in \mathbb{N}$,

$$\|x - x_n\| = \|x - x_{\sigma(n)} + x_{\sigma(n)} - x_n\| \leq \|x - x_{\sigma(n)}\| + \|x_{\sigma(n)} - x_n\| \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon$$

■

Y, finalmente, alcanzamos la particularización deseada:

Definición 1.6. Diremos que un espacio vectorial es un **espacio pre-hilbertiano** si está dotado de un producto escalar.

Definición 1.7. Diremos que un espacio vectorial es un **espacio de Hilbert** si es de Banach y está dotado de un producto escalar.

Ahora ahondamos en el último concepto estudiando, para aquellos espacios de Hilbert de dimensión infinita, una generalización de la base de Hamel propia del Álgebra lineal y el

1. Fundamento

espacio euclídeo: la base de Hilbert.

Pongamos H un K -espacio vectorial de Hilbert de dimensión infinita.

Definición 1.8. Sean I un conjunto de índices y $\{\phi_i\}_{i \in I}$ una familia no vacía de elementos de $H \setminus \{0\}$. Se dirá que tal familia es un **sistema ortogonal** de H si sus elementos son ortogonales dos a dos; esto es, $\langle \phi_i, \phi_j \rangle = 0 \quad \forall i \neq j, \quad i, j \in I$.

Definición 1.9. Sean I un conjunto de índices y $\{\phi_i\}_{i \in I}$ una familia no vacía de elementos de $H \setminus \{0\}$. Se dirá que tal familia es un **sistema ortonormal** de H si se trata de un sistema ortogonal y, además, sus elementos están normalizados; es decir, $\|\phi_i\| = 1 \quad \forall i \in I$.

Definición 1.10. Sean I un conjunto de índices y $\{\phi_i\}_{i \in I}$ una familia no vacía de elementos de $H \setminus \{0\}$. Se dirá que tal familia es una **base de Hilbert** de H si se trata de un sistema ortonormal y el subespacio vectorial que genera es denso en H ; es decir, si $\overline{L(\{\phi_i\}_{i \in I})} = H$.

Observación 1.6. Notemos que, dado $\{\phi_i\}_{i \in I}$ sistema ortogonal de H , podemos construir, por medio de la normalización de sus elementos, el sistema ortonormal $\{\frac{\phi_i}{\|\phi_i\|}\}_{i \in I}$.

En referencia a estos conceptos, cabe destacar la siguiente

Proposición 1.4. (Desigualdad de Bessel). Sea E un K -espacio vectorial pre-hilbertiano, con producto escalar $\langle \cdot, \cdot \rangle : E \times E \rightarrow K$ y sea $\|\cdot\|$ la norma asociada. Sea $\{\phi_n\}_{n \in \mathbb{N}}$ un sistema ortonormal en H . Entonces se cumple

$$\sum_{n \geq 1} |\langle x, \phi_n \rangle|^2 \leq \|x\|^2 \quad \forall x \in E$$

A los valores $\langle x, \phi_n \rangle$ se les denomina *coeficientes de Fourier* de x respecto del sistema $\{\phi_n\}_{n \in \mathbb{N}}$.

Demostración. Sea $x \in H$ y consideremos las sumas parciales

$$S_n := \sum_{k=1}^n \langle x, \phi_k \rangle \phi_k \quad \forall n \in \mathbb{N}$$

Se tiene, por un lado,

$$\langle x, S_n \rangle = \langle x, \sum_{k=1}^n \langle x, \phi_k \rangle \phi_k \rangle = \sum_{k=1}^n \overline{\langle x, \phi_k \rangle} \langle x, \phi_k \rangle = \sum_{k=1}^n |\langle x, \phi_k \rangle|^2$$

y, por otro,

$$\langle S_n, S_n \rangle = \langle \sum_{k=1}^n \langle x, \phi_k \rangle \phi_k, \sum_{k=1}^n \langle x, \phi_k \rangle \phi_k \rangle = \sum_{k=1}^n \sum_{j=1}^n \langle x, \phi_k \rangle \overline{\langle x, \phi_j \rangle} \langle \phi_k, \phi_j \rangle = \sum_{k=1}^n \overline{\langle x, \phi_k \rangle} \langle x, \phi_k \rangle = \sum_{k=1}^n |\langle x, \phi_k \rangle|^2$$

Así,

$$\begin{aligned} 0 \leq \|x - S_n\|^2 &= \langle x - S_n, x - S_n \rangle \\ &= \langle x - S_n, x \rangle - \langle x - S_n, S_n \rangle \\ &= \langle x, x \rangle - \langle S_n, x \rangle - \langle x, S_n \rangle + \langle S_n, S_n \rangle \\ &= \langle x, x \rangle - \langle S_n, x \rangle - \sum_{k=1}^n |\langle x, \phi_k \rangle|^2 + \sum_{k=1}^n |\langle x, \phi_k \rangle|^2 \\ &= \langle x, x \rangle - \langle S_n, x \rangle \end{aligned}$$

Luego

$$\langle S_n, x \rangle \leq \|x\|^2 \Rightarrow \sum_{k=1}^n |\langle x, \phi_k \rangle|^2 = \sum_{k=1}^n \langle x, \phi_k \rangle \overline{\langle x, \phi_k \rangle} = \left\langle \sum_{k=1}^n \langle x, \phi_k \rangle \phi_k, x \right\rangle \leq \|x\|^2 \quad \forall n \in \mathbb{N}$$

Tomando límite en esta última desigualdad alcanzamos el resultado buscado. ■

Pasamos a hablar acerca de ortogonalidad, lo que nos permitirá probar una variante de un resultado central y de gran intuición geométrica: el Teorema de la proyección ortogonal. Este otorga un mayor entendimiento de la estructura de los espacios pre-hilbertianos y de Hilbert y, además, es clave en lo que respecta a teoría de aproximación.

Definición 1.11. Sea E un K -espacio vectorial pre-hilbertiano. Dados dos elementos $x, y \in E$, se dirán **ortogonales** si $\langle x, y \rangle = 0$.

En base a esto,

Definición 1.12. Sean E un K -espacio vectorial pre-hilbertiano y S un subconjunto de E . Diremos que un elemento $x \in E$ es **ortogonal a S** si lo es con respecto a todo elemento de dicho conjunto; esto es, si $\langle x, y \rangle = 0 \quad \forall y \in S$. Se dirá **complemento ortogonal de S** al conjunto

$$S^\perp = \{x \in E : \langle x, y \rangle = 0 \quad \forall y \in S\}$$

Observación 1.7. A causa de la sesquilinealidad del producto escalar, cabe observar que el complemento ortogonal de un subespacio vectorial es, igualmente, un subespacio vectorial. También es claro que $S \cap S^\perp = \emptyset$.

Lema 1.1. (Identidad del paralelogramo). Sean E un K -espacio vectorial pre-hilbertiano, con producto escalar $\langle \cdot, \cdot \rangle : E \times E \rightarrow K$ y sea $\|\cdot\|$ la norma asociada. Entonces, se tiene

$$\|x + y\|^2 + \|x - y\|^2 = 2\|x\|^2 + 2\|y\|^2 \quad \forall x, y \in E$$

Demostración. Sean $x, y \in E$ arbitrarios. Se tiene

$$\begin{aligned} \|x + y\|^2 + \|x - y\|^2 &= \langle x + y, x + y \rangle + \langle x - y, x - y \rangle \\ &= \langle x, x \rangle + \langle x, y \rangle + \langle y, x \rangle + \langle y, y \rangle + \langle x, x \rangle - \langle x, y \rangle - \langle y, x \rangle + \langle y, y \rangle \\ &= 2\langle x, x \rangle + 2\langle y, y \rangle \\ &= 2\|x\|^2 + 2\|y\|^2 \end{aligned}$$

Y, como veníamos anunciando,

Teorema 1.1. (Teorema de la proyección ortogonal). Sean H un K -espacio vectorial de Hilbert y S un subespacio vectorial cerrado de H . Entonces, dado $x \in H$, $\exists! \tilde{x} \in S$ de manera que x se expresa de forma única como $x = \tilde{x} + (x - \tilde{x})$ con $x - \tilde{x} \in S^\perp$; es decir, $H = S \oplus S^\perp$, siendo \tilde{x} la mejor aproximación de x por elementos de S , donde con ello se entiende que

$$\|x - \tilde{x}\| < \|x - y\| \quad \forall y \in S \setminus \{\tilde{x}\}$$

1. Fundamento

con $\|\cdot\|$ la norma inducida por el producto escalar.

Demostración. Probaremos, en primera instancia, una desigualdad sobre la que apoyaremos la demostración de este teorema. Sea $x \in H$ y sean $y, z \in S$ arbitrarios. De la desigualdad del paralelogramo se obtiene que

$$\begin{aligned} \|(x-y) + (x-z)\|^2 + \|(x-y) - (x-z)\|^2 &= 2\|x-y\|^2 + 2\|x-z\|^2 \\ \iff \|z-y\|^2 &= 2\|x-y\|^2 + 2\|x-z\|^2 - 4\|x - \frac{y+z}{2}\|^2 \end{aligned}$$

Ahora bien, llamando al ínfimo de las distancias de x a elementos de S

$$d(x, S) := \inf\{\|x - w\| : w \in S\}$$

y teniendo en cuenta que $\frac{y+z}{2} \in S$ por ser subespacio vectorial, podemos tornar la igualdad anterior en la desigualdad

$$\|z-y\|^2 \leq 2\|x-y\|^2 + 2\|x-z\|^2 - 4d(x, S)^2$$

puesto que $d(x, S) \leq \|x - w\| \quad \forall w \in S$.

Ya tenemos la desigualdad que buscábamos. El siguiente paso es tomar una sucesión $\{y_n\}_{n \in \mathbb{N}}$ de elementos de S de manera que $\{\|x - y_n\|\} \rightarrow d(x, S)$, lo cual se justifica por definición del ínfimo de un conjunto. Así, $\{\|x - y_n\|^2\} \rightarrow d(x, S)^2$ y dado $\epsilon > 0$, $\exists N_\epsilon \in \mathbb{N} / \forall n \geq N_\epsilon$,

$$\|x - y_n\|^2 - d(x, S)^2 < \frac{\epsilon^2}{4} \iff \|x - y_n\|^2 < d(x, S)^2 + \frac{\epsilon^2}{4}$$

En consecuencia, considerados $n, m \geq N_\epsilon$ poniendo $y = y_m$ y $z = y_n$, la desigualdad demostrada antes implica que

$$\begin{aligned} \|y_n - y_m\|^2 &= 2\|x - y_m\|^2 + 2\|x - y_n\|^2 - 4d(x, S)^2 \\ &< 2\left(d(x, S)^2 + \frac{\epsilon^2}{4}\right) + 2\left(d(x, S)^2 + \frac{\epsilon^2}{4}\right) - 4d(x, S)^2 = \epsilon^2 \end{aligned}$$

de donde deduce que $\{y_n\}_{n \in \mathbb{N}}$ es una sucesión de Cauchy en S . Por ser H completo y S cerrado, $\exists \tilde{x} \in S / \{y_n\} \rightarrow \tilde{x}$. Este hecho, sumado a la continuidad de la norma, nos permite concluir que

$$\|x - \tilde{x}\| = \lim_{n \rightarrow +\infty} \|x - y_n\| = d(x, S) < \|x - w\| \quad \forall w \in S \setminus \{\tilde{x}\}$$

Finalmente, supongamos que tanto \tilde{x} como z son ambas mejores aproximaciones de x por elementos de S . Entonces, $\|x - \tilde{x}\| = d(x, S) = \|x - z\|$, luego

$$\|z - \tilde{x}\|^2 \leq 2\|x - \tilde{x}\|^2 + 2\|x - z\|^2 - 4d(x, S)^2 = 0$$

de donde necesariamente $\tilde{x} = z$, lo que prueba la unicidad de la mejor aproximación.

Veamos a continuación que $x - \tilde{x} \in S^\perp$. Para ello, puesto que $\forall w \in S$

$$\begin{aligned} \|x - \tilde{x}\|^2 &\leq \|x - w\|^2 = \|(x - \tilde{x}) - (w - \tilde{x})\|^2 \\ &= \langle (x - \tilde{x}) - (w - \tilde{x}), (x - \tilde{x}) - (w - \tilde{x}) \rangle \\ &= \langle x - \tilde{x}, x - \tilde{x} \rangle - \langle x - \tilde{x}, w - \tilde{x} \rangle - \langle w - \tilde{x}, x - \tilde{x} \rangle + \langle w - \tilde{x}, w - \tilde{x} \rangle \\ &= \|x - \tilde{x}\|^2 + \|w - \tilde{x}\|^2 - 2\operatorname{Re}(\langle x - \tilde{x}, w - \tilde{x} \rangle) \end{aligned}$$

razonamos que

$$\begin{aligned} 2\operatorname{Re}(\langle x - \tilde{x}, w - \tilde{x} \rangle) &\leq \|w - \tilde{x}\|^2 \quad \forall w \in S \\ &\iff \bigwedge_{t(w-\tilde{x}) \in S} 2\operatorname{Re}(\langle x - \tilde{x}, tw \rangle) \leq \|w - \tilde{x}\|^2 \quad \forall w \in S, \quad \forall t \in K \\ &\implies \operatorname{Re}(\langle x - \tilde{x}, w \rangle) \leq \frac{2\|w - \tilde{x}\|^2}{t} \quad \forall w \in S, \quad \forall t \in K \setminus \{0\} \\ &\iff \pm \operatorname{Re}(\langle x - \tilde{x}, w \rangle) = \operatorname{Re}(\langle x - \tilde{x}, \pm w \rangle) \leq \frac{2\|w - \tilde{x}\|^2}{t} \quad \forall w \in S, \quad \forall t \in K \setminus \{0\} \\ &\iff |\operatorname{Re}(\langle x - \tilde{x}, w \rangle)| \leq \frac{2\|w - \tilde{x}\|^2}{t} \quad \forall w \in S, \quad \forall t \in K \setminus \{0\} \end{aligned}$$

Tomando límites en esta desigualdad para $t \rightarrow +\infty$, logramos $\operatorname{Re}(\langle x - \tilde{x}, w \rangle) = 0 \quad \forall w \in S$. Equivalentemente, $\operatorname{Im}(\langle x - \tilde{x}, w \rangle) = \operatorname{Re}(\langle x - \tilde{x}, iw \rangle) = 0 \quad \forall w \in S$, por lo cual

$$\langle x - \tilde{x}, w \rangle = 0 \quad \forall w \in S$$

y, así, $x - \tilde{x} \in S^\perp$ como queríamos.

Advertimos que $x = \tilde{x} + (x - \tilde{x}) \in S + S^\perp$. Añadidos la arbitrariedad del elemento x elegido y que $S \cap S^\perp = \emptyset$, conseguimos la igualdad $H = S \oplus S^\perp$. ■

A continuación, se muestra una versión de este teorema para espacios pre-hilbertianos y subconjuntos de dimensión finita, escenario en el que podemos conocer algo más acerca de la mejor aproximación.

Teorema 1.2. Sean E un K -espacio vectorial pre-hilbertiano y S un subespacio vectorial de E de dimensión finita con $\dim_K(S) = n \in \mathbb{N}$. Entonces, dado $x \in E$, $\exists! \tilde{x} \in S$ de manera que x se expresa de forma única como $x = \tilde{x} + (x - \tilde{x})$ donde $x - \tilde{x} \in S^\perp$; es decir, $E = S \oplus S^\perp$.

Además, dada $\{\phi_1, \phi_2, \dots, \phi_n\}$ base ortonormal de S , el elemento \tilde{x} , denominado **proyección ortogonal de x sobre S** , se conoce explícitamente, siendo

$$\tilde{x} = \sum_{k=1}^n \langle x, \phi_k \rangle \phi_k$$

y esta es la mejor aproximación de x por elementos de S , donde con ello se entiende que

$$\|x - \tilde{x}\| < \|x - y\| \quad \forall y \in S \setminus \{\tilde{x}\}$$

con $\|\cdot\|$ la norma inducida por el producto escalar.

1. Fundamento

Demostración. Sea $x \in E$ arbitrario y comencemos probando la existencia de un elemento tal. Tomamos $\{\phi_1, \phi_2, \dots, \phi_n\}$ una base ortonormal de S cualquiera y llamamos

$$\tilde{x} = \sum_{k=1}^n \langle x, \phi_k \rangle \phi_k \in S$$

Es claro que $x = \tilde{x} + (x - \tilde{x})$. Veamos que $x - \tilde{x} \in S^\perp$, para lo cual tan solo basta con probar que este elemento es ortogonal a todos y cada uno de los elementos de la base de S elegida. En efecto, dado $j \in \{1, 2, \dots, n\}$ arbitrario,

$$\begin{aligned} \langle x - \tilde{x}, \phi_j \rangle &= \langle x, \phi_j \rangle - \langle \tilde{x}, \phi_j \rangle \\ &= \langle x, \phi_j \rangle - \left\langle \sum_{k=1}^n \langle x, \phi_k \rangle \phi_k, \phi_j \right\rangle \\ &= \langle x, \phi_j \rangle - \sum_{k=1}^n \langle x, \phi_k \rangle \langle \phi_k, \phi_j \rangle \\ &= \langle x, \phi_j \rangle - \langle x, \phi_j \rangle = 0 \end{aligned}$$

por ortogonalidad de los elementos de la base, luego $x - \tilde{x} \in S^\perp$.

Pongamos ahora $\hat{x} \in S$ de forma que, igualmente, $x = \hat{x} + (x - \hat{x})$ con $x - \hat{x} \in S^\perp$. Por ser S y S^\perp subespacios vectoriales, $\tilde{x} - \hat{x} \in S$ y $(x - \hat{x}) - (x - \tilde{x}) = \tilde{x} - \hat{x} \in S^\perp$ y, así,

$$\|\tilde{x} - \hat{x}\|^2 = \langle \tilde{x} - \hat{x}, \tilde{x} - \hat{x} \rangle = 0 \implies \tilde{x} - \hat{x} = 0$$

lo que prueba la unicidad, junto a que $E = S \oplus S^\perp$.

Finalmente, probemos que \tilde{x} es la mejor aproximación de x por elementos de S . Ciertamente, dado $y \in S \setminus \{\tilde{x}\}$ arbitrario, se tiene $x - \tilde{x} \in S^\perp$, $\tilde{x} - y \in S$, luego

$$\begin{aligned} \|x - y\|^2 &= \|(x - \tilde{x}) + (\tilde{x} - y)\|^2 \\ &= \langle (x - \tilde{x}) + (\tilde{x} - y), (x - \tilde{x}) + (\tilde{x} - y) \rangle \\ &= \langle x - \tilde{x}, x - \tilde{x} \rangle + \underbrace{\langle x - \tilde{x}, \tilde{x} - y \rangle}_0 + \underbrace{\langle \tilde{x} - y, x - \tilde{x} \rangle}_0 + \langle \tilde{x} - y, \tilde{x} - y \rangle \\ &= \langle x - \tilde{x}, x - \tilde{x} \rangle + \langle \tilde{x} - y, \tilde{x} - y \rangle \\ &= \|x - \tilde{x}\|^2 + \|\tilde{x} - y\|^2 \underset{y \neq \tilde{x}}{>} \|x - \tilde{x}\|^2 \end{aligned}$$

■

Ponemos fin a esta sección probando un resultado extremadamente relevante a este respecto, no sin antes demostrar un pequeño resultado auxiliar.

Lema 1.2. Sea $\{\phi_n\}_{n \in \mathbb{N}}$ una base de Hilbert numerable de H . Sea $x \in H$ / $\langle x, \phi_n \rangle = 0 \quad \forall n \in \mathbb{N}$. Entonces se tiene $x = 0$.

Demostración. Sea $x \in H$. Por hipótesis, existe una sucesión $\{x_n\}_{n \in \mathbb{N}}$ en $L(\{\phi_n\}_{n \in \mathbb{N}})$ de manera que $x = \lim_{n \rightarrow +\infty} x_n$. Ahora bien, puesto que $\langle x, \phi_n \rangle = 0 \quad \forall n \in \mathbb{N}$, necesariamente $\langle x, x_n \rangle = 0 \quad \forall n \in \mathbb{N}$. Podemos tomar límites en esta expresión para, tras utilizar la continuidad del producto escalar 1.2 y la unicidad del límite, llegar a $\langle x, x \rangle = 0 \implies x = 0$.



Teorema 1.3. Sea $\{\phi_n\}_{n \in \mathbb{N}}$ una base de Hilbert numerable de H . Entonces se cumple

$$x = \sum_{n \geq 1} \langle x, \phi_n \rangle \phi_n \quad \forall x \in H$$

Demostración. Sea $x \in H$ y consideremos la sucesión $\{x_n\}_{n \in \mathbb{N}}$ definida por

$$x_n := \sum_{k=1}^n \langle x, \phi_k \rangle \phi_k \quad \forall n \in \mathbb{N}$$

Probemos, en primer lugar, que dicha sucesión es de Cauchy.

Dados $n, m \in \mathbb{N}$, $n < m$, resulta

$$\begin{aligned} \|x_n - x_m\|^2 &= \|x_m - x_n\|^2 \\ &= \langle x_m - x_n, x_m - x_n \rangle \\ &= \left\langle \sum_{k=n+1}^m \langle x, \phi_k \rangle \phi_k, \sum_{k=n+1}^m \langle x, \phi_k \rangle \phi_k \right\rangle \\ &= \sum_{k=n+1}^m \langle x, \phi_k \rangle^2 \end{aligned}$$

La desigualdad de Bessel implica que la serie $\sum_{n \geq 1} \langle x, \phi_k \rangle^2$ es convergente, de manera que la serie de restos converge a 0; es decir,

$$\lim_{n, m \rightarrow +\infty} \sum_{k=n+1}^m \langle x, \phi_k \rangle^2 = \lim_{n \rightarrow +\infty} \sum_{k=n+1}^{+\infty} \langle x, \phi_k \rangle^2 = 0$$

como queríamos.

Estamos ahora capacitados para escribir $y = \lim_{n \rightarrow +\infty} x_n$; veamos que, en realidad, $y = x$. Para esta tarea haremos uso del lema 1.2, en virtud del cual tan solo será preciso comprobar que $\langle x - y, \phi_n \rangle = 0 \quad \forall n \in \mathbb{N}$.

Primero, advirtamos que

$$\langle x_m, \phi_n \rangle = \left\langle \sum_{k=1}^m \langle x, \phi_k \rangle \phi_k, \phi_n \right\rangle = \begin{cases} 0 & \text{si } m < n \\ \langle x, \phi_n \rangle & \text{si } m \geq n \end{cases}$$

En consecuencia, finalmente

$$\begin{aligned} \langle x - y, \phi_n \rangle &= \langle x, \phi_n \rangle - \langle y, \phi_n \rangle \\ &= \langle x, \phi_n \rangle - \left\langle \lim_{m \rightarrow +\infty} x_m, \phi_n \right\rangle \\ &\stackrel{\text{Prop. 1.2}}{=} \underbrace{\langle x, \phi_n \rangle}_{\text{Prop. 1.2}} - \lim_{m \rightarrow +\infty} \langle x_m, \phi_n \rangle \\ &= \langle x, \phi_n \rangle - \langle x, \phi_n \rangle = 0 \end{aligned}$$



Observación 1.8. En caso de disponer de $\{\phi_n\}_{n \in \mathbb{N}}$ un sistema ortogonal de manera que

1. Fundamento

$\overline{L(\{\phi_i\}_{i \in I})} = H$, entonces se verifica

$$x = \sum_{n \geq 1} \frac{\langle x, \phi_n \rangle}{\|\phi_n\|^2} \phi_n \quad \forall x \in H$$

Observación 1.9. De forma general, una base de Hilbert puede ser, en efecto, no numerable. Sin embargo, la sumabilidad de sus elementos pondría en jaque a resultados como el anterior. Es por esto que se adopta la hipótesis de numerabilidad de la base, lo que implica inmediatamente que lo que se trabaja no son sino espacios de Hilbert separables. Con todo, esta tipología es común en el ámbito de la Física, y no se pierde generalidad en lo referente a nuestro objeto de estudio.

1.2. Espacios de Lebesgue

Como principal objeto de estudio, nuestro recorrido se asienta, especialmente, en un conjunto de espacios normados concreto: los espacios de Lebesgue. En su seno, destacamos fundamentalmente al espacio L^2 , que introduciremos más adelante.

Definición 1.13. Sea $\Omega \subseteq \mathbb{R}$ un conjunto medible con medida de Lebesgue no nula. Denotaremos al conjunto de las **funciones medibles sobre Ω y con imagen en \mathbb{C}** por

$$\mathcal{L}(\Omega) := \{f : \Omega \longrightarrow \mathbb{C} \mid f \text{ es medible}\}$$

Definición 1.14. Dada $f \in \mathcal{L}(\Omega)$, diremos que f es una **función p -integrable**, $p \geq 1$, cuando

$$\int_{\Omega} |f|^p < +\infty$$

y denotaremos al conjunto de **funciones p -integrables** por

$$\mathcal{L}_p(\Omega) := \{f \in \mathcal{L}(\Omega) : \int_{\Omega} |f|^p < +\infty\}$$

Observación 1.10. Cabe notar que para $p = 1$, $\mathcal{L}_1(\Omega)$ identifica al conjunto de funciones integrables.

Nuestro objetivo es encontrar una estructura de espacio normado. Hemos definido, a un lado, los conjuntos de elementos que interesa estudiar. Ahora nos centramos en verificar la existencia de una posible norma.

Lema 1.3. (*Desigualdad de Young*). Sean $a, b \in \mathbb{R}_0^+$ y $p \in \mathbb{R}, p > 1$. Sea $q \in \mathbb{R} / \frac{1}{p} + \frac{1}{q} = 1$. Se cumple que

$$ab \leq \frac{a^p}{p} + \frac{b^q}{q}$$

Demostración. El resultado es trivial para $ab = 0$.

Supongamos $a, b \in \mathbb{R}^+$. En este caso, por concavidad del logaritmo,

$$\log\left(\frac{a^p}{p} + \frac{b^q}{q}\right) \geq \frac{1}{p} \log(a^p) + \frac{1}{q} \log(b^q) = \log(ab)$$

Finalmente, el crecimiento de la función exponencial nos otorga la desigualdad buscada.

■

Observación 1.11. Es preciso señalar que, siendo $p > 1$ y $\frac{1}{p} + \frac{1}{q} = 1$, entonces debe ser igualmente $q > 1$. A dos números en tales condiciones se les conoce como **exponentes conjugados**.

Lema 1.4. (Desigualdad de Hölder). Sean $f \in \mathcal{L}_p(\Omega)$, $g \in \mathcal{L}_q(\Omega)$, con $p, q \in \mathbb{R}$, $p > 1$ y $\frac{1}{p} + \frac{1}{q} = 1$. Entonces la función fg es integrable y

$$\int_{\Omega} |fg| \leq \left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} \left(\int_{\Omega} |g|^q \right)^{\frac{1}{q}}$$

Demostración. Supongamos $\left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} = 0$. Entonces, $\int_{\Omega} |f|^p = 0 \Leftrightarrow |f|^p = 0$ c.p.d. $\Leftrightarrow f = 0$ c.p.d. $\Rightarrow |fg| = 0$ c.p.d. $\Leftrightarrow \int_{\Omega} |fg| = 0$ y se tiene. De manera análoga sucede en caso de que $\left(\int_{\Omega} |g|^q \right)^{\frac{1}{q}} = 0$.

Supongamos, asimismo, que $\left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} \left(\int_{\Omega} |g|^q \right)^{\frac{1}{q}} \neq 0$. La desigualdad de Young 1.3 permite afirmar que

$$\frac{|f(x)||g(x)|}{\left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} \left(\int_{\Omega} |g|^q \right)^{\frac{1}{q}}} \leq \frac{|f(x)|^p}{p \int_{\Omega} |f|^p} + \frac{|g(x)|^q}{q \int_{\Omega} |g|^q} \quad \forall x \in \Omega$$

Integrando ambos miembros de la desigualdad y aplicando linealidad de la integral de Lebesgue, tenemos

$$\frac{1}{\left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} \left(\int_{\Omega} |g|^q \right)^{\frac{1}{q}}} \int_{\Omega} |fg| \leq \frac{1}{p \int_{\Omega} |f|^p} \int_{\Omega} |f|^p + \frac{1}{q \int_{\Omega} |g|^q} \int_{\Omega} |g|^q = \frac{1}{p} + \frac{1}{q} = 1$$

de donde se sigue fácilmente el resultado.

■

Proposición 1.5. (Desigualdad de Minkowski). Sean $f, g \in \mathcal{L}_p(\Omega)$, con $p \in \mathbb{R}$, $p \geq 1$. Se cumple que

$$\left(\int_{\Omega} |f + g|^p \right)^{\frac{1}{p}} \leq \left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} + \left(\int_{\Omega} |g|^p \right)^{\frac{1}{p}}$$

Demostración. Aplicando crecimiento y luego linealidad de la integral de Lebesgue, se tiene

$$\int_{\Omega} |f + g| \leq \int_{\Omega} (|f| + |g|) = \int_{\Omega} |f| + \int_{\Omega} |g|$$

lo que prueba el caso $p = 1$.

Sea ahora $p > 1$. Por ser $f, g \in \mathcal{L}_p(\Omega)$, los sumandos de la derecha y, por consiguiente, su suma, son finitos. Veamos, por tanto y en primer lugar, que $f + g \in \mathcal{L}_p(\Omega)$.

Definamos el conjunto

$$A := \{x \in \Omega : |f(x)| \geq |g(x)|\}$$

luego necesariamente

$$\Omega \setminus A = \{x \in \Omega : |f(x)| < |g(x)|\}$$

Esto nos permite dar dos cotas:

$$|f(x) + g(x)| \leq |f(x)| + |g(x)| \leq 2|f(x)| \quad \forall x \in A$$

1. Fundamento

$$|f(x) + g(x)| \leq |f(x)| + |g(x)| < 2|g(x)| \quad \forall x \in \Omega \setminus A$$

Así,

$$\begin{aligned} \int_{\Omega} |f + g|^p &= \int_A |f + g|^p + \int_{\Omega \setminus A} |f + g|^p \\ &\leq 2^p \int_A |f|^p + 2^p \int_{\Omega \setminus A} |g|^p \\ &\leq 2^p \int_A |f|^p + 2^p \int_{\Omega} |g|^p < +\infty \end{aligned}$$

Luego, en efecto, $f + g \in \mathcal{L}_p(\Omega)$.

Podemos continuar, y ahora advertimos que el caso $\int_{\Omega} |f + g|^p = 0$ resulta trivial, así que supongamos $\int_{\Omega} |f + g|^p > 0$. Tenemos

$$|f + g|^p = |f + g| |f + g|^{p-1} \leq |f| |f + g|^{p-1} + |g| |f + g|^{p-1}$$

Integrando en ambos miembros y aplicando la desigualdad de Hölder 1.4 (lo que justificamos tomando $q \in \mathbb{R} / \frac{1}{p} + \frac{1}{q} = 1$ y notando que entonces $q(p-1) = p$, luego $(f + g)^{p-1} \in \mathcal{L}_q(\Omega)$),

$$\begin{aligned} \int_{\Omega} |f + g|^p &\leq \int_{\Omega} (|f| |f + g|^{p-1}) + \int_{\Omega} (|g| |f + g|^{p-1}) \\ &\leq \left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} \left(\int_{\Omega} |f + g|^{q(p-1)} \right)^{\frac{1}{q}} + \left(\int_{\Omega} |g|^p \right)^{\frac{1}{p}} \left(\int_{\Omega} |f + g|^{q(p-1)} \right)^{\frac{1}{q}} \\ &= \left(\left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} + \left(\int_{\Omega} |g|^p \right)^{\frac{1}{p}} \right) \left(\int_{\Omega} |f + g|^{q(p-1)} \right)^{\frac{1}{q}} \\ &= \left(\left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} + \left(\int_{\Omega} |g|^p \right)^{\frac{1}{p}} \right) \left(\int_{\Omega} |f + g|^p \right)^{\frac{1}{q}} \end{aligned}$$

Bajo la última suposición, resulta correcto dividir ambos miembros de la desigualdad por la cantidad positiva $(\int_{\Omega} |f + g|^p)^{\frac{1}{q}}$, quedando

$$\left(\int_{\Omega} |f + g|^p \right)^{1 - \frac{1}{q}} \leq \left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} + \left(\int_{\Omega} |g|^p \right)^{\frac{1}{p}}$$

Pero $1 - \frac{1}{q} = \frac{1}{p}$.

■

Está todo preparado para la definición de los espacios que buscamos, a falta de arrojar un poco de luz sobre la aplicación que sigue, dado $p \in \mathbb{R}$, $p \geq 1$.

$$\varphi_p : \mathcal{L}_p(\Omega) \rightarrow \mathbb{R}_0^+ / \varphi_p(f) := \left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} \quad \forall f \in \mathcal{L}_p(\Omega)$$

Percatémonos, por un lado, de que

$$\varphi_p(\lambda f) = \left(\int_{\Omega} |\lambda f|^p \right)^{\frac{1}{p}} = |\lambda| \left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} = |\lambda| \varphi_p(f) \quad \forall f \in \mathcal{L}_p(\Omega), \quad \forall \lambda \in K$$

Y, por otro, de que la desigualdad de Minkowski 1.5 presenta, inmediata, la desigualdad triangular para φ_p ; es decir, que

$$\varphi_p(f + g) \leq \varphi_p(f) + \varphi_p(g) \quad \forall f, g \in \mathcal{L}_p(\Omega)$$

Lo que convierte a φ_p en una seminorma en $\mathcal{L}_p(\Omega)$.

No obstante, jamás podría ser una norma completa, si bien dada $f \in \mathcal{L}_p(\Omega)$,

$$\varphi_p(f) = 0 \Leftrightarrow \left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} = 0 \Leftrightarrow |f|^p = 0 \text{ c.p.d.} \Leftrightarrow f = 0 \text{ c.p.d.}$$

Y esto no implica necesariamente que f se anule en todo punto.

Para arreglar esto acudimos al conjunto de funciones medibles

$$\mathcal{N}(\Omega) := \{f \in \mathcal{L}_p(\Omega) : f = 0 \text{ c.p.d.}\}$$

Conviene apreciar que $\mathcal{N}(\Omega)$ no es sino el núcleo de la seminorma φ_p y un subespacio vectorial de $\mathcal{L}_p(\Omega)$, lo que justifica la definición que sigue pues, a continuación, manifestamos los espacios de Lebesgue.

Definición 1.15. Sea $\Omega \subseteq \mathbb{R}$ un conjunto medible con medida de Lebesgue no nula. Llamaremos, dado $p \in \mathbb{R}$, $p \geq 1$, **espacio de Lebesgue** $L_p(\Omega)$ al espacio vectorial cociente

$$L_p(\Omega) := \mathcal{L}_p(\Omega) / \mathcal{N}(\Omega) = \{f + \mathcal{N}(\Omega) : f \in \mathcal{L}_p(\Omega)\}$$

Pondremos $[f] := f + \mathcal{N}(\Omega) \quad \forall f \in \mathcal{L}_p(\Omega)$

Así descritos, dos representantes en $\mathcal{L}_p(\Omega)$ de un mismo elemento de $L_p(\Omega)$ son diferentes tan solo en un conjunto de medida nula. En consecuencia, sus integrales de Lebesgue son idénticas, y así ocurre igualmente con las evaluaciones realizadas respecto a φ_p , lo que motiva y justifica la definición siguiente.

Definición 1.16. Definimos, sobre el espacio de Lebesgue $L_p(\Omega)$, la aplicación

$$\|\cdot\|_p : L_p(\Omega) \rightarrow \mathbb{R}_0^+ / \|[f]\|_p := \varphi_p(f) = \left(\int_{\Omega} |f|^p \right)^{\frac{1}{p}} \quad \forall [f] \in L_p(\Omega)$$

Esta aplicación está bien definida en virtud de lo mencionado anteriormente, por ser idénticas las evaluaciones sobre la seminorma de representantes distintos de un mismo elemento. Y, de facto, por ser φ_p seminorma, $\|\cdot\|_p$ también lo es. Pero, además,

$$\|[f]\|_p = 0 \Leftrightarrow f \in \mathcal{N}(\Omega) \Leftrightarrow [f] = [0]$$

Así, se concluye que $\|\cdot\|_p$ es una norma de pleno derecho en $L_p(\Omega)$ y eleva estos a la categoría de espacios normados.

1. Fundamento

Asociado a estos suele considerarse otro espacio vectorial más al que hacemos mención brevemente ahora.

Definición 1.17. Se define

$$\begin{aligned}\mathcal{L}_\infty(\Omega) &:= \{f \in \mathcal{L}(\Omega) : f \text{ está esencialmente acotada}\} \\ &= \{f \in \mathcal{L}(\Omega) : \exists M > 0 / |f(x)| \leq M \text{ c.p.d.}\}\end{aligned}$$

Por ser igualmente $\mathcal{N}(\Omega)$ subespacio vectorial de $\mathcal{L}_\infty(\Omega)$, ponemos

$$L_\infty(\Omega) := \mathcal{L}_\infty(\Omega) / \mathcal{N}(\Omega) = \{f + \mathcal{N}(\Omega) : f \in \mathcal{L}_\infty(\Omega)\}$$

Y, finalmente, puede verse que la aplicación

$$\| \cdot \|_\infty : L_\infty(\Omega) \rightarrow \mathbb{R}_0^+ / \| [f] \|_\infty := \inf \{ M > 0 : |f(x)| \leq M \text{ c.p.d.} \} \quad \forall [f] \in L_\infty(\Omega)$$

es una norma en este último.

Veamos a continuación una propiedad esencial de los espacios de Lebesgue, no sin antes recordar un resultado que utilizaremos con asiduidad.

Teorema 1.4. (Teorema de la convergencia dominada). Sean $\Omega \subseteq \mathbb{R}$ un conjunto medible y $\{f_n\}_{n \in \mathbb{N}}$ una sucesión de funciones integrables de Ω en \mathbb{C} que converge puntualmente a una función medible f . Sea luego g una función integrable verificando $|f_n| \leq g$ en Ω . Entonces f es integrable y se cumple

$$\int_\Omega f = \lim_{n \rightarrow +\infty} \int_\Omega f_n$$

Ahora sí,

Teorema 1.5. Dados $\Omega \subseteq \mathbb{R}$ un conjunto medible y $p \in \mathbb{R}$, $p \geq 1$, el espacio $L_p(\Omega)$ es completo y, por tanto, un espacio de Banach.

Demostración. Consideremos $f_n \in \mathcal{L}_p(\Omega)$ para cada $n \in \mathbb{N}$ de manera que $\{[f_n]\}_{n \in \mathbb{N}}$ sea sucesión de Cauchy en $L_p(\Omega)$ y veamos que converge en este espacio. Para ello, tan solo es preciso probar que admite una sucesión parcial convergente en virtud de la proposición 1.3. Por ser $\{[f_n]\}_{n \in \mathbb{N}}$ de Cauchy, dado $n \in \mathbb{N}$, $\exists a_n \in \mathbb{N}$ verificando que, $\forall m, h \in \mathbb{N}$, $m \geq a_n$,

$$\varphi_p(f_{m+h} - f_m) = \|[f_{m+h}] - [f_m]\|_p < \frac{1}{2^n}$$

De esta manera, puede definirse una aplicación

$$\sigma : \mathbb{N} \longrightarrow \mathbb{N} / \sigma(1) := a_1, \sigma(n+1) := \max\{a_{n+1}, \sigma(n) + 1\} \quad \forall n \in \mathbb{N}$$

que trivialmente resulta estrictamente creciente, siendo así $\{[f_{\sigma(n)}]\}_{n \in \mathbb{N}}$ la sucesión parcial de $\{[f_n]\}_{n \in \mathbb{N}}$ que trabajaremos y que, en pro de lo buscado cumple, por ser $\sigma(n+1) > \sigma(n) \geq a_n \quad \forall n \in \mathbb{N}$ y en virtud de la condición anterior,

$$\varphi_p(f_{\sigma(n+1)} - f_{\sigma(n)}) = \|[f_{\sigma(n+1)}] - [f_{\sigma(n)}]\|_p < \frac{1}{2^n} \quad \forall n \in \mathbb{N}$$

Definimos ahora, por conveniencia, $f_{\sigma(0)} = 0$ la aplicación idénticamente nula en $\mathcal{L}_p(\Omega)$. También consideraremos

$$g_n : \mathcal{L}_p(\Omega) \longrightarrow \mathbb{R}_0^+ / g_n := \sum_{k=1}^n |f_{\sigma(k)} - f_{\sigma(k-1)}| \quad \forall n \in \mathbb{N}$$

sucesión creciente de funciones medibles no negativas. Hacemos hincapié en la buena definición de estas aplicaciones, si bien $|f_{\sigma(k)} - f_{\sigma(k-1)}| \in \mathcal{L}_p(\Omega)$ por $\varphi_p(|f_{\sigma(k)} - f_{\sigma(k-1)}|) = \varphi_p(f_{\sigma(k)} - f_{\sigma(k-1)}) < \frac{1}{2^{k-1}} \forall k \in \mathbb{N}$.

Pondremos

$$g : \Omega \longrightarrow [0, +\infty] / g := \lim_{n \rightarrow +\infty} g_n = \sum_{k=1}^{\infty} |f_{\sigma(k)} - f_{\sigma(k-1)}|$$

que se trata igualmente de una función medible. Notemos que $\{g_n\} \nearrow g$ por definición.

Ahora, de la desigualdad triangular que cumple la seminorma φ_p y de que $\sum_{k=1}^{\infty} \frac{1}{2^k} = 1$ es suma de una serie estrictamente creciente, extraemos

$$\varphi_p(g_n) \leq \sum_{k=1}^n \varphi_p(|f_{\sigma(k)} - f_{\sigma(k-1)}|) = \varphi_p(f_{\sigma(1)}) + \sum_{k=2}^n \frac{1}{2^{k-1}} < \varphi_p(f_{\sigma(1)}) + 1 \quad \forall n \geq 2, n \in \mathbb{N}$$

pero de igual forma

$$\varphi_p(g_1) = \varphi_p(|f_{\sigma(1)}|) = \varphi_p(f_{\sigma(1)}) < \varphi_p(f_{\sigma(1)}) + 1$$

luego

$$\varphi_p(g_n) < \varphi_p(f_{\sigma(1)}) + 1 \iff \int_{\Omega} |g_n|^p = \varphi_p(f)^p < (\varphi_p(f_{\sigma(1)}) + 1)^p \quad \forall n \in \mathbb{N}$$

Considerando el hecho de que $\{|g_n|^p\} \nearrow |g|^p$ y el Teorema de la convergencia monótona, deducimos que

$$\int_{\Omega} |g|^p = \lim_{n \rightarrow +\infty} \int_{\Omega} |g_n|^p \leq (\varphi_p(f_{\sigma(1)}) + 1)^p$$

Por tanto, como la integral de g es finita, forzosamente la propia función debe ser finita *c.p.d.*; esto es, el conjunto $A = \{x \in \Omega : g(x) < +\infty\}$ es tal que $\Omega \setminus A$ es de medida nula. Equivalentemente, se puede afirmar que la serie $\sum_{k=1}^{\infty} f_{\sigma(k)} - f_{\sigma(k-1)}$ converge absolutamente en \mathbb{C} y, así, converge puntualmente en dicho cuerpo sobre el conjunto A .

Todo esto nos posibilita definir la función medible

$$f : \Omega \longrightarrow \mathbb{C} / f := \sum_{k=1}^{\infty} f_{\sigma(k)} - f_{\sigma(k-1)}$$

que es finita en A .

Como, por otro lado, claramente

$$f_{\sigma(n)} = \sum_{k=1}^n f_{\sigma(k)} - f_{\sigma(k-1)}$$

es inmediato que

$$f = \lim_{n \rightarrow +\infty} f_{\sigma(n)}$$

1. Fundamento

y, por desigualdad triangular,

$$|f_{\sigma(n)}| \leq \sum_{k=1}^n |f_{\sigma(k)} - f_{\sigma(k-1)}| \leq g \iff |f_{\sigma(n)}|^p \leq |g|^p$$

Aplicando de nuevo el Teorema de la convergencia dominada y que $\{|f_{\sigma(n)}|^p\} \rightarrow |f|^p$,

$$\int_{\Omega} |f|^p = \lim_{n \rightarrow +\infty} \int_{\Omega} |f_{\sigma(n)}|^p \leq \int_{\Omega} |g|^p \leq (\varphi_p(f_{\sigma(1)}) + 1)^p$$

de donde $f \in \mathcal{L}_p(\Omega)$. Tan solo resta comprobar que $\{[f_{\sigma(n)}]\} \rightarrow [f]$ o lo que es lo mismo, que $\|[f] - [f_{\sigma(n)}]\|_p = \varphi_p(f - f_{\sigma(n)}) \rightarrow 0$.

Apreciemos que $|f - f_{\sigma(n)}|^p \rightarrow 0$ en A . Esto, sumado a que $|f| + g \in \mathcal{L}_p(\Omega)$ y

$$|f - f_{\sigma(n)}|^p \leq (|f| + g)^p \quad \forall n \in \mathbb{N}$$

nos permite aplicar una última vez el Teorema de la convergencia monótona para concluir que

$$\lim_{n \rightarrow +\infty} \varphi_p(f - f_{\sigma(n)})^p = \lim_{n \rightarrow +\infty} \int_{\Omega} |f - f_{\sigma(n)}|^p = \int_{\Omega} \lim_{n \rightarrow +\infty} |f - f_{\sigma(n)}|^p = 0$$

■

Finalmente, veremos un último resultado general que nos permitirá observar la estrecha relación que guardan estos espacios vectoriales entre sí.

Proposición 1.6. Sea $\Omega \subseteq \mathbb{R}$ un conjunto medible de medida finita; es decir, $\lambda(\Omega) < +\infty$. Entonces, dados $p, q \in [1, +\infty[$, $p \leq q$, se tiene $L_q(\Omega) \subseteq L_p(\Omega)$.

Demostración. Sean $f \in \mathcal{L}_q(\Omega)$ arbitraria y χ_{Ω} la función característica en Ω .

Advirtamos en primer lugar algunos hechos relevantes para la prueba, en virtud de las hipótesis asumidas:

- $\|[f]\|_q < +\infty \Rightarrow \|[f]\|_q^{qr} = (\int_{\Omega} |f|^q)^r < +\infty \quad \forall r \in \mathbb{R}$.
- $\|\chi_{\Omega}\|_r = (\int_{\Omega} |\chi_{\Omega}|^r)^{\frac{1}{r}} = \lambda(\Omega)^{\frac{1}{r}} < +\infty \quad \forall r \in [1, +\infty[\Rightarrow \chi_{\Omega} \in \mathcal{L}_r(\Omega) \quad \forall r \in [1, +\infty[$.
- $g\chi_{\Omega} = g$ en $\Omega \quad \forall g \in \mathcal{L}(\Omega)$.
- $\frac{1}{p} + \frac{1}{q-p} = 1$

Así,

$$\begin{aligned} \|[f]\|_p^p &= \int_{\Omega} |f|^p = \|[f^p]\|_1 = \|[f^p]\chi_{\Omega}\|_1 = \|[f^p][\chi_{\Omega}]\|_1 \\ &\stackrel{\text{Hölder}}{\leq} \|[f^p]\|_{\frac{q}{p}} \|\chi_{\Omega}\|_{\frac{q}{q-p}} = \left(\int_{\Omega} |f|^q \right)^{\frac{p}{q}} \lambda(\Omega)^{\frac{q-p}{q}} < +\infty \end{aligned}$$

Luego $\|[f]\|_p < +\infty \Rightarrow [f] \in L_p(\Omega)$.

■

Observación 1.12. Aunque no vamos a detenernos a demostrarlo, la anterior propiedad constituye, en verdad, una caracterización.

1.2.1. El espacio de Lebesgue $L_2(\Omega)$

A lo largo del presente trabajo, dirigiremos nuestra mirada a un espacio de Lebesgue en especial: el espacio $L_2(\Omega)$. Y la razón es sencilla, pues él, de entre todos los demás, es el único que permite definir un producto escalar, como bien indica la siguiente

Proposición 1.7. Sea $\Omega \subseteq \mathbb{R}$ un conjunto medible. La aplicación $\langle \cdot, \cdot \rangle : L_2(\Omega) \times L_2(\Omega) \rightarrow \mathbb{C}$ dada por

$$\langle [f], [g] \rangle := \int_{\Omega} f \bar{g} \quad \forall [f], [g] \in L_2(\Omega)$$

es un producto escalar en este espacio de Lebesgue.

Demostración. Veamos, primeramente, que tal aplicación está bien definida. Para ello, tomamos dos elementos $f, g \in \mathcal{L}_2(\Omega)$ arbitrarios y observamos que

$$(|f| - |g|)^2 \geq 0 \iff |f|^2 + |g|^2 - 2|fg| \geq 0 \iff |f\bar{g}| = |fg| \leq \frac{|f|^2 + |g|^2}{2} \in \mathcal{L}_1(\Omega)$$

Así, $|f\bar{g}| \in \mathcal{L}_1(\Omega) \Rightarrow f\bar{g} \in \mathcal{L}_1(\Omega)$. Además, la integral de Lebesgue no depende de los representantes elegidos.

Ahora, poniendo $f, g, h \in \mathcal{L}_2(\Omega)$, $\lambda, \mu \in \mathbb{C}$ arbitrarios, pasamos a probar los requisitos de producto escalar:

- $\langle [f], [f] \rangle = \int_{\Omega} f \bar{f} = \int_{\Omega} |f|^2 \geq 0$ pues $|f|^2 \geq 0$; además, $\langle [f], [f] \rangle = 0 \Leftrightarrow \int_{\Omega} |f|^2 = 0 \Leftrightarrow |f|^2 = 0 \text{ c.p.d.} \Leftrightarrow f = 0 \text{ c.p.d.} \Leftrightarrow [f] = [0]$.
- $\langle [f], [g] \rangle = \int_{\Omega} f \bar{g} = \int_{\Omega} \overline{g \bar{f}} = \overline{\int_{\Omega} g \bar{f}} = \overline{\langle [g], [f] \rangle}$.
- $\langle \lambda[f] + \mu[g], [h] \rangle = \langle [\lambda f + \mu g], [h] \rangle = \int_{\Omega} (\lambda f + \mu g) \bar{h} = \int_{\Omega} (\lambda f \bar{h} + \mu g \bar{h}) = \lambda \int_{\Omega} f \bar{h} + \mu \int_{\Omega} g \bar{h} = \lambda \langle [f], [h] \rangle + \mu \langle [g], [h] \rangle$.

■

Observación 1.13. Conviene percatarse de que la norma inducida por este producto escalar no es otra que la norma usual definida para este espacio en la sección anterior, $\|\cdot\|_2$.

Nota: Será más cómodo trabajar a partir de ahora con los elementos de estos espacios como si de funciones se tratara, en lugar de como clases de equivalencia. Adoptaremos, por tanto, este abuso del lenguaje y de notación y consideraremos "funciones" $f \in L_p(\Omega)$.

1.3. Series de Fourier en $L_2([0, T])$

Si bien el escenario más global de definición de la herramienta que presentamos ahora se trata del de los espacios L_1 , las propiedades que enmarcan a los espacios L_2 hacen de esta un poderosísimo artilugio para tratar y caracterizar clases de funciones, así como para condensar, llegado el momento, la información que identifica a una clase concreta.

En lo sucesivo estudiaremos lo concerniente al espacio $L_2([0, T])$ definido sobre el intervalo compacto de \mathbb{R} $[0, T]$, $T > 0$.

1. Fundamento

Señalaremos, en primer lugar, las funciones

$$\begin{aligned} e_k : [0, T] &\longrightarrow \mathbb{C} \\ x &\longrightarrow e^{i\frac{2\pi kx}{T}} \end{aligned}$$

para cada $k \in \mathbb{Z}$.

Notemos que

$$\left(\int_0^T |e^{i\frac{2\pi kx}{T}}|^2 dx \right)^{\frac{1}{2}} = \left(\int_0^T dx \right)^{\frac{1}{2}} = \sqrt{T} \quad \forall k \in \mathbb{Z}$$

luego $\|e_k\|_2 = \sqrt{T} \quad \forall k \in \mathbb{Z}$ y se tiene $e_k \in L_2([0, T])$. Por tanto,

Definición 1.18. Definimos la familia de funciones $\{\phi_k\}_{k \in \mathbb{Z}}$ en $L_2([0, T])$ dada por

$$\begin{aligned} \phi_k : [0, T] &\longrightarrow \mathbb{C} \\ x &\longrightarrow \frac{e_k(x)}{\|e_k(x)\|_2} = \frac{1}{\sqrt{T}} e^{i\frac{2\pi kx}{T}} \end{aligned}$$

para cada $k \in \mathbb{Z}$.

A la familia $\{\phi_k\}_{k \in \mathbb{Z}}$ se la conoce por el nombre de **sistema trigonométrico** en $L_2([0, T])$.

A pesar de que aún no se ha descrito su funcionalidad, resulta necesario invocar los coeficientes de Fourier, y más tarde daremos cuenta de su significado.

Definición 1.19. Dado $k \in \mathbb{Z}$, se define el k -ésimo coeficiente de Fourier de f como

$$\hat{f}(k) := \frac{\langle f, e_k \rangle}{\|e_k\|^2} = \frac{1}{T} \int_0^T f(x) e^{-i\frac{2\pi kx}{T}} dx$$

Enunciamos, seguidamente, un importante teorema sobre el que nos apoyaremos ahora.

Teorema 1.6. (Teorema de unicidad). Sean $f, g \in L_1([0, T])$ verificando que $\hat{f}(k) = \hat{g}(k) \quad \forall k \in \mathbb{Z}$. Entonces $f = g$ c.p.d.

Pues resulta crucial probar el siguiente

Teorema 1.7. El sistema trigonométrico en $L_2([0, T])$ es una base de Hilbert de este espacio.

Demostración. En virtud de lo que acabamos de ver, claramente $\|\phi_k\|_2 = 1 \quad \forall k \in \mathbb{Z}$. Por otro lado, dados $p, q \in \mathbb{Z}$ con $p \neq q$ arbitrarios, se tiene

$$\begin{aligned} \langle \phi_p, \phi_q \rangle &= \int_0^T \phi_p(x) \overline{\phi_q(x)} dx \\ &= \int_0^T \frac{1}{\sqrt{T}} e^{i\frac{2\pi px}{T}} \frac{1}{\sqrt{T}} e^{-i\frac{2\pi qx}{T}} dx \\ &= \frac{1}{T} \int_0^T e^{i\frac{2\pi(p-q)x}{T}} dx \\ &= \frac{1}{T} \left[\frac{e^{i\frac{2\pi(p-q)x}{T}}}{i\frac{2\pi(p-q)}{T}} \right]_0^T = 0 \end{aligned}$$

luego $\{\phi_k\}$ es un sistema ortonormal en $L_2([0, T])$.

Tan solo resta demostrar que $S = \overline{L(\{\phi_k\}_{k \in \mathbb{Z}})} = L_2([0, T])$. Para ello, consideremos $f \in S^\perp$ y notemos entonces que, en particular,

$$\langle f, \phi_k \rangle = 0 \Rightarrow \langle f, e_k \rangle = 0 \Rightarrow \widehat{f}(k) = 0 \quad \forall k \in \mathbb{Z}$$

En consecuencia, el Teorema de unicidad 1.6 indica que $f = 0$ c.p.d. y, por tanto, que $S^\perp = \{0\}$ subespacio vectorial de $L_2([0, T])$ de dimensión 0.

Y, además, S es un subespacio vectorial de $L_2([0, T])$ a causa de que, dados $f, g \in S$, $\lambda \in \mathbb{C}$, $\exists \{f_n\}_{n \in \mathbb{N}}, \{g_n\}_{n \in \mathbb{N}}$ sucesiones de elementos de $L(\{\phi_k\}_{k \in \mathbb{Z}})$ convergentes a f y g , respectivamente, luego $\{\lambda f_n + g_n\}_{n \in \mathbb{N}}$ es una sucesión de elementos de $L(\{\phi_k\}_{k \in \mathbb{Z}})$ que converge a $\lambda f + g \in L_2([0, T]) \Rightarrow \lambda f + g \in S$.

Así, rematamos la prueba mediante el Teorema de la proyección ortogonal 1.1, que indica que $L_2([0, T]) = S \oplus S^\perp = S \oplus \{0\} \Rightarrow S = L_2([0, T])$. ■

Por tanto, como consecuencia final del Teorema 1.3, obtenemos el corolario que sigue.

Corolario 1.1. *Dada una función $f \in L_2([0, T])$, se verifica que la serie*

$$\sum_{k \in \mathbb{Z}} \langle f, \phi_k \rangle \phi_k$$

converge en norma 2 a f en dicho espacio; es decir,

$$f = \sum_{k=-\infty}^{+\infty} \langle f, \phi_k \rangle \phi_k = \sum_{k=-\infty}^{+\infty} \frac{\langle f, e_k \rangle}{\|e_k\|^2} e_k = \sum_{k=-\infty}^{+\infty} \frac{1}{T} \int_0^T f(x) e^{-i \frac{2\pi k x}{T}} dx e_k = \sum_{k=-\infty}^{+\infty} \widehat{f}(k) e_k$$

*A esta serie se la conoce por el nombre de **serie de Fourier** de f .*

Consecuentemente, entendidos f y la suma de su serie de Fourier como elementos de $L_2([0, T])$, podemos decir que

$$f(x) = \sum_{k=-\infty}^{+\infty} \widehat{f}(k) e^{i \frac{2\pi k x}{T}} \text{ c.p.d. en } [0, T]$$

Observación 1.14. Nótese que, asimismo, cada $f \in L_2([0, T])$ queda biunívocamente caracterizada por sus coeficientes de Fourier, que se trata de un conjunto numerable de valores. Esto resulta sorprendente en tanto que encerramos elementos de naturaleza aparentemente no numerable en un conjunto de medida nula.

Además, este hecho permite definir una isometría entre los espacios $L_2([0, T])$ y l_2 , siendo este último el espacio de sucesiones reales de cuadrado sumable, bien conocido y con mayor profundidad estudiado en la asignatura *Análisis Funcional* del Grado.

En lo que respecta al Análisis Funcional y de Fourier, se remite al lector a las referencias [Duo03], [Fol92], [Mac09] y [PAo8].

2. Discretización

Una vez desarrolladas las herramientas necesarias para trabajar con nuestras señales, es preciso discutir hasta qué punto resulta legítimo tratarlas computacionalmente, si bien la teoría se fundamenta sobre elementos extremadamente abstractos (clases de equivalencia) que, para más inri, habrían de ser representados con una cantidad finita de datos. Justificar su utilización en este ambiente es tarea del siguiente capítulo.

2.1. Planteamiento

Tal y como se plantea en el preámbulo, probar la fidelidad de los procedimientos prácticos con respecto a la teoría analítica resulta de una importancia vital.

Sea una señal T -periódica $f : \mathbb{R} \rightarrow \mathbb{C}$, para $T > 0$, verificando que $f|_{[0,T]} \in L_2([0,T])$. Planteamos el problema siguiente: extraer sus coeficientes de Fourier, pero en ambiente computacional. Esto significa que no conocemos f en su totalidad; a lo sumo, dispondremos en su lugar de un conjunto finito de evaluaciones suyas.

Esto supone un obstáculo que, de forma alguna, hemos de sortear. Sin embargo, podría argumentarse, el hecho de que el conjunto de datos que preservamos de f sea finito induce por completo al error si pensamos que, en análisis de Fourier, trabajamos equiparando funciones coincidentes *c.p.d.*; es decir, partiendo de un número finito de evaluaciones podemos hallarnos ante cualquier función imaginable en dicho espacio.

Ante este problema propondremos realizar la discretización de la señal por medio de una aproximación por funciones constantes a trozos. Estudiaremos bajo qué condiciones se trabaja con soltura y qué escenarios ameritan proceder de manera más sofisticada.

En primer lugar, proponemos el planteamiento inicial. Consideraremos, dado $N \in \mathbb{N}$, la partición equiespaciada $P_N = \{0 < \frac{T}{N} < \frac{2T}{N} < \dots < T\}$ del intervalo de trabajo $[0, T]$. Denotaremos por $I_k^N = [\frac{(k-1)T}{N}, \frac{kT}{N}] \quad \forall k \in \{1, 2, \dots, N-1\}$, $I_N^N = [\frac{(N-1)T}{N}, T]$, a los intervalos que conforman dicha partición (la apertura por la derecha es deliberada, por conveniencia) y tomaremos un único $x_k^N \in I_k$ para cada $k \in \{1, 2, \dots, N\}$. Finalmente, supondremos conocidos los valores $f(x_k^N) \quad \forall k \in \{1, 2, \dots, N\}$. Notemos que N es el número de muestras tomadas de nuestra señal.

Con esto, definiremos las funciones f_N T -periódicas y constantes a trozos de la siguiente manera:

$$f_N : \mathbb{R} \rightarrow \mathbb{C} / f_N(x) = f(x_k^N) \text{ si } x \in I_k^N$$

Llegados a este punto, cabe observar por qué estamos permitiendo evaluar f si precisamente en los espacios cociente esto no tiene sentido alguno. Pues bien, es que realmente es así. En verdad, puesto que de lo que partimos es de N datos, nada impediría afirmar que

2. Discretización

la función que estamos estudiando no es otra que f_N ; en cuyo caso, habríamos acabado al disponer directamente de una definición diáfana de la señal. Este enfoque es, sin embargo, demasiado simple.

En la realidad, la señal periódica f tiene una forma concreta, $f|_{[0,T]} \in \mathcal{L}_2(\Omega)$. Esto nos permite evaluar en la señal, de manera que podemos decir que los datos del problema son tales evaluaciones y, con ellas, pondremos como meta tratar de aproximar f tanto como queramos. Luego, obtenidas las funciones aproximantes, elevaremos el proceso a $L_2([0, T])$ para aplicar Fourier en las respectivas clases de equivalencia, respaldando estos procedimientos con las justificaciones teóricas que siguen.

2.2. Aproximaciones

Como veníamos diciendo, suponer que f_N es, en sí misma, la señal, es un enfoque válido, si bien un tanto simplista. Así, estudiemos la bondad de estas aproximaciones constantes a trozos en caso de hallarnos ante diferentes tipos de señal.

No resulta extraño pensar en señales de carácter continuo, pues la naturaleza suele actuar de manera igualmente continua. Asimismo, gran número de señales que analicemos, más aún en el campo de la música, verificarán esta sencilla propiedad. Por tanto, conviene observar que, de forma inmediata, las aproximaciones propuestas resultan excelentes si trabajamos con funciones continuas.

Recordemos un resultado clásico del análisis matemático.

Teorema 2.1. (De Heine). Sea $f : K \longrightarrow \mathbb{C}$ una aplicación continua definida sobre K espacio métrico compacto. Entonces se verifica que f es uniformemente continua.

En consecuencia, tal y como queríamos, veamos que

Proposición 2.1. Sean $f : \mathbb{R} \longrightarrow \mathbb{C}$ una señal T -periódica continua, $T > 0$ y f_N las aplicaciones anteriormente definidas, para cada $N \in \mathbb{N}$. Entonces la sucesión $\{f_N\}$ converge a f en norma 2.

Demostración. Probaremos, equivalentemente, que $\|f - f_N\|_2 \longrightarrow 0$.

Puesto que f es continua, el Teorema de Heine implica que f es uniformemente continua en el compacto $[0, T]$. Entonces, por definición, dado $\epsilon > 0$, $\exists \delta_\epsilon > 0 / \forall x, y \in [0, T], |x - y| < \delta_\epsilon$, se tiene $|f(x) - f(y)| < \frac{\epsilon}{\sqrt{T}}$.

También hemos de advertir que $\exists N_\epsilon / \forall N \geq N_\epsilon$ se cumple $\frac{T}{N} < \delta_\epsilon$. Por tanto, $\forall N \geq N_\epsilon$ se verifica $|f(x) - f(x_i^N)| < \frac{\epsilon}{\sqrt{T}}$ por ser $|x - x_i^N| < \frac{T}{N} < \delta_\epsilon$, luego

$$\begin{aligned} \|f - f_N\|_2^2 &= \int_0^T |f(x) - f_N(x)|^2 dx \\ &= \sum_{i=1}^N \int_{(i-1)\frac{T}{N}}^{i\frac{T}{N}} |f(x) - f(x_i^N)|^2 dx \\ &< \sum_{i=1}^N \frac{\epsilon^2}{T} \frac{T}{N} = \epsilon^2 \end{aligned}$$

de donde $\|f - f_N\|_2 < \epsilon$.

■

Observación 2.1. Esta proposición implica directamente que $\{f_N\} \rightarrow f$ puntualmente *c.p.d.*

Observación 2.2. Nótese que este resultado no depende de la elección de las abscisas x_k^N .

Lo que desfigura, no obstante, el ambiente no continuo, podemos solventarlo de manera sencilla conociendo el siguiente hecho.

Proposición 2.2. *El espacio $C_c(]0, T[)$ de las funciones continuas con soporte compacto en $]0, T[$ es denso en $L_2(]0, T[) = L_2([0, T])$.*

Y se han de realizar una serie de observaciones muy relevantes a este respecto. Dada $g \in C_c(]0, T[)$, se verifica que g es continua y de soporte compacto; es decir,

$$\text{supp}(g) = \overline{\{x \in]0, T[: g(x) \neq 0\}}$$

es acotado (luego compacto). Esto tiene una consecuencia fundamental, y es que debe existir forzosamente $\epsilon > 0$ de manera que g se anula en los abiertos $]0, \epsilon[$ y $]T - \epsilon, T[$, que a su vez implica que g puede extenderse de forma continua imponiendo $g(0) = g(T) = 0$ y, así, g admite una periodificación continua definida en todo \mathbb{R} sin mayor dificultad que definir $g(x + kT) = g(x) \quad \forall x \in [0, T[, \quad \forall k \in \mathbb{Z}$.

Asimismo, sin necesidad de suponer que f es continua, se sabe por la proposición anterior que $\exists \{g_n\}_{n \in \mathbb{N}}$ sucesión de funciones continuas y T -periódicas que converge a f en norma 2, lo cual a su vez permite concluir que existe convergencia puntual *c.p.d.* en \mathbb{R} .

Por tanto,

Proposición 2.3. *Sea $f : \mathbb{R} \rightarrow \mathbb{C}$ una señal T -periódica, $T > 0$, verificando que $f|_{[0, T]} \in L_2([0, T])$. Podemos aproximar f , en norma 2, tanto como se quiera, por las funciones constantes a trozos del planteamiento.*

Demostración. Basta tomar una sucesión $\{g_n\}_{n \in \mathbb{N}}$ de funciones continuas y T -periódicas que converjan a f en norma 2 y elegir una cota del error $\epsilon > 0$ tan pequeña como sea preciso. Por definición de convergencia, $\exists N_\epsilon \in \mathbb{N} / \|f - g_{N_\epsilon}\|_2 < \frac{\epsilon}{2}$. Y ahora aplicamos el procedimiento para funciones continuas a g_{N_ϵ} , encontrando una sucesión de funciones constantes a trozos $\{g_{N_\epsilon}^m\}_{m \in \mathbb{N}}$ que converge a g_{N_ϵ} en norma 2; esto es, $\exists M_\epsilon \in \mathbb{N} / \|g_{N_\epsilon} - g_{N_\epsilon}^{M_\epsilon}\|_2 < \frac{\epsilon}{2}$. Finalmente, por la desigualdad triangular,

$$\|f - g_{N_\epsilon}^{M_\epsilon}\|_2 = \|f - g_{N_\epsilon} + g_{N_\epsilon} - g_{N_\epsilon}^{M_\epsilon}\|_2 \leq \|f - g_{N_\epsilon}\|_2 + \|g_{N_\epsilon} - g_{N_\epsilon}^{M_\epsilon}\|_2 < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon$$

■

Corolario 2.1. *El espacio de las funciones constantes a trozos es denso en $L_2([0, T])$.*

Como puede verse, la idea fundamental es aplicar el proceso descrito en el planteamiento a una función continua que aproxime convenientemente a nuestra señal.

2.3. Coeficientes de Fourier aproximados y convergencia

Aun tras manipular toda esta maquinaria, todavía no hemos expuesto el resultado fundamental que da sentido finalmente al proceso.

2. Discretización

Proposición 2.4. Sea $f \in L_2([0, T])$ y $\{f_n\}_{n \in \mathbb{N}}$ una sucesión de elementos de $L_2([0, T])$ convergente a f en dicho espacio. Entonces, para cada $k \in \mathbb{Z}$, se cumple que $\{\widehat{f}_n(k)\} \rightarrow \widehat{f}(k)$.

Demostración. Basta advertir que

$$\begin{aligned}
 |\widehat{f}(k) - \widehat{f}_n(k)| &= \left| \frac{1}{T} \int_0^T f(x) e^{-i \frac{2\pi}{T} kx} dx - \frac{1}{T} \int_0^T f_n(x) e^{-i \frac{2\pi}{T} kx} dx \right| \\
 &= \frac{1}{T} \left| \int_0^T (f(x) - f_n(x)) e^{-i \frac{2\pi}{T} kx} dx \right| \\
 &\leq \frac{1}{T} \int_0^T |f(x) - f_n(x)| dx \\
 &= \frac{1}{T} \|f - f_n\|_1 \\
 &= \frac{1}{T} \|(f - f_n) \chi_{[0, T]}\|_1 \\
 &\stackrel{\text{Hölder}}{\leq} \frac{1}{T} \|f - f_n\|_2 \|\chi_{[0, T]}\|_2 \\
 &= \frac{1}{T} \|f - f_n\|_2 \sqrt{T} = \frac{1}{\sqrt{T}} \|f - f_n\|_2
 \end{aligned}$$

■

La consecuencia fundamental de esta proposición es que el k -ésimo coeficiente de Fourier de una aproximación de f aproxima al k -ésimo coeficiente de Fourier de f .

Por tanto, y puesto que nuestra intención es utilizar la sucesión de funciones constantes a trozos $\{f_N\}_{N \in \mathbb{N}}$ con el fin de aproximar f , de igual manera los coeficientes de Fourier de las funciones f_N tienen la potestad de hacer las veces de los coeficientes de f , lo cual motiva su cálculo explícito.

2.3. Coeficientes de Fourier aproximados y convergencia

Dado $k \in \mathbb{Z}^*$, para la función f_N se tiene

$$\begin{aligned}
 \widehat{f_N}(k) &= \frac{1}{T} \int_0^T f_N(x) e^{-i\frac{2\pi}{T}kx} dx \\
 &= \frac{1}{T} \sum_{j=1}^N \int_{\frac{(j-1)T}{N}}^{\frac{jT}{N}} f(x_j^N) e^{-i\frac{2\pi}{T}kx} dx \\
 &= \frac{1}{T} \sum_{j=1}^N f(x_j^N) \int_{\frac{(j-1)T}{N}}^{\frac{jT}{N}} e^{-i\frac{2\pi}{T}kx} dx \\
 &= \frac{1}{T} \sum_{j=1}^N f(x_j^N) \left[\frac{e^{-i\frac{2\pi}{T}kx}}{-i\frac{2\pi}{T}k} \right]_{\frac{(j-1)T}{N}}^{\frac{jT}{N}} \\
 &= \frac{T}{N} \sum_{j=1}^N f(x_j^N) \frac{iT}{2\pi k} \left(e^{-i\frac{2\pi}{T}k\frac{jT}{N}} - e^{-i\frac{2\pi}{T}k\frac{(j-1)T}{N}} \right) \\
 &= \sum_{j=1}^N f(x_j^N) \frac{i}{2\pi k} e^{-i2\pi k\frac{j}{N}} \left(1 - e^{i2\pi k\frac{1}{N}} \right)
 \end{aligned}$$

Y para $k = 0$,

$$\begin{aligned}
 \widehat{f_N}(0) &= \frac{1}{T} \int_0^T f_N(x) dx \\
 &= \frac{1}{T} \sum_{j=1}^N \int_{\frac{(j-1)T}{N}}^{\frac{jT}{N}} f(x_j^N) dx \\
 &= \frac{1}{T} \sum_{j=1}^N f(x_j^N) \int_{\frac{(j-1)T}{N}}^{\frac{jT}{N}} dx \\
 &= \frac{1}{T} \sum_{j=1}^N f(x_j^N) \frac{T}{N} \\
 &= \sum_{j=1}^N f(x_j^N) \frac{1}{N}
 \end{aligned}$$

Por tanto, finalmente obtenemos

$$\widehat{f_N}(k) = \langle (f(x_1^N), \dots, f(x_N^N)), \left(\frac{i}{2\pi k} e^{-i2\pi k\frac{1}{N}} \left(1 - e^{i2\pi k\frac{1}{N}} \right), \dots, \frac{i}{2\pi k} e^{-i2\pi k} \left(1 - e^{i2\pi k\frac{1}{N}} \right) \right) \rangle \quad \forall k \in \mathbb{Z}^*$$

$$\widehat{f_N}(0) = \langle (f(x_1^N), \dots, f(x_N^N)), \left(\frac{1}{N}, \dots, \frac{1}{N} \right) \rangle$$

Llamaremos

$$u_N(k) = \left(\frac{i}{2\pi k} e^{-i2\pi k\frac{1}{N}} \left(1 - e^{i2\pi k\frac{1}{N}} \right), \left(\frac{i}{2\pi k} e^{-i2\pi k\frac{2}{N}} \left(1 - e^{i2\pi k\frac{1}{N}} \right), \dots, \frac{i}{2\pi k} e^{-i2\pi k} \left(1 - e^{i2\pi k\frac{1}{N}} \right) \right) \right) \quad \forall k \in \mathbb{Z}^*$$

$$u_N(0) = \left(\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N} \right)$$

2.4. Paso a \mathbb{C}^N . Ortogonalidad y transformaciones

Como hemos visto, el producto de la sección anterior es el cálculo explícito de unos coeficientes de Fourier aproximados a través de un sencillo producto escalar entre las evaluaciones o los datos conocidos de nuestra señal, y unos vectores contenidos en \mathbb{C}^N y dependientes de k .

Por tanto, nuestro siguiente paso será comprobar un hecho sorprendente a la par que extremadamente útil, y es que

Proposición 2.5. *Fijado $N \in \mathbb{N}$, la familia $\{u_N(k) : k \in \{0, \dots, N-1\}\}$ de vectores en \mathbb{C}^N verifica que sus elementos son ortogonales dos a dos y que*

$$\|u_N(k)\|_2^2 = \frac{N}{\pi^2 k^2} \sin^2 \frac{\pi k}{N} \quad \forall k \in \{1, \dots, N-1\}$$

$$\|u_N(0)\|_2^2 = \frac{1}{N}$$

donde $\|\cdot\|_2$ denota la norma euclídea en \mathbb{C}^N .

Demostración. En efecto, dados $k, p, q \in \{1, \dots, N-1\}$ arbitrarios verificando $p \neq q$, se cumple

$$\begin{aligned} \langle u_N(p), u_N(q) \rangle &= \sum_{j=1}^N \frac{i}{2\pi p} e^{-i2\pi p \frac{j}{N}} \left(1 - e^{i2\pi p \frac{1}{N}}\right) \overline{\frac{i}{2\pi q} e^{-i2\pi q \frac{j}{N}} \left(1 - e^{i2\pi q \frac{1}{N}}\right)} \\ &= \sum_{j=1}^N \frac{i}{2\pi p} e^{-i2\pi p \frac{j}{N}} \left(1 - e^{i2\pi p \frac{1}{N}}\right) \frac{-i}{2\pi q} e^{i2\pi q \frac{j}{N}} \left(1 - e^{-i2\pi q \frac{1}{N}}\right) \\ &= \sum_{j=1}^N \frac{1}{4\pi^2 pq} \left(1 - e^{i2\pi p \frac{1}{N}}\right) \left(1 - e^{-i2\pi q \frac{1}{N}}\right) e^{i2\pi(q-p) \frac{j}{N}} \\ &= \frac{1}{4\pi^2 pq} \left(1 - e^{i2\pi p \frac{1}{N}}\right) \left(1 - e^{-i2\pi q \frac{1}{N}}\right) \sum_{j=1}^N e^{i2\pi(q-p) \frac{j}{N}} \\ &= \frac{1}{4\pi^2 pq} \left(1 - e^{i2\pi p \frac{1}{N}}\right) \left(1 - e^{-i2\pi q \frac{1}{N}}\right) \frac{e^{i2\pi(q-p) \frac{1}{N}} - e^{i2\pi(q-p) \frac{N+1}{N}}}{1 - e^{i2\pi(q-p) \frac{1}{N}}} \\ &= \frac{1}{4\pi^2 pq} \left(1 - e^{i2\pi p \frac{1}{N}}\right) \left(1 - e^{-i2\pi q \frac{1}{N}}\right) \frac{e^{i2\pi(q-p) \frac{1}{N}} - e^{i2\pi(q-p) \frac{1}{N}}}{1 - e^{i2\pi(q-p) \frac{1}{N}}} = 0 \end{aligned}$$

$$\begin{aligned} \langle u_N(k), u_N(0) \rangle &= \sum_{j=1}^N \frac{i}{2\pi k} e^{-i2\pi k \frac{j}{N}} \left(1 - e^{i2\pi k \frac{1}{N}}\right) \overline{\frac{1}{N}} \\ &= \frac{i}{2\pi Nk} \left(1 - e^{i2\pi k \frac{1}{N}}\right) \sum_{j=1}^N e^{-i2\pi k \frac{j}{N}} \\ &= \frac{i}{2\pi Nk} \left(1 - e^{i2\pi k \frac{1}{N}}\right) \frac{e^{-i2\pi k \frac{1}{N}} - e^{-i2\pi k \frac{N+1}{N}}}{1 - e^{-i2\pi k \frac{1}{N}}} \\ &= \frac{i}{2\pi Nk} \left(1 - e^{i2\pi k \frac{1}{N}}\right) \frac{e^{-i2\pi k \frac{1}{N}} - e^{-i2\pi k \frac{1}{N}}}{1 - e^{-i2\pi k \frac{1}{N}}} = 0 \end{aligned}$$

lo que prueba la ortogonalidad dos a dos. Por otro lado,

$$\begin{aligned}
 \|u_N(k)\|_2^2 &= \langle u_N(k), u_N(k) \rangle \\
 &= \sum_{j=1}^N \frac{i}{2\pi k} e^{-i2\pi k \frac{j}{N}} \left(1 - e^{i2\pi k \frac{1}{N}}\right) \overline{\frac{i}{2\pi k} e^{-i2\pi k \frac{j}{N}} \left(1 - e^{i2\pi k \frac{1}{N}}\right)} \\
 &= \sum_{j=1}^N \left| \frac{i}{2\pi k} e^{-i2\pi k \frac{j}{N}} \left(1 - e^{i2\pi k \frac{1}{N}}\right) \right|^2 \\
 &= \sum_{j=1}^N \frac{1}{4\pi^2 k^2} |1 - e^{i2\pi k \frac{1}{N}}|^2 \\
 &= \frac{N}{4\pi^2 k^2} \left(1 - e^{i2\pi k \frac{1}{N}}\right) \left(1 - e^{-i2\pi k \frac{1}{N}}\right) \\
 &= \frac{N}{4\pi^2 k^2} \left(2 - e^{i2\pi k \frac{1}{N}} - e^{-i2\pi k \frac{1}{N}}\right) \\
 &= \frac{N}{4\pi^2 k^2} \left(2 - 2 \cos \frac{2\pi k}{N}\right) \\
 &= \frac{N}{\pi^2 k^2} \frac{1 - \cos \frac{2\pi k}{N}}{2} = \frac{N}{\pi^2 k^2} \sin^2 \frac{\pi k}{N}
 \end{aligned}$$

$$\begin{aligned}
 \|u_N(0)\|_2^2 &= \langle u_N(0), u_N(0) \rangle \\
 &= \sum_{j=1}^N \frac{1}{N^2} = \frac{1}{N}
 \end{aligned}$$

■

En consecuencia,

Corolario 2.2. Fijado $N \in \mathbb{N}$, la familia $\{u_N(k) : k \in \{0, \dots, N-1\}\}$ resulta una base ortogonal en \mathbb{C}^N .

Observación 2.3. A lo largo de la proposición 2.5 se ha utilizado que, dados dos vectores $x = (x_1, x_2, \dots, x_N), y = (y_1, y_2, \dots, y_N) \in \mathbb{C}^N$, se define

$$\langle x, y \rangle = \sum_{j=1}^N x_j \overline{y_j}$$

Una vez comprobadas las propiedades de los vectores $u_N(k)$, tan solo resta proponer un método de cálculo sencillo de los coeficientes aproximados. Generosamente, también se obtendrá un método inverso para el cálculo de los datos que definen la señal a partir de los coeficientes aproximados. Esto tiene que ver con el hecho de que dichos vectores formen una base de \mathbb{C}^N .

En primer lugar, dado $N \in \mathbb{N}$, notaremos $F_N = (f(x_1^N), f(x_2^N), \dots, f(x_N^N))^T$ al vector de evaluaciones y $\widehat{F}_N = (\widehat{f}_N(0), \widehat{f}_N(1), \dots, \widehat{f}_N(N-1))^T$ al vector de coeficientes de Fourier apro-

2. Discretización

ximados de la señal f .

Advirtamos que, entonces,

$$\widehat{F}_N = A_N F_N$$

donde

$$A_N = \begin{pmatrix} u_N(0) \\ u_N(1) \\ \vdots \\ u_N(N-1) \end{pmatrix} \in \mathcal{M}_N(\mathbb{C})$$

A esta matriz A_N , que nos permite calcular los coeficientes de Fourier aproximados a partir de los datos conocidos de la señal, se la conoce por el nombre de **matriz de análisis**.

Y como venía adelantando, puesto que los vectores de la familia $\{u_N(k) : k \in \{0, \dots, N-1\}\}$ son linealmente independientes, inmediatamente es conocido que A_N es invertible, luego de manera complementaria obtenemos que

$$F_N = A_N^{-1} \widehat{F}_N$$

A la matriz inversa A_N^{-1} se la denomina **matriz de síntesis**, dado que permite recuperar las evaluaciones en la señal a partir de los coeficientes de Fourier aproximados.

Finalmente, exponemos el cálculo explícito de A_N^{-1} a partir de A_N .

Proposición 2.6. *Se verifica que*

$$A_N^{-1} = \overline{A_N}^T D_N^{-1}$$

donde D_N es la matriz diagonal

$$D_N = \begin{pmatrix} \|u_N(0)\|_2^2 & 0 & \dots & 0 \\ 0 & \|u_N(1)\|_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \|u_N(N-1)\|_2^2 \end{pmatrix}$$

Demostración. Por ser todas estas matrices invertibles, bastará con probar que

$$D_N = A_N \overline{A_N}^T$$

En efecto,

$$\begin{aligned}
 A_N \overline{A_N}^T &= \begin{pmatrix} u_N(0) \\ u_N(1) \\ \vdots \\ u_N(N-1) \end{pmatrix} (u_N(0) \mid u_N(1) \mid \dots \mid u_N(N-1)) \\
 &= \begin{pmatrix} \langle u_N(0), u_N(0) \rangle & \langle u_N(0), u_N(1) \rangle & \dots & \langle u_N(0), u_N(N-1) \rangle \\ \langle u_N(1), u_N(0) \rangle & \langle u_N(1), u_N(1) \rangle & \dots & \langle u_N(1), u_N(N-1) \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle u_N(N-1), u_N(0) \rangle & \langle u_N(N-1), u_N(1) \rangle & \dots & \langle u_N(N-1), u_N(N-1) \rangle \end{pmatrix} \\
 &= \begin{pmatrix} \|u_N(0)\|_2^2 & 0 & \dots & 0 \\ 0 & \|u_N(1)\|_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \|u_N(N-1)\|_2^2 \end{pmatrix} = D_N
 \end{aligned}$$

■

2.5. La transformada discreta de Fourier

La anterior sección nos permitió definir y justificar el buen comportamiento de nuestro propio método de discretización de señales. No obstante, una ligera simplificación del modelo propuesto, que radica en una pérdida de precisión pero no de convergencia (salvo producto por un factor constante), como veremos, nos conduce a una de las herramientas más utilizadas hoy en día en el tratamiento de señales: la transformada discreta de Fourier.

Para alcanzar su fórmula, retomamos el modelo de aproximación por funciones constantes a trozos descrito anteriormente, en el cual los coeficientes de Fourier de dichas funciones se calculaban, para cada $k \in \mathbb{Z}$, como

$$\begin{aligned}
 \widehat{f_N}(k) &= \frac{1}{T} \int_0^T f_N(x) e^{-i \frac{2\pi}{T} kx} dx \\
 &= \frac{1}{T} \sum_{j=1}^N \int_{\frac{(j-1)T}{N}}^{\frac{jT}{N}} f(x_j^N) e^{-i \frac{2\pi}{T} kx} dx \\
 &= \frac{1}{T} \sum_{j=1}^N f(x_j^N) \int_{\frac{(j-1)T}{N}}^{\frac{jT}{N}} e^{-i \frac{2\pi}{T} kx} dx
 \end{aligned}$$

La simplificación de la que hacía mención se halla en las integrales de la exponencial compleja en los subintervalos en que se divide $[0, T]$.

2. Discretización

Sin más preámbulo, se trata de

$$\int_{\frac{(j-1)T}{N}}^{\frac{jT}{N}} e^{-i\frac{2\pi}{T}kx} dx \approx \frac{T}{N} e^{-i\frac{2\pi}{T}k\frac{(j-1)T}{N}} = \frac{T}{N} e^{-i2\pi k\frac{(j-1)}{N}}$$

Y la justificación es la siguiente. Como sabemos, las aplicaciones

$$\begin{aligned} \bar{e}_k : \mathbb{R} &\longrightarrow \mathbb{C} \\ x &\longrightarrow e^{-i\frac{2\pi}{T}kx} \end{aligned}$$

para cada $k \in \mathbb{Z}$, resultan continuas en todo punto y T -periódicas. A partir de esta información, la continuidad de \bar{e}_k nos permite deducir su continuidad uniforme en compactos por el Teorema de Heine 2.1; en particular, sobre $[0, T]$.

De este modo, la continuidad uniforme de \bar{e}_k garantiza, dado $\epsilon > 0$ tan pequeño como se quiera, que la distancia entre las imágenes por la aplicación de dos puntos es menor que ϵ con tal de que la distancia entre las abscisas sea menor que un cierto $\delta > 0$ que tan solo depende de ϵ . Asimismo, tomando $N \in \mathbb{N}$ tal que $\frac{T}{N} < \delta$, podemos aproximar la imagen de un elemento de un subintervalo con la de cualquier otra abscisa en el mismo.

Por tanto, tomando N adecuado a la cota ϵ admitida, por el Teorema el valor medio para la integral sabemos que $\exists \xi \in \bar{I}_j$ tal que

$$\int_{\frac{(j-1)T}{N}}^{\frac{jT}{N}} e^{-i\frac{2\pi}{T}kx} dx \approx \frac{T}{N} e^{-i\frac{2\pi}{T}k\xi} \approx \frac{T}{N} e^{-i\frac{2\pi}{T}k\frac{(j-1)T}{N}} = \frac{T}{N} e^{-i2\pi k\frac{(j-1)}{N}}$$

si bien $|\xi - \frac{(j-1)T}{N}| \leq \frac{T}{N} < \delta$.

Una vez justificada la simplificación, que obedece intuitivamente a la idea de las *sumas de Riemann* en tanto que se aproxima una integral por el producto de la longitud del intervalo y la evaluación en un punto del mismo, los coeficientes $\widehat{f}_N(k)$ se aproximarían, a su vez, como sigue. Para cada $k \in \mathbb{Z}$,

$$\begin{aligned} \widehat{f}_N(k) &= \frac{1}{T} \sum_{j=1}^N f(x_j^N) \int_{\frac{(j-1)T}{N}}^{\frac{jT}{N}} e^{-i\frac{2\pi}{T}kx} dx \\ &\approx \frac{1}{T} \sum_{j=1}^N f(x_j^N) \frac{T}{N} e^{-i2\pi k\frac{(j-1)}{N}} \\ &= \frac{1}{N} \sum_{j=1}^N f(x_j^N) e^{-i2\pi k\frac{(j-1)}{N}} \end{aligned}$$

Por tanto, para un N lo suficientemente grande, se verifica que

$$\sum_{j=1}^N f(x_j^N) e^{-i2\pi k\frac{(j-1)}{N}} \approx N \widehat{f}_N(k) \quad \forall k \in \mathbb{Z}$$

Y esto implica que los coeficientes

$$\widehat{Nf_N}(k) := \sum_{j=1}^N f(x_j^N) e^{-i2\pi k \frac{(j-1)}{N}}$$

aportan una información equivalente, salvo homotecia, a la de los coeficientes aproximados $\widehat{f_N}(k)$.

Pues bien, a esta fórmula para el cálculo de los coeficientes aproximados se la conoce como **transformada discreta de Fourier** o **DFT** por sus siglas en inglés.

De manera análoga al estudio realizado en la sección previa, tenemos

$$\widehat{Nf_N}(k) = \langle (f(x_1^N), f(x_2^N), \dots, f(x_N^N)), (1, e^{-i2\pi k \frac{1}{N}}, \dots, e^{-i2\pi k \frac{(N-1)}{N}}) \rangle \quad \forall k \in \mathbb{Z}$$

luego notando

$$v_N(k) = (1, e^{-i2\pi k \frac{1}{N}}, \dots, e^{-i2\pi k \frac{(N-1)}{N}}) \quad \forall k \in \mathbb{Z}$$

puede comprobarse igualmente que

Proposición 2.7. Fijado $N \in \mathbb{N}$, la familia $\{v_N(k) : k \in \{0, \dots, N-1\}\}$ de vectores en \mathbb{C}^N verifica que sus elementos son ortogonales dos a dos y que

$$\|v_N(k)\|_2^2 = N \quad \forall k \in \{0, \dots, N-1\}$$

donde $\|\cdot\|_2$ denota la norma euclídea en \mathbb{C}^N .

Demostración. Dados $k, p, q \in \{1, \dots, N-1\}$ arbitrarios verificando $p \neq q$, se cumple

$$\begin{aligned} \langle v_N(p), v_N(q) \rangle &= \sum_{j=1}^N e^{-i2\pi p \frac{(j-1)}{N}} \overline{e^{-i2\pi q \frac{(j-1)}{N}}} \\ &= \sum_{j=1}^N e^{-i2\pi p \frac{(j-1)}{N}} e^{i2\pi q \frac{(j-1)}{N}} \\ &= \sum_{j=1}^N e^{i2\pi(q-p) \frac{(j-1)}{N}} \\ &= \frac{1 - e^{i2\pi(q-p)}}{1 - e^{i2\pi(q-p) \frac{1}{N}}} = 0 \end{aligned}$$

luego se tiene ortogonalidad dos a dos. Finalmente,

2. Discretización

$$\begin{aligned}
\|v_N(k)\|_2^2 &= \langle v_N(k), v_N(k) \rangle \\
&= \sum_{j=1}^N e^{-i2\pi k \frac{(j-1)}{N}} \overline{e^{-i2\pi k \frac{(j-1)}{N}}} \\
&= \sum_{j=1}^N |e^{-i2\pi k \frac{(j-1)}{N}}|^2 \\
&= \sum_{j=1}^N 1 = N
\end{aligned}$$

■

Corolario 2.3. Fijado $N \in \mathbb{N}$, la familia $\{v_N(k) : k \in \{0, \dots, N-1\}\}$ resulta una base ortogonal en \mathbb{C}^N .

En este caso, la matriz de análisis resulta

$$B_N = \begin{pmatrix} v_N(0) \\ v_N(1) \\ \vdots \\ v_N(N-1) \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & e^{-i2\pi \frac{1}{N}} & \dots & e^{-i2\pi \frac{N-1}{N}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-i2\pi(N-1)\frac{1}{N}} & \dots & e^{-i2\pi(N-1)\frac{N-1}{N}} \end{pmatrix} \in \mathcal{M}_N(\mathbb{C})$$

que, curiosamente, es una matriz de Vandermonde.

Y la matriz de síntesis

$$B_N^{-1} = \overline{B_N}^T C_N^{-1}$$

donde $C_N = NId_N$, siendo Id_N la matriz identidad de orden N . Equivalentemente,

$$B_N^{-1} = \frac{1}{N} \overline{B_N}^T$$

de lo que se obtiene la fórmula

$$f(x_j^N) = \frac{1}{N} \sum_{k=0}^{N-1} \widehat{Nf_N}(k) e^{i2\pi k \frac{(j-1)}{N}} \quad \forall j \in \{1, 2, \dots, N\}$$

que se denomina **transformada inversa de Fourier discreta** o **IDFT**.

Remitimos al análisis exhaustivo de la DFT y sus propiedades en [AD], [Smio7], [Sun01], [BBo8] y [LN14].

Y merece la pena colocar la guinda a este capítulo realizando una observación de gran relevancia a nivel computacional. En resumidas cuentas, nuestro propósito era modelizar el cálculo de unos coeficientes de Fourier aproximados de una señal periódica y alcanzar una sencilla expresión como producto matriz-vector.

En los términos en los que hablábamos, el coste computacional de un algoritmo general de cálculo de unos coeficientes es idéntico al del producto matricial mencionado; es decir, $O(N^2)$ con N el número de entradas del algoritmo. No obstante, la especial forma de la transformada de Fourier discreta, que esgrime una matriz muy particular, permite idear un algoritmo especial para el producto cuyo factor es este tipo de matriz, de modo que el orden del problema se reduce a $O(N \log N)$. A este algoritmo se le conoce comúnmente por el nombre de **transformada rápida de Fourier** o **FFT**.

Esta será la herramienta de la que haremos uso profusamente en la práctica.

3. Implementación del conversor WAV-MIDI

Alcanzamos por fin el objetivo último: la implementación de un conversor de ficheros WAV a MIDI o, equivalentemente, el paso a partitura, en el marco de audios de carácter monódico (una melodía). En aras de una mayor eficiencia, la transformada rápida de Fourier tomará el papel protagonista desde el inicio de nuestras andadas representando la materia prima con la que resolveremos, de manera heurística, el que responde a ser el mayor y más importante problema que habremos de atajar: el hallazgo de la frecuencia fundamental en un intervalo de tiempo.

3.1. Introducción. De la nota musical, el timbre y medidas de distancia

Consideremos nuevamente nuestro espacio de señales, $L_2([0, T])$, y recordemos que, del análisis teórico realizado anteriormente, se desprende, dada una señal T -periódica, que podemos escribirla como suma de un conjunto (finito o infinito) de señales que podríamos calificar de "simples", ponderadas por sus correspondientes coeficientes de Fourier. Estas señales "simples" no eran sino las aplicaciones

$$\begin{aligned} e_k : [0, T] &\longrightarrow \mathbb{C} \\ x &\longrightarrow e^{i\frac{2\pi kx}{T}} \end{aligned}$$

a las que se conoce igualmente como señales u ondas puras, si bien su desarrollo en serie de Fourier coincide con ellas mismas $\forall k \in \mathbb{Z}$ y es fácil comprobar, sabido que la exponencial compleja es una aplicación $2\pi i$ -periódica, que la señal e_k es $\frac{T}{|k|}$ -periódica $\forall k \in \mathbb{Z}$, lo cual arroja la conclusión de que su frecuencia, inversa del período, es $\frac{|k|}{T}$.

En efecto,

$$\begin{aligned} e_k(x) &= e_k(x+t) \Leftrightarrow \\ e^{i\frac{2\pi kx}{T}} &= e^{i\frac{2\pi k(x+t)}{T}} \Leftrightarrow \\ i\frac{2\pi kx}{T} &= i\frac{2\pi k(x+t)}{T} + 2\pi iq, \text{ para cierto } q \in \mathbb{Z} \Leftrightarrow \\ kx &= kx + kt + Tq \Leftrightarrow \\ t &= \frac{-Tq}{k} \end{aligned}$$

siendo el período $\frac{T}{|k|}$.

Dejando a un lado esta reflexión, a nivel físico, una nota musical pura consiste en una

3. Implementación del conversor WAV-MIDI

sinusoidal de frecuencia f , valor que representa unívocamente a su nota pura asociada, razón por la cual, en lo sucesivo, se confundirán deliberadamente los conceptos de nota y frecuencia. Véase, por tanto, que en ambiente más empírico, estamos tomando una señal compleja como compuesto de notas puras (ignorando el término constante que representa e_0) que guardan una estrecha relación: sus frecuencias son múltiplos de $f_0 = \frac{1}{T}$.

Definición 3.1. Llamaremos a $f_0 = \frac{1}{T}$ la **frecuencia fundamental del intervalo** $[0, T]$. Y, al contrario, dada una frecuencia fundamental f_0 , llamaremos **armónicos de f_0** a cada uno de sus múltiplos, nf_0 , $n \in \mathbb{N}$. En música, al conjunto formado por una nota fundamental y sus armónicos se lo conoce como la **serie armónica** de dicha fundamental.

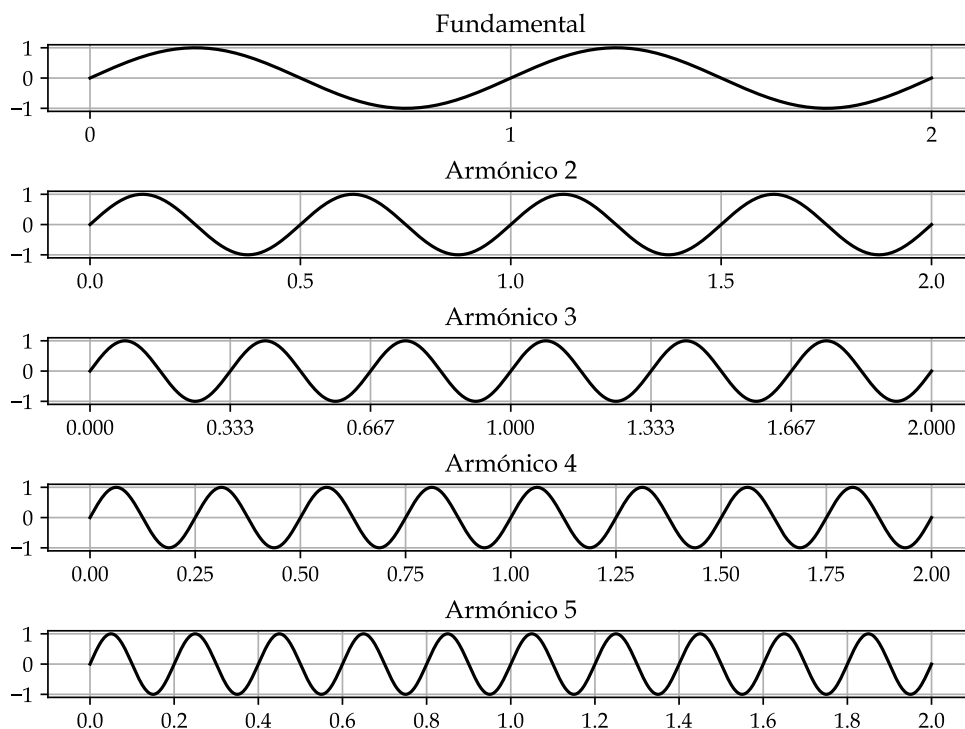


Figura 3.1.: Primeros cinco armónicos para $f_0 = 1$ Hz

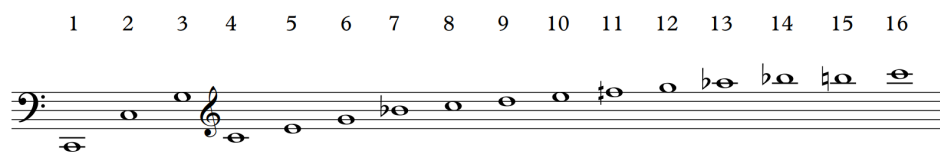


Figura 3.2.: Serie armónica de D_2 ($f_0 = 65.406$ Hz)

Así, en definitiva, dada una señal concreta en $\mathcal{L}_2([0, T])$, la maquinaria levantada nos permite estudiar las contribuciones de los elementos de la serie armónica de la nota pura

$f_0 = \frac{1}{T}$ en su composición. Hay que tener en cuenta una importante observación, y es que dicha frecuencia depende únicamente de la longitud del intervalo $[0, T]$, y el oído humano puede no estar escuchando ninguna de las frecuencias contenidas en su correspondiente serie armónica; de hecho, será lo que suceda más asiduamente, claro está. Hablaremos de este hecho más adelante.

3.1.1. El oído humano: funcionamiento. Intervalos

Por último, para la consecución de nuestro objetivo, habremos de tratar brevemente el funcionamiento del oído humano y la percepción auditiva, dado que, lógicamente, pretendemos imitar sus resultados. Y es que, en su formidable complejidad, el oído percibe la duplicación de frecuencias de manera logarítmica; esto es, **a la audición, la distancia de altura entre una frecuencia cualquiera y el doble es la misma** (obsérvese en el contorno de alturas de la escritura musical de 3.3). Este será uno de los principios físicos básicos al que deberemos atenernos, y que dará lugar a dos medidas distintas de altura entre sonidos.

Definición 3.2. En música, se denomina **intervalo** a la diferencia en altura entre dos sonidos, definida de manera física como el cociente entre sus frecuencias (nótese que existen dos sentidos: uno ascendente y otro descendente). En particular, al valor 2 (altura entre una frecuencia y el doble de la misma) se le denomina **octava**.

En consecuencia, a juzgar por la naturaleza divisiva del intervalo físico y definiendo la relación 2 como la octava, resulta claro que, tomado d_f un intervalo como cociente de frecuencias (de manera física), el número $\log_2 d_f$ contabiliza el número de octavas que caben en tal relación. Si, además, consideramos que en música se divide la octava en doce partes iguales (en clave lineal, tras haber aplicado el logaritmo; a este sistema de afinación se lo conoce como **temperamento igual**, y es el más extendido en la actualidad), denominadas **semitonos**, entonces

$$d_t = 12 \log_2 d_f$$

representa el número de semitonos en el intervalo.

Y, por supuesto, dado un intervalo en semitonos, obtenemos, despejando:

$$d_f = 2^{\frac{d_t}{12}}$$

Sobre afinación y temperamentos, consúltese [Gan97] y [ETe23].

Definición 3.3. Por el principio antedicho, dadas dos cantidades de semitonos, definimos el **intervalo musical o auditivo** simplemente como la diferencia entre ellos.

En consecuencia, dada $d_f \in \mathbb{R}^+$ relación de frecuencias o intervalo físico, el número de semitonos que corresponde al sonido que surge al incrementar en dicho intervalo un sonido base con t_0 semitonos viene dado por

$$t(d_f) = t_0 + 12 \log_2 d_f$$

Y, por otro lado, dada $d_t \in \mathbb{R}$ diferencia de semitonos o intervalo musical, la frecuencia del sonido que surge al incrementar en dicho intervalo un sonido base con frecuencia f_0 se

3. Implementación del conversor WAV-MIDI

calcula como

$$f(d_t) = f_0 2^{\frac{d_t}{12}}$$

Utilizaremos estas aplicaciones muy pronto.

Como enunciaba, el temperamento igual implica la división en doce partes iguales (en base a la concepción logarítmica, auditiva) de la octava, denominadas semitonos. De acuerdo a esto, el intervalo del semitono viene representado por la relación de frecuencias

$$d_f = 2^{\frac{1}{12}} = \sqrt[12]{2}$$

y el sistema se construye aplicando este intervalo ascendente y descendientemente a partir de una altura de partida, establecida por convenio para ser el La de la octava cuarta (La_4 según la numeración propuesta por el índice acústico denominado **sistema de los físicos**), y que se corresponde con la frecuencia de 440 Hz.

Dejando a un lado esta discusión, hemos de pasar a mencionar el importante hecho de que el oído percibe la serie armónica que parte de una fundamental dada como un único sonido, con tal de que sus amplitudes asociadas sean lo suficientemente grandes como para percibirse propiamente como tal. Atendiendo a este fenómeno, la variabilidad de intensidades de los armónicos (o de sus coeficientes de Fourier asociados) es, precisamente, lo que da pie al **timbre**, cualidad que permite diferenciar sonidos iguales en altura e intensidad. Así, por ejemplo, un violín y un piano, aún tocando la misma nota con la misma intensidad, suenan diferente porque su timbre es distinto; es decir, la forma en que ponderan los armónicos dada su morfología y otros múltiples factores físicos, difiere.

De hecho, es tal la asociación que realiza el oído de sonidos pertenecientes a la misma serie armónica que, podría suceder (y de hecho, sucede) la sorprendente patología consistente en la audición de una fundamental que no está sonando, mientras sí lo hagan el resto de sus armónicos con la suficiente intensidad.

Se puede imaginar entonces que el mayor obstáculo en la detección de alturas consiste en la imitación efectiva y digital del complejo mecanismo de la audición, materia que discutiremos profusamente en el futuro mediante la reflexión sobre múltiples y variadas alternativas.

Explicaciones más profusas acerca de armonía y teoría de la música pueden hallarse en [Pis12], [dP14] y [Zam97], y al respecto de cuestiones acústicas referentes a la música y la audición humana, en [CB16], [DO93] y [CM91].

3.2. Esbozo del algoritmo

En origen, el objeto de entrada con que se trabaja es un fichero de audio en formato WAV, que puede comprenderse fundamentalmente como un conjunto equiespaciado de muestras que representan una discretización de la señal en el dominio del tiempo; esto es, a cada instante considerado, le corresponde un valor de intensidad o amplitud. A la representación

de esta suerte en el dominio del tiempo, *tiempo* \rightarrow *intensidad*, se la conoce como su **forma de osciloscopio**.

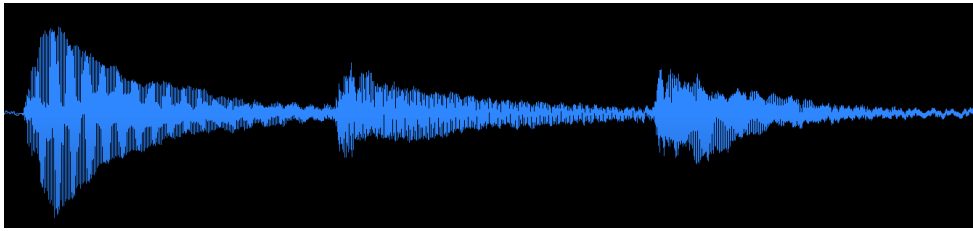


Figura 3.3.: Ejemplo de vista de osciloscopio de una señal, extraída vía *iZotope RX8*

3.2.1. Ventanas y solapamiento

Como hemos visto, la industria desarrollada resulta aplicable en intervalos temporales de una cierta longitud; en consecuencia, es lógico pensar que habremos de trabajar el audio en fragmentos, que denominaremos **ventanas** ([NI]), y cuya longitud bien podría variar de forma dinámica, bajo la pretensión de dilucidar la nota musical que el oído humano percibiría como consecuencia de la información contenida en dicho espacio de tiempo. Ante esta idea, lo razonable sería seleccionar una longitud de intervalo lo más pequeña posible, a la intención de evitar el indeseable escenario en que dos o más notas se confundan como una sola, fruto de integrarlas todas ellas en la misma ventana. Desgraciadamente, esto no es tan sencillo como parece.

La problemática surge naturalmente de una de las conclusiones extraídas en la sección anterior: el cómo la elección de T determina la frecuencia fundamental y, en consecuencia, sus armónicos. Observemos que, a menor longitud de intervalo, es decir, a menor T , la frecuencia fundamental de la ventana es mayor, y sus armónicos, consecuentemente, se encuentran más espaciados, lo que dificulta el encaje o la búsqueda de series armónicas de fundamentales diferentes, disminuyendo la precisión de un potencial algoritmo de cálculo de la frecuencia fundamental real, que el oído verdaderamente percibe. En particular, al aumentar los valores de la serie armónica empleada para la descomposición, perdemos efectividad especialmente sobre las frecuencias graves.

En resumidas cuentas, si pretendemos una considerable precisión en el análisis de frecuencias, deviene imperativo aumentar la longitud de la ventana. Sin embargo, debemos actuar con precaución, pues un mayor tamaño de ventana podría ocasionar la confusión de sonidos que sucedan demasiado rápido. Así es que nos encontramos ante un particular “tira y afloja” en que la configuración óptima debe amoldarse, idealmente, a la señal tratada en cada caso.

Por otro lado, y acerca de la fragmentación del fichero de audio, no tomaremos una partición (un conjunto exhaustivo de intervalos disjuntos), sino que se admitirá y, de hecho, se gestionará como recurso adicional, el solapamiento entre ventanas. El interés depositado en la intersección de ventanas, para un procedimiento de análisis (y no de síntesis) como es este, alude principal y sencillamente a disponer de un mayor conjunto de ventanas que permitan vislumbrar la nota correspondiente en un determinado momento. Es decir, permite agregar

3. Implementación del conversor WAV-MIDI

varios resultados sobre un mismo intervalo de tiempo para decidir con mayor información. Por supuesto, a expensas de un mayor coste computacional que, sin embargo, no aumenta el orden de complejidad en tiempo.

A este respecto, la conjugación de ventanas con solapamiento exige, en todo proceso de síntesis, la ponderación de las evaluaciones en la ventana por una función continua, cóncava y cercana a cero en ambos extremos, conocida como **función de hanning**, con el objetivo de suavizar las transiciones entre una ventana y la siguiente, si bien las discontinuidades generadas en la señal sintetizada en caso de no aplicar un hanning añadirán ruido indeseable. En un procedimiento de análisis, como es el que vamos a emprender, el especial interés del hanning radica en la dilatación de picos tras la aplicación de la FFT, lo que palia los efectos de la dispersión de los armónicos de la frecuencia fundamental de la ventana, inconveniente que discutiremos más adelante.

En particular, la función de hanning escogida para el algoritmo es la siguiente:

$$h(x) = \frac{1}{2} (1 - \cos(\frac{2\pi}{T}x)) \quad \forall x \in [0, T]$$

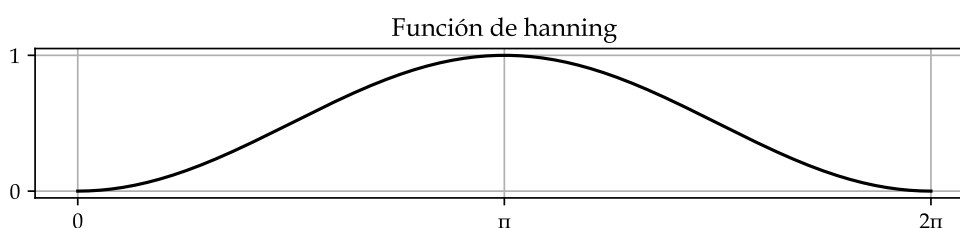


Figura 3.4.: Función de hanning escogida para $T = 2\pi$.

3.2.2. Esquema general

Con todo, habiendo comprendido las particularidades de la división en ventanas y el solapamiento, se procede a listar el orden de actuación en pos de lograr el objetivo que se tiene entre manos:

1. Lectura del fichero de audio como un vector de muestras (evaluaciones temporalmente equiespaciadas y ordenadas de la señal).
2. Establecimiento de los parámetros esenciales, como son el tamaño de la ventana y el solapamiento, de manera acorde a las características del audio a convertir (se debe tener en cuenta la velocidad de sonidos cortos con el fin de elegir el tamaño de manera óptima).
3. Iteración sobre el conjunto de ventanas solapadas y cálculo, para cada una, de la nota musical audible junto con un peso basado en su intensidad, como medida de fiabilidad de la estimación. Ambos valores se dispondrán en vectores independientes.

4. Corrección de la estimación de cada ventana atendiendo a las estimaciones de ventanas en la vecindad y a sus pesos asociados.
5. Iteración sobre el vector corregido de notas por ventana y agrupación de conjuntos de notas iguales consecutivas junto con el cálculo, para cada una, de la duración real que le corresponde, tanto en segundos como en negras (lo cual depende de la velocidad de la negra, a establecer como parámetro).
6. Construcción final del archivo *MIDI* y de la partitura.

Donde, sin lugar a dudas, el paso tercero resulta el de mayor complejidad como consecuencia de la ardua tarea que supone el cálculo de la frecuencia fundamental audible en cada ventana; equivalentemente, de la nota musical percibida por el oído.

3.3. El algoritmo de conversión

Comencemos, pues, a desglosar las diferentes etapas que componen el algoritmo de conversión propuesto.

3.3.1. Lectura del fichero WAV y cálculo de parámetros

Con el fin de completar las primeras fases del proceso, precisamos la ruta del archivo de audio que emplearemos para la conversión, y los tamaños de ventana y de solapamiento entre ellas, dados en segundos. Los llamaremos, respectivamente, *WINDOW_SIZE_SECS* y *OVERLAPPING_SECS*.

La lectura se realiza de manera sencilla haciendo uso de la función *wavfile.read* importada directamente del módulo *scipy.io*, a la que meramente habrá que indicar la ruta especificada por el usuario. Como retorno, se obtienen la frecuencia de muestreo, que almacenaremos en la variable global *SAMPLE_RATE* y un vector con las evaluaciones, por canal de audio, de cada muestra de la señal. Nos contentamos, sin embargo, con obtener la información asociada únicamente al primer canal.

Recordemos que la frecuencia de muestreo consiste en el número de muestras tomadas por segundo y que los valores usualmente utilizados son 44100 Hz, 48000 Hz y otros superiores. El porqué radica precisamente en la consecuencia de un resultado de inmensa importancia en el procesamiento de señales: el **Teorema de Nyquist-Shannon**, que ofrece la solución al problema que recibe el nombre de **aliasing** y que consiste en la distorsión de la señal de partida (continua) al discretizarla por medio de una frecuencia de muestreo demasiado baja, lo que, intuitivamente, causa la pérdida de información sobre movimientos más breves de la señal; es decir, se pierde información al respecto de las frecuencias agudas. Los ejemplos 3.5 y 3.6 ilustran el fenómeno:

3. Implementación del conversor WAV-MIDI

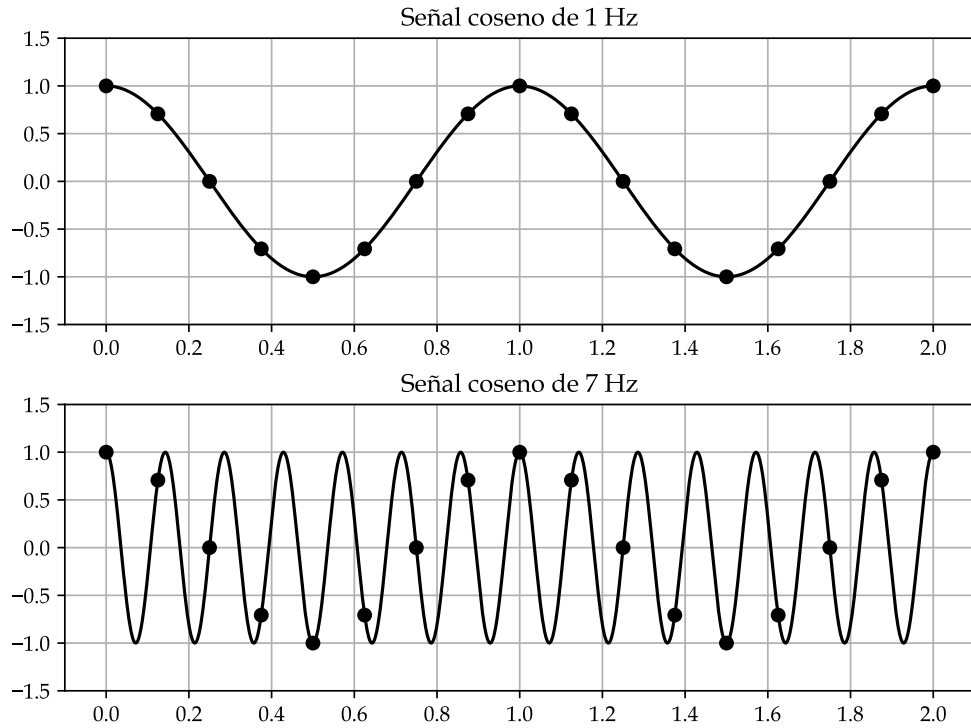


Figura 3.5.: Confusión de las señales $\cos(2\pi x)$ y $\cos(14\pi x)$ a causa de muestrear en puntos de corte (concretamente, en $\frac{1}{8}\mathbb{N}_0$)

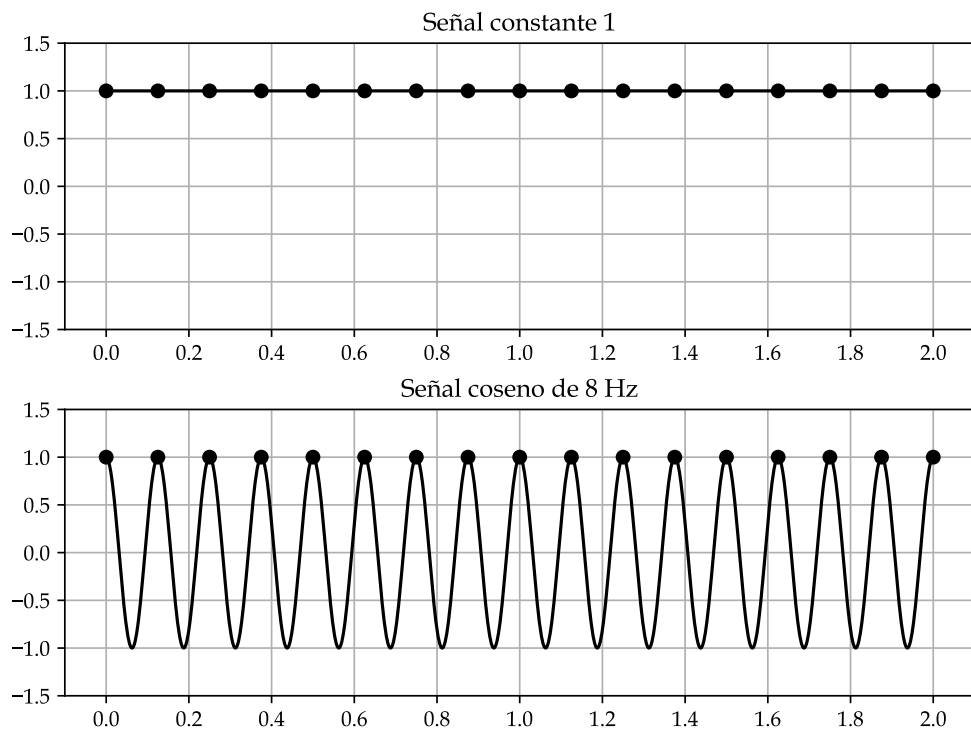


Figura 3.6.: Confusión de las señales 1 y $\cos(16\pi x)$ a causa de muestrear en puntos de corte (concretamente, en $\frac{1}{8}\mathbb{N}_0$)

Así, en breve:

Teorema 3.1. (De Nyquist-Shannon). Con el objetivo de prevenir la pérdida de información acerca de las frecuencias que componen una determinada señal continua, su discretización habrá de realizarse con una frecuencia de muestreo de, al menos, el doble del mayor valor de frecuencia que compone dicha señal.

Por tanto, y teniendo en cuenta que el oído humano no es capaz de percibir frecuencias superiores a los 20000 Hz, los tamaños antedichos claramente satisfacen nuestras necesidades auditivas, independientemente de cuál sea la mayor frecuencia de la señal continua considerada. Es más, la discretización en sí misma puede entenderse como un mecanismo de compresión de audio propiamente, al tiempo que, por añadidura, no notamos la diferencia.

Habiendo aclarado este concepto, podemos calcular de manera sencilla tres parámetros a partir de la información de que disponemos en estos instantes: el tamaño de la ventana en número de muestras, el tamaño de solapamiento en número de muestras y el tamaño total del audio en segundos. Respectivamente definidos como `WINDOW_SIZE_SAMPLES`, `OVERLAPPING_SAMPLES` y `AUDIO_SIZE_SECS`, se verifica:

$$WINDOW_SIZE_SAMPLES = \frac{SAMPLE_RATE}{WINDOW_SIZE_SECS}$$

$$OVERLAPPING_SAMPLES = \frac{SAMPLE_RATE}{OVERLAPPING_SECS}$$

$$AUDIO_SIZE_SECS = \frac{AUDIO_SIZE_SAMPLES}{SAMPLE_RATE}$$

donde `AUDIO_SIZE_SAMPLES` denota la longitud del vector que representa la señal, el número total de muestras del fichero de audio.

3.3.2. Estimación de la frecuencia fundamental

Finalmente topamos con la mayor dificultad, la tarea más intrincada: el cálculo de la frecuencia fundamental real o audible en cada ventana. No obstante, definimos previamente un conjunto de métodos que nos acompañarán a lo largo de las variadas alternativas propuestas para su resolución.

Pero, en primer lugar, traigamos de vuelta a debate el temperamento igual. De acuerdo a lo expuesto anteriormente, la **notación MIDI** propone numerar con naturales positivos las alturas consideradas por el temperamento igual, siendo el número *MIDI* 69 el correspondiente a la nota La_4 .

En consecuencia, la función de conversión de frecuencia a número *MIDI* consiste en

$$n(f) = 69 + 12 \log_2 \frac{f}{440}$$

3. Implementación del conversor WAV-MIDI

Y su contraparte, la conversión de número *MIDI* a frecuencia toma la forma

$$f(n) = 440 \cdot 2^{\frac{n-69}{12}}$$

A continuación, se presenta un diagrama de equivalencias entre números *MIDI*, notación musical y frecuencial, a modo de referencia para cálculos en el contexto del temperamento igual.

MIDI number	Note name	Keyboard	Frequency
21	22	A0	27.500
23	B0		30.868 29.135
24	C1		32.703
25	D1		36.708 34.648
26	E1		41.203 38.891
28	F1		43.654
29	G1		48.999 46.249
31	A1		55.000 51.913
32	B1		61.735 58.270
33	C2		65.406
34	D2		73.416 69.296
36	E2		82.407 77.782
37	F2		87.307
38	G2		97.999 92.499
40	A2		110.00 103.83
41	B2		123.47 116.54
42	C3		130.81
43	D3		146.83 138.59
44	E3		164.81 155.56
45	F3		174.61
46	G3		196.00 185.00
47	A3		220.00 207.65
48	B3		246.94 233.08
49	C4		261.63
50	D4		293.67 277.18
51	E4		329.63 311.13
52	F4		349.23
53	G4		392.00 369.99
54	A4		440.00 415.30
55	B4		493.88 466.16
56	C5		523.25
57	D5		587.33 554.37
58	E5		659.26 622.25
59	F5		698.46
60	G5		783.99 739.99
61	A5		880.00 830.61
62	B5		987.77 932.33
63	C6		1046.5
64	D6		1174.7 1108.7
65	E6		1318.5 1244.5
66	F6		1396.9
67	G6		1568.0 1480.0
68	A6		1760.0 1661.2
69	B6		1975.5 1864.7
70	C7		2093.0
71	D7		2349.3 2217.5
72	E7		2637.0 2489.0
73	F7		2793.0
74	G7		3136.0 2960.0
75	A7		3520.0 3322.4
76	B7		3951.1 3729.3
77	C8		4186.0

Figura 3.7.: Gráfico de equivalencias

Además, convenientemente, numeradas del 0 al 12 las notas de la **escala cromática** (desde

Do hasta Si de manera ordenada, por semitonos), es fácil comprobar que el índice de cada nota coincide con la congruencia módulo 12 del número MIDI asociado,

$$NOTE_NAME_INDEX(n) = n \bmod 12 = n \% 12$$

y el número de octava, según el sistema de los físicos, puede calcularse como

$$NOTE_OCTAVE(n) = \lfloor \frac{n}{12} \rfloor - 1$$

Estos tres procedimientos de conversión nos serán vitales en lo que sigue. Pero, además, consideramos un método auxiliar de extracción de ventanas, entendidas como vectores contenidos en la señal, conocidos *WINDOW_SIZE_SAMPLES*, *OVERLAPPING_SAMPLES* y disponiendo, en forma de parámetro, del número de ventana a extraer. Lógicamente, en función del tamaño de ventana elegido, la última de ellas podrá o no extraerse completamente de la señal; con frecuencia, ocurrirá lo segundo, lo cual se suplirá rellenando el resto de valores del vector con cero (lo que se conoce como **zero padding**), de manera que se puedan seguir estudiando las frecuencias graves como hasta el momento.

A partir de estos elementos, comenzamos a iterar sobre el conjunto de ventanas en que fragmentamos el fichero de audio y, para cada una, aplicamos hanning y efectuamos la transformada rápida de Fourier por medio de la implementación aportada por la librería *numpy*; más en particular, hablamos del método *numpy.fft.fft*. En aras de mayor eficiencia, la FFT se aplica sobre conjuntos de muestras de cardinalidad potencia de dos, luego cabe esperar que el algoritmo utilizado realice, de manera independiente, zero padding sobre el vector que se pasa como parámetro.

Por otro lado, tal y como se detalló, la FFT provee la descomposición espectral de la ventana ponderada mediante hanning, utilizando como señales puras los armónicos de la frecuencia fundamental de aquella. En definitiva, para cada armónico de dicha frecuencia, obtenemos un valor $z_k \in \mathbb{C}$ que se corresponde con la aproximación de su k -ésimo coeficiente de Fourier asociado. Dispuesto en forma polar,

$$z_k = |z_k| e^{i \cdot \arg(z_k)}$$

luego, al presentarse como factor de la señal pura e_k , el número $e^{i \cdot \arg(z_k)}$ tan solo genera, a efectos prácticos, un desfase en la aplicación e_k . Teniendo en consideración que, para el oído humano, el desfase de una señal no afecta al reconocimiento de su altura, según apunta la Ley acústica de Ohm [Ohm22], podemos tomar dicho factor complejo, el giro $e^{i \cdot \arg(z_k)}$, como despreciable, de manera que utilizaremos el módulo de los valores complejos computados a través de la FFT, con la gratificante ofrenda adicional de trabajar en \mathbb{R} en lugar de en \mathbb{C} , lo cual hará los cálculos y, especialmente, la representación gráfica, extremadamente sencillos. En definitiva, se interpretará el módulo de estos complejos z_k como los valores de amplitud o intensidad asociados a la presencia de la señal e_k , armónico k -ésimo de la frecuencia fundamental de la ventana, en la identidad del sonido audible.

Y, más en particular, el conocimiento de las frecuencias de estos armónicos junto con sus correspondientes medidas de amplitud permitirá trabajar con gráficas frecuencia-intensidad,

3. Implementación del conversor WAV-MIDI

que interpolen los valores discretos de que disponemos en forma de función continua. A este tipo de representación se la conoce como **análisis espectral**, y será la herramienta base para el reconocimiento de tonos, fin para el cual las diferentes propuestas exploradas se detallan a continuación.

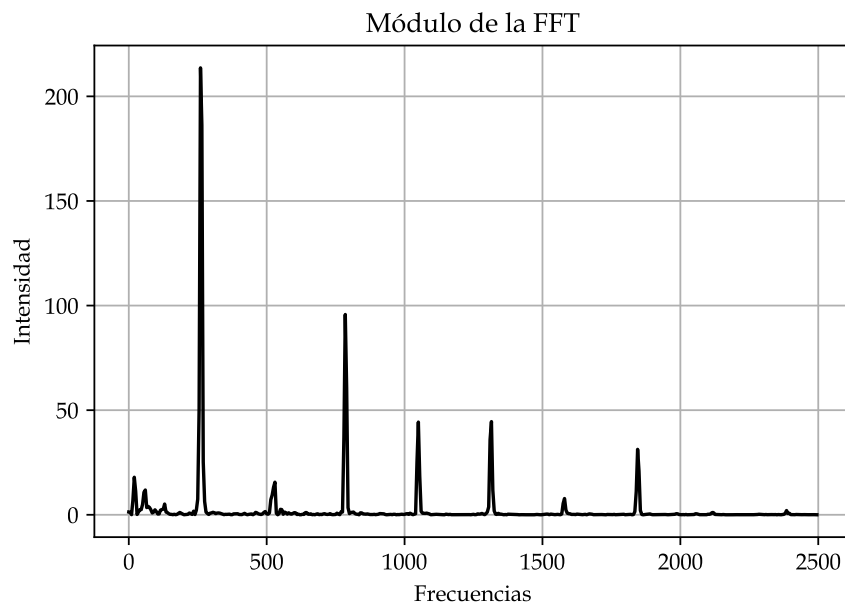


Figura 3.8.: Ejemplo de análisis espectral. ¹

Propuesta primera: frecuencia de máxima amplitud

Inocentemente, se podría pensar que la frecuencia fundamental real se localice en el pico más elevado de la gráfica de análisis espectral. Sin embargo, como ya se advertía, podría suceder hasta que la componente pura correspondiente tuviese amplitud nula, lo cual inmediatamente obliga a descartar este método. De hecho, para ilustrar mejor la facilidad con que este fenómeno se manifiesta, adjuntamos a continuación el análisis espectral de una ventana perteneciente a una grabación de piano, en la cual la preeminencia del segundo armónico resulta absolutamente desmedida.

¹La gráfica muestra el análisis espectral de una ventana en que suena, claramente apreciable, un Do_4 (261.63 Hz), como puede intuirse en la distribución de picos: múltiplos de dicha frecuencia fundamental.

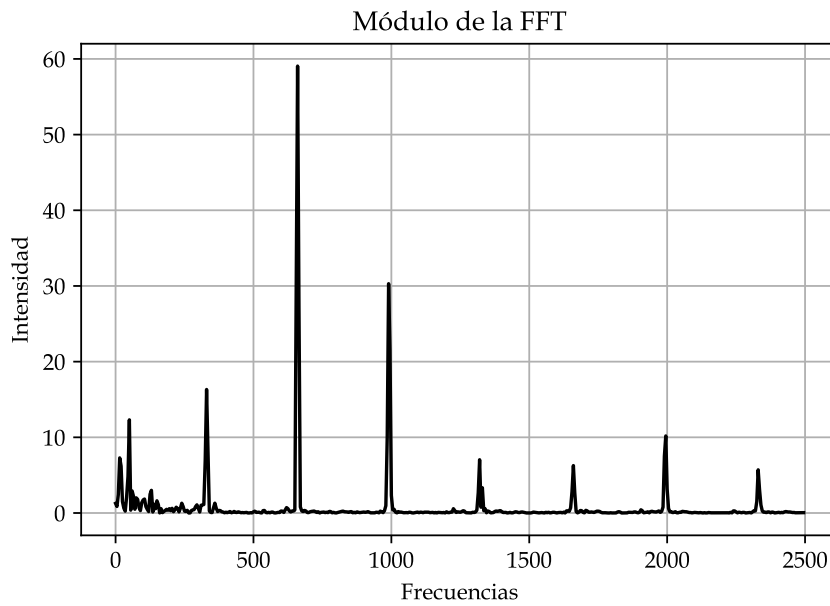


Figura 3.9.: Escenario de segundo armónico preeminente.²

A continuación, se muestra el resultado (nombre de la nota estimada en cada ventana) sobre una grabación de 14 segundos en piano (*DoM-piano.wav*, descargada de internet, de calidad media-baja) de una escala sencilla de *DoM*, en sentido ascendente y descendente, en la octava cuarta según el sistema de los físicos. Se juega con una frecuencia de muestreo de 44100 Hz, con un tamaño de ventana de 0.2 segundos y un solapamiento de 0.15 segundos:

[C4, C4, C4, C4, C4, C4, C5, C4, C4, C4, C5, C5, C5, C4, C5, G5, D4, D4, D5, D4, D4, D4, D4, D4, D4, D4, D4, D4, D5, D4, D4, D5, D4, F4, E6, E5, E5, E5, E5, E5, E5, E5, E5, E5, E5, E5, E5, E5, B1, D#0, F4, F4, F4, F4, F4, F5, F5, F5, F4, F4, F4, F4, F4, F4, F4, F4, G#4, D6, G4, G5, G4, G5, G4, G5, G4, G5, G5, G5, G5, G5, G5, G5, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A5, D#0, D#0, D#0, A1, D#0, D#0, B4, B4, B4, B4, B5, B4, B4, B4, B4, F#6, B4, B4, B5, B4, D#0, D#0, D#0, D#0, D#0, B4, C5, C5, C5, C5, C6, C5, C6, C5, C5, C5, C5, C5, C5, C5, C5, C5, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B5, B4, A1, B4, B4, A1, D#0, D#0, A4, A4, A4, A4, A5, A5, A5, E6, A4, A4, A4, A4, A4, A4, A4, B1, B1, B1, G#4, G4, G5, G4, G5, G5, G5, G5, G5, G5, G5, G4, G5, B1, G5, G4, G5, F4, F4, F4, F4, F4, F4, F4, F5, F5, F5, F5, F4, F4, F4, F4, F4, F4, E5, E4, E6, E5, E6, E5, E6, E6, E5, E6, B5, A1, D#0, D#0, D#0, D#0, A1, E6, D4, D4, D4, F#6, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D5, D4, D4, D5, D4, C4, C4, C4, C4, C5, C4, C4, C4, C5, C4, C4, C5, A1, G5, C4, G5, G5, D#0, C5, D#0, G5, C5, D#0, D#0]

Mientras que, sin embargo, la respuesta correcta era:

[C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, G4, G4, G4,

²En este caso, la gráfica corresponde a un Mi_4 (329.63 Hz).

3. Implementación del conversor WAV-MIDI

[illegible]

Comparemos visualmente estas secuencias en la forma de gráfica representando, para cada ventana, el número MIDI (en escala logarítmica, auditiva) de la nota real (en rojo) contrapuesto al de la nota computada por el algoritmo (en negro):

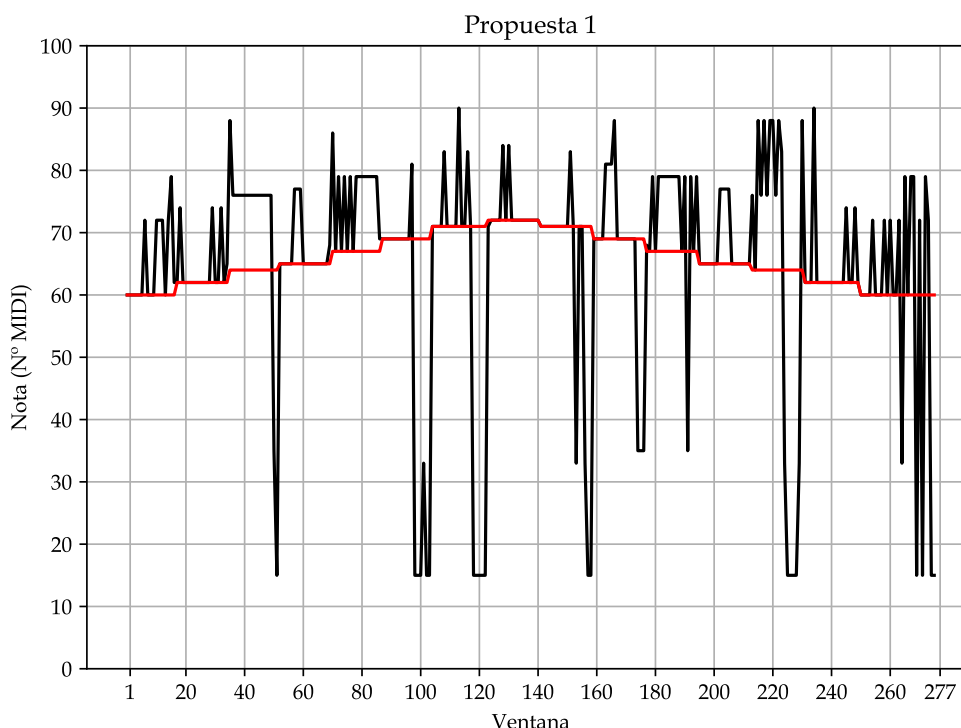


Figura 3.10.: Comparativa entre el desempeño de la propuesta primera y la clasificación real.

Propuesta segunda: frecuencia del primer máximo

La desencaminada alternativa primera conduce a pensar que, quizás, la respuesta se halle en la frecuencia que corresponde al primer máximo prominente de la señal si bien, aun ignorando el principio de la no necesidad de presencia de la fundamental, a nivel práctico, en la naturaleza, tal situación no es esencialmente común. Con todo, esta alternativa es tan desdeñable como la primera por el mismo argumento.

Pero, además, será en añadido la dificultad en la definición de "pico prominente" lo que

interesó investigar en este momento. Y la conclusión, tomada sobre numerosas ventanas, fue que todo intento de establecer un umbral constante sobre cada una de ellas basado en estimaciones sencillas como la media de las evaluaciones (demasiado bajo) o la media del máximo y el mínimo de las mismas (demasiado alto) resultaba del todo infructuoso, tal y como se puede observar a continuación.

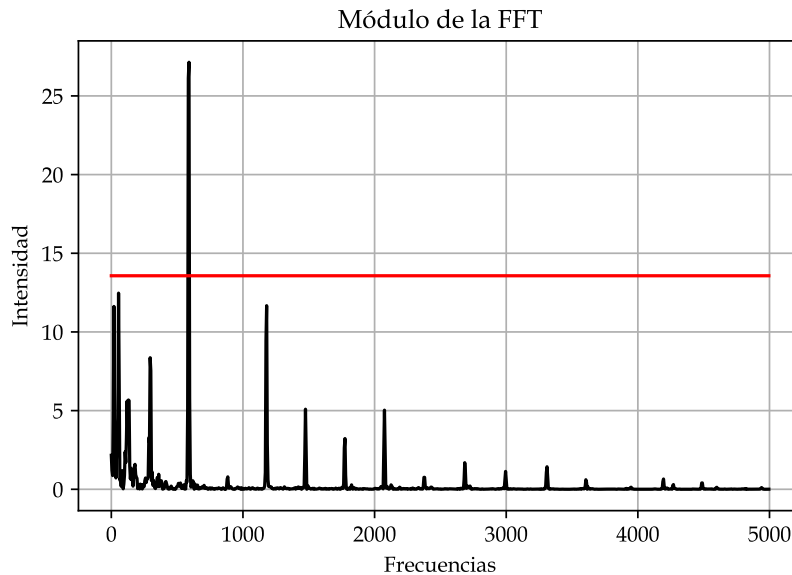


Figura 3.11.: Error al clasificar por el umbral; la fundamental queda debajo. ³

Finalmente, el resultado sobre *DoM-piano.wav* en las condiciones previamente descritas, y haciendo uso de la media entre máximo y mínimo como umbral de picos relevantes, es:

[C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, D#0, D#0, D#0, D4, C#4, D4, D4, D4, D4, D4, D4, D4, D4, D5, D4, D4, D#0, D4, D4, D#4, F4, E4, E4, E4, E4, E4, E5, E5, E5, A1, E5, A1, E5, D#0, D#0, D#0, F4, F4, F4, F4, F4, F4, F5, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F#4, G4, G4, G5, G4, G4, G4, G5, G4, G4, G5, G4, A1, G5, D#0, G4, G4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, D#0, D#0, D#0, D#0, D#0, D#0, D#0, D#0, B4, B4, B4, B4, B4, B4, B4, B4, B4, A1, B4, B4, D#0, D#0, D#0, D#0, D#0, D#0, D#0, D#0, C5, C5, C5, C5, C5, C5, C6, C5, C5, C5, C5, C5, C5, C5, C5, C5, A1, D#0, B4, B4, B4, B4, B4, B4, B4, A1, B4, B4, B4, B4, A1, B1, B4, D#0, D#0, D#0, G#4, A4, A4, A4, A4, A5, A5, D#0, B1, A4, A4, A4, A4, A4, A1, B1, F#1, D#0, F4, G4, G5, G4, G4, G4, G5, G4, G4, G5, G1, G4, G4, G5, B1, B1, B1, C2, F4, F4, F4, F4, F4, F4, F4, F4, F5, F5, F4, F4, F4, F4, F4, F4, F4, F4, D#4, F4, E4, E4, E4, E4, E4, E5, E5, D#0, B5, D#0, D#0, D#0, D#0, D#0, D#0, D#0, D#0, D4, D4, D4, F#6, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D#0, D4, D4, D#0, D4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C5, D#0, C5, D#0, D#0, D#0, D#0, D#0, D#0, D#0, D#0, D#0, D#0]

siendo la gráfica comparativa asociada

³La nota audible es Re_4 (293.67 Hz).

3. Implementación del conversor WAV-MIDI

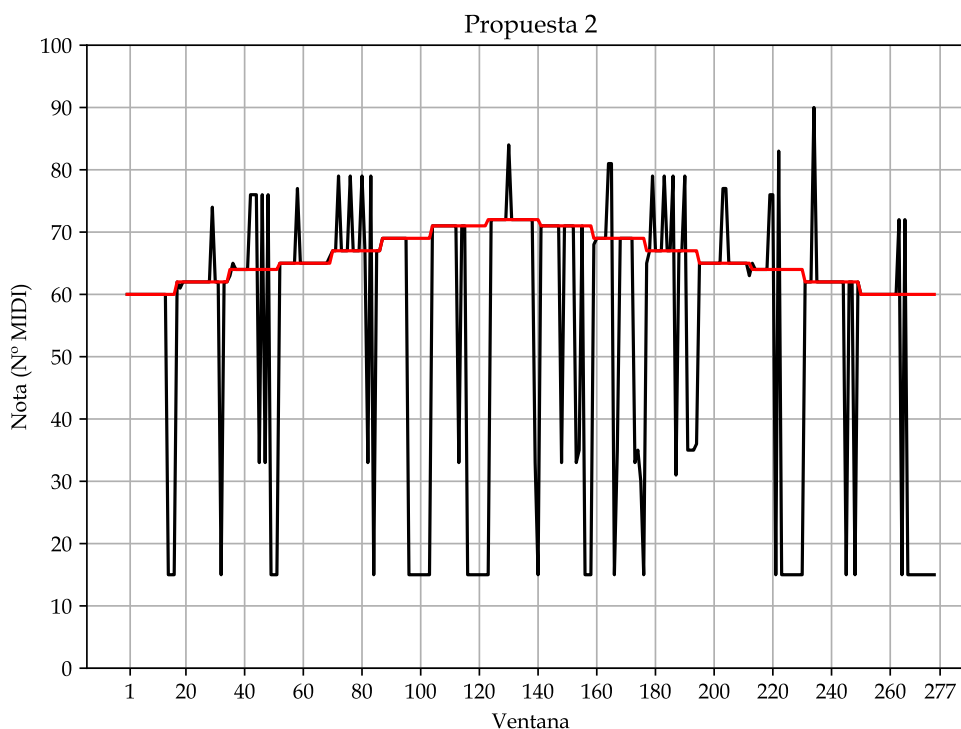


Figura 3.12.: Comparativa entre el desempeño de la propuesta segunda y la clasificación real.

Propuesta tercera: frecuencia con armónicos de mayor amplitud media

A partir de los anteriores aperitivos, el primer enfoque serio trató de considerar no solo frecuencias individuales, sino grupos de ellas; por supuesto, hablamos del conjunto de armónicos. Asimismo, se barajó la razonable hipótesis de relevancia de las intensidades asociadas a cada uno de estos armónicos.

Como consecuencia de la conjunción de ambas ideas nace la alternativa tercera, que propone recorrer el conjunto de los armónicos de la frecuencia fundamental de la ventana, para los cuales se dispone de una evaluación vía la FFT, y calcular, por cada uno de ellos, la media de las amplitudes de sus propios armónicos. Lógicamente tomaremos, como estimación de la frecuencia fundamental audible, aquella de mayor amplitud media entre las calculadas. Posteriormente, será aproximada al temperamento igual por medio de una argucia que nos será de gran utilidad en lo sucesivo: la transformación continua a número *MIDI* y su redondeo al natural más cercano, con lo que ya estaríamos en disposición de transformar a nota musical propiamente.

Desgraciadamente, a pesar de la industria de este modelo, existen varios inconvenientes para nada despreciables:

En primer lugar, es imperativo señalar el tremendo coste computacional que requiere su ejecución, de orden cuadrático $O(n^2)$ en el número de muestras, agravado todavía más si cabe por el solapamiento entre ventanas, que conlleva esperas nada agradables incluso para

tamaños de audio pequeños. En definitiva, se trata de un algoritmo de fuerza bruta que peca por su planteamiento poco eficiente.

Pero, por otro lado, la eficacia también queda rezagada a causa de realizar la media de las intensidades de los armónicos. Y la causa es, como no puede ser de otra manera, la suposición de igualdad en la ponderación de las frecuencias. Concretamente, en clave empírica se observó que dividir la suma de las alturas dadas por la FFT entre el número de sumandos considerados (la mera media aritmética) suponía una penalización demasiado fuerte para las frecuencias graves, al respecto de las cuales, por supuesto, el número de armónicos considerados siempre será mayor que para una frecuencia aguda (simplemente porque caben más armónicos de frecuencias pequeñas en un mismo intervalo).

En efecto, como se verá a continuación sobre el fichero de prueba, los errores cometidos frecuentemente estiman por frecuencias superiores a la real (usualmente por el segundo armónico, la octava):

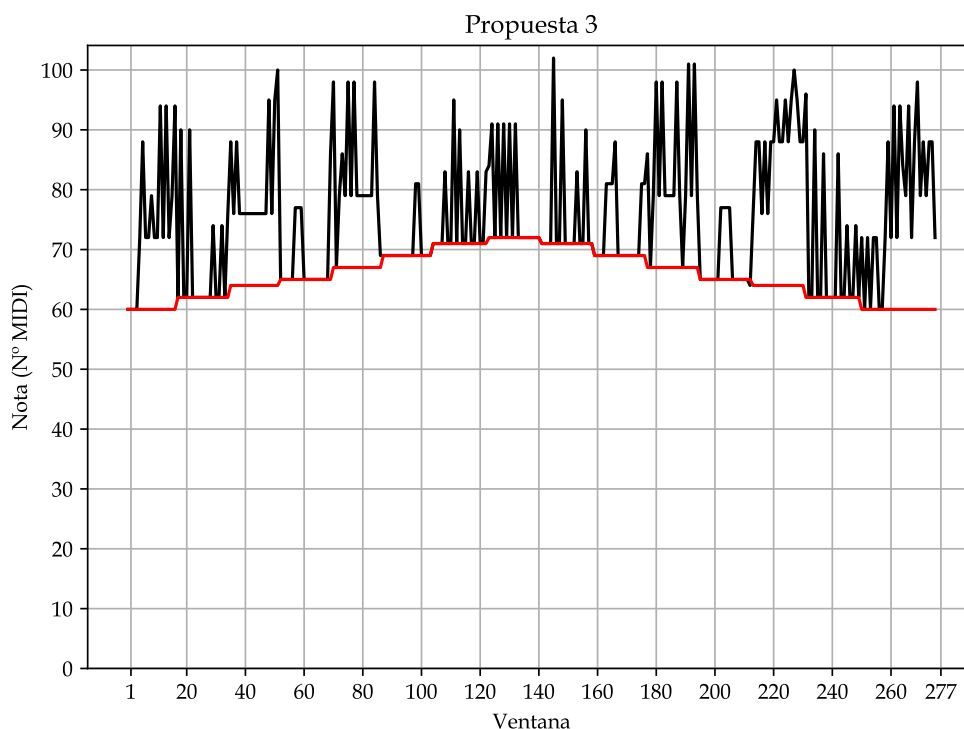


Figura 3.13.: Comparativa entre el desempeño de la propuesta tercera y la clasificación real.

para la sucesión de predicciones

[C4, C4, C4, C4, C5, E6, C5, C5, G5, C5, C5, A#6, C5, A#6, C5, G5, A#6, D4, F#6, D4, D4, F#6, D4, D4, D4, D4, D4, D4, D4, D5, D4, D4, D5, D4, E5, E6, E5, E6, E5, E5, E5, E5, E5, E5, E5, E5, E5, B6, E5, B6, E7, F4, F4, F4, F4, F4, F5, F5, F5, F4, F4, F4, F4, F4, F4, F4, F4, D6, D7, G4, G5, D6, G5, D7, G5, D7, G5, G5, G5, G5, G5, G5, G5, D7, G5, A4, A4, A4, A4, A4, A4, A4, A4,

3. Implementación del conversor WAV-MIDI

A4, A4, A4, A4, A5, A5, A4, A4, A4, A4, B4, B4, B4, B4, B5, B4, B4, B6, B4, F#6, B4, B4, B5, B4, B4, B5, B4, B4, B5, C6, G6, C5, G6, C5, G6, C5, G6, C5, G6, C5, C5, C5, C5, C5, C5, B4, B4, B4, B4, F#7, B4, B4, B6, B4, B4, B4, B4, B5, B4, B4, F#6, B4, B4, A4, A4, A4, A4, A5, A5, A5, E6, A4, A4, A4, A4, A4, A4, A4, A5, A5, D6, G4, G5, D7, G5, D7, G5, G5, G5, G5, D7, G5, G4, G5, F7, G5, F7, G5, F4, F4, F4, F4, F4, F4, F4, F5, F5, F5, F5, F4, F4, F4, F4, F4, F4, E4, E5, E6, E6, E5, E6, E5, E6, E6, B6, E6, E6, B6, E6, B6, E7, B6, E6, E6, C7, D4, D4, F#6, D4, D4, D6, D4, D4, D4, D4, D6, D4, D4, D5, D4, D4, D5, D4, C5, C4, C5, C4, C5, C5, C4, C4, C5, E6, C5, A#6, C5, A#6, C6, G5, A#6, C5, E6, D7, G5, E6, G5, E6, E6, C5]

Propuesta cuarta: frecuencia con primeros armónicos de mayor amplitud media

Colocando el orden cuadrático del algoritmo en el punto de mira, cabía entonces preguntarse si podía afinarse, por un lado la eficiencia, y por otro la eficacia tratando evitar de alguna manera la penalización a las frecuencias graves como consecuencia, tal y como se explicó, del mayor número de armónicos en el mismo intervalo de frecuencias.

Para lograrlo, una solución simple pero ingeniosa derivó de la idea de inutilizar la división por el número de amplitudes sumadas por medio de la igualación de dicho número de sumas; en resumidas cuentas, considerar tan solo hasta un cierto número de los primeros armónicos de la frecuencia examinada, asumiendo que la información relevante de la señal está contenida, principalmente, en el conjunto de sus primeros armónicos. De esta manera, en la mayoría de los casos (salvo frecuencias agudas extremas), el número de sumas realizadas será siempre el mismo y, por tanto, el orden de complejidad del algoritmo se tornará lineal, $O(n)$, y la antigua penalización pasará meramente a convertirse en una homotecia constante que no intervenga en absoluto en la comparación y en el subsecuente cálculo del máximo.

Así, establecido *NUM_FIRST_HARMONICS* el número máximo de primeros armónicos a considerar como nuevo parámetro de entrada, encontramos, inicializado al valor 5, la siguiente salida que, aun falible, ciertamente supone una mejora con respecto a la anterior:

[C4, C4, C4, C4, C4, C4, C5, C4, C4, C4, C4, C5, C4, C4, C3, D#-1, D3, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, Bo, D#0, D#-1, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, G#4, G4, G4, G4, G4, G4, G5, G4, G4, G4, G5, G4, G4, G4, D#0, G4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, D#-1, D#-1, D#-1, D#-1, D#-1, D#-1, D#-1, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, D#-1, D#-1, D#-1, D#-1, D#-1, D#-1, C5, C5, C5, C5, C5, C5, C5, C5, C5, C5, C5, C5, C5, C5, C5, C5, D#-1, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, D#0, D#-1, D#-1, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, Bo, Bo, Bo, G4, G4, G4, G4, G4, G4, G4, G4, G4, G4, G4, G3, G5, D#0, G4, G4, F4, E4, E5, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, D#-1, D#-1, D#-1, D#0, D#-1, B1, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D#-1, D4, D4, D4, D4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, D#0, C4, D#-1, C4, C4, D#-1, D#-1, D#-1, D#0, D#0, D#0, D#-1]

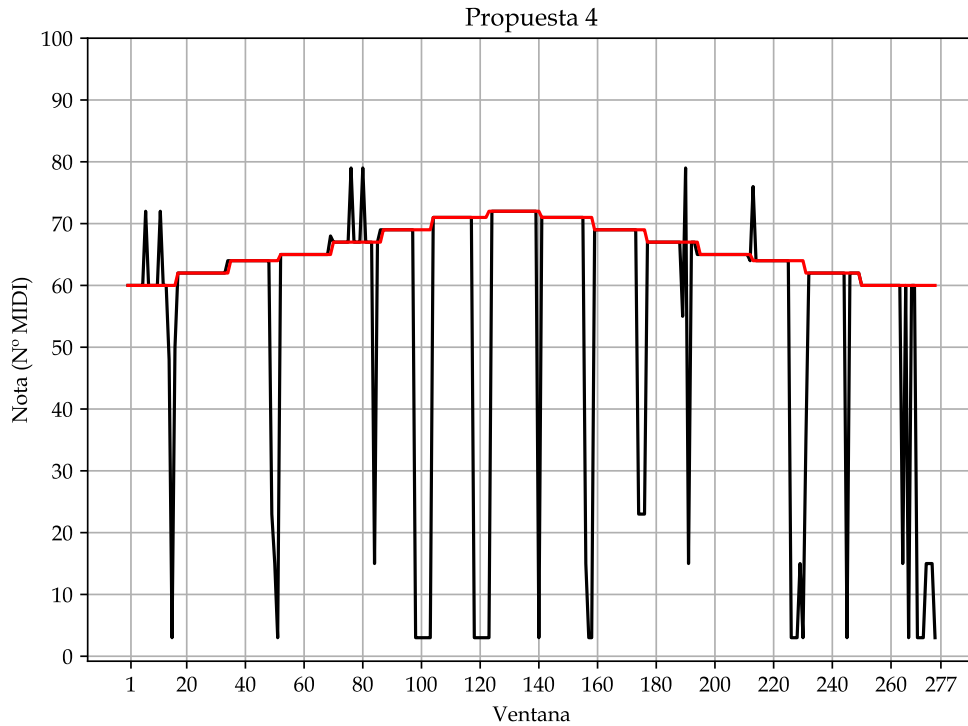


Figura 3.14.: Comparativa entre el desempeño de la propuesta cuarta y la clasificación real.

Propuesta quinta: frecuencia del sistema temperado con armónicos de mayor amplitud media

En discrepancia aún con la precisión del algoritmo, un análisis profundo y exhaustivo del problema llevó irremediablemente, y como se advirtió, a la elección del tamaño de la ventana, revelándose una fuerte dependencia de este valor por parte del enfoque previo.

La causa se halla en la dispersión de los armónicos de la fundamental de la ventana que, para un tamaño como el considerado, 0.2 segundos, se corresponde con $\frac{1}{0.2} = 5$ Hz. En este ejemplo, se dispone de evaluaciones de la FFT para toda frecuencia múltiplo de 5 Hz pero, el resultado de incrementar o decrementar un semitono, a medida que nos aproximamos a las frecuencias graves, dista menos cada vez de la frecuencia a la que se aplica (precisamente por la naturaleza multiplicativa del intervalo físico). A modo de ejemplo, si el armónico seleccionado por el algoritmo anterior fue 125 Hz y se aproximó por Si_2 (123.47 Hz), quizá realmente la falta de información en el entorno de estos puntos llevó a descartar un posible real Do_3 (130.81 Hz).

Resumiendo, la pérdida de precisión conforme reducimos el valor de la frecuencia es un factor que impulsó el cambio de paradigma hacia un nuevo enfoque más específico al problema que nos ocupa y, especialmente, al ámbito del temperamento igual.

Concretamente, el nuevo procedimiento itera ya no sobre los armónicos de la fundamental de la ventana, sino sobre las frecuencias que integran el temperamento igual. Para cada una

3. Implementación del conversor WAV-MIDI

de ellas, se interpolarán los valores de amplitud de sus armónicos en base a la información discreta que se posee y se computará la media de forma similar a como se propuso en la alternativa tercera. A este respecto, puesto que dividir por el número de armónicos contabilizados suponía una penalización excesiva a las frecuencias graves, se propone intercambiar, con ánimo absolutamente experimental, el denominador por la raíz cuadrada de dicho número, en pos de suavizar la penalización. A la luz de las pruebas realizadas, resulta curiosamente en una mejora sustancial de la eficacia del algoritmo.

Y, por otro lado, cabe detallar el proceso de interpolación mencionado en pos de estimar la amplitud de una frecuencia arbitraria (recordemos que solo disponemos de aquellas asociadas a los armónicos de la frecuencia fundamental de la ventana). En el caso trivial en el que la amplitud de la frecuencia sea conocida, habríamos terminado.

En caso contrario, utilizaremos una media ponderada de las amplitudes conocidas de las dos frecuencias más cercanas a ella (entre las que se encuentra) por la distancia a cada una. Dicha media ponderada se realiza, lógicamente, en la forma lineal, luego habremos de aplicar una transformación logarítmica que induzca el tratamiento deseado. En cuanto al cálculo de estas dos frecuencias más cercanas, se recomienda la implementación recursiva por búsqueda binaria; en cualquier caso, conocidas y notadas como f_1 y f_2 , siendo f la frecuencia cuya amplitud se espera estimar y F la función amplitud (norma de la FFT), definimos \hat{F} la estimación de F como sigue:

$$\hat{F}(f) := \frac{F(f_1) \log_2 \frac{f_2}{f} + F(f_2) \log_2 \frac{f}{f_1}}{\log_2 \frac{f_2}{f_1}}$$

Con los valores estimados de amplitud para cualquier frecuencia, la iteración sobre las frecuencias del temperamento igual y la penalización relajada vía la raíz cuadrada en el denominador, obtenemos el siguiente resultado, todavía con una complejidad computacional reprochable:

[C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, A#0, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D4, D5, D4, D4, D4, D4, C1, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E5, E5, E4, Ao, F4, F4, F4, F4, F4, F4, F5, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, Bo, G4, G4, G5, G4, G4, G4, G5, G4, G4, G4, G5, G4, G4, G4, G4, G4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A5, A4, A4, A4, A4, A4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B5, B4, B4, B4, B4, Ao, Ao, D1, C5, C5, C5, C5, C5, C6, C5, C5, C5, C5, C5, C5, C5, C5, C5, C5, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, B4, A1, B4, B4, A#0, Bo, A4, A4, A4, A4, A5, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, A4, Ao, C1, G4, G4, G4, G4, G4, G4, G4, G4, G4, G4, G4, G4, G5, G4, G4, G4, C1, F4, F4, F4, F4, F4, F4, F4, F4, F5, F4, F4, F4, F4, F4, F4, F4, Ao, F1, E4, E4, E4, E4, E4, E4, E5, E4, E4, E4, E4, E4, E4, E4, Ao, A1, Ao, D4, D4, D4, F#6, D4, D4, D4, D4, D4, D4, D4, D4, D5, D4, D4, D5, D4, C4, A#0, F1]

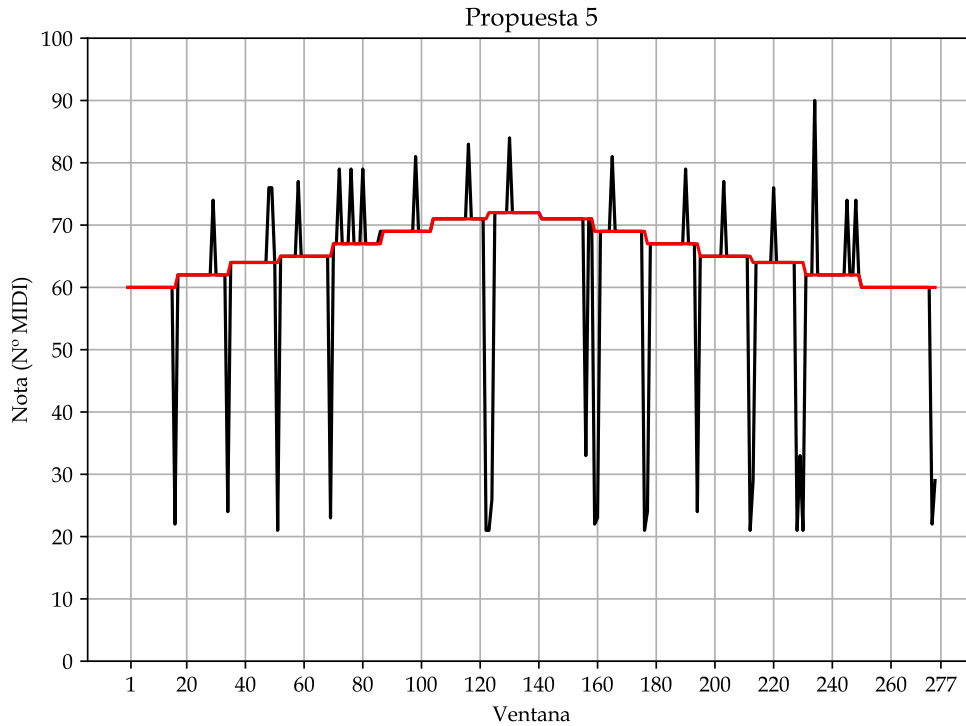


Figura 3.15.: Comparativa entre el desempeño de la propuesta quinta y la clasificación real.

Como puede verse, comenzamos a vislumbrar cuantiosas mejoras en lo que a precisión refiere.

Propuesta sexta: frecuencia del sistema temperado con primeros armónicos de mayor amplitud media

La cuestión natural que surge ahora, especialmente teniendo en cuenta la trayectoria seguida, es si podemos seguir mejorando la eficiencia y la eficacia, gracias a eliminar la penalización, tal y como se efectuó para la propuesta cuarta. Así, la presente alternativa propone tomar el modus operandi de la quinta y calcular, para cada frecuencia considerada en el sistema temperado, la media de las amplitudes de un número prefijado de sus primeros armónicos.

Para `NUM_FIRST_HARMONICS` con un valor de 5; es decir, tomando únicamente los cinco primeros armónicos, regresamos a orden lineal, $O(n)$, y el resultado obtenido es el siguiente:

[C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, D4, D4, D4, D4, D4, G2,
D4, D4, D4, D4, D4, D4, G2, D4, D4, D3, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4,
E4, E4, E4, E4, E4, F4, F4, F4, F4, F4, F4, F4, F4, F4, A#2, F4, F4, F4, F4, F4, F4, C#3, G4, G4,
C4, G4, G4, G4, C4, G4, G4, G4, C4, G4, G4, C4, G4, G4, A4, A4, A4, A4, A4, A4, A4, A4, A4,
A4, A4, A4, A4, A4, A4, Ao, A4, A4, E3, E3, B4, E3, B4, B4, B4, E3, E3, B4, B4, B4, E4, B4, B4,
Ao, Ao, Ao, Ao, C5, C5, F3, F3, F3, C4, F3, F4, F3, C3, F3, F3, F3, F3, F3, F3, F3, E3, E3, B4,
E3, B4, E3, B4, B4, B4, B4, B4, Ao, B4, B4, Ao, B4, B4, G#3, D3, A4, A4, A4, A4, A4, A4, A4,

3. Implementación del conversor WAV-MIDI

A4, A4, A4, A4, A4, A4, A#0, A#0, B0, G#2, G4, C4, G4, G4, G4, C4, G4, G4, C4, G4, G4, G4, C4, A#0, G4, G4, C4, A#2, F4, A#2, F4, F4, F4, F4, F4, F4, F4, F4, F4, F4, A#2, A#2, F4, F4, E4, E4, E4, E4, E4, E4, E4, E4, E4, E4, A0, E4, E4, E4, A#0, A0, E4, D4, D4, D4, F#4, D4, D4, D4, G2, D4, D4, D4, D4, D4, D4, D4, D4, D3, D4, D4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, C4, A0, C4, C4, C4, C4, A0, C4, C4, C4, C4, B0, F1]

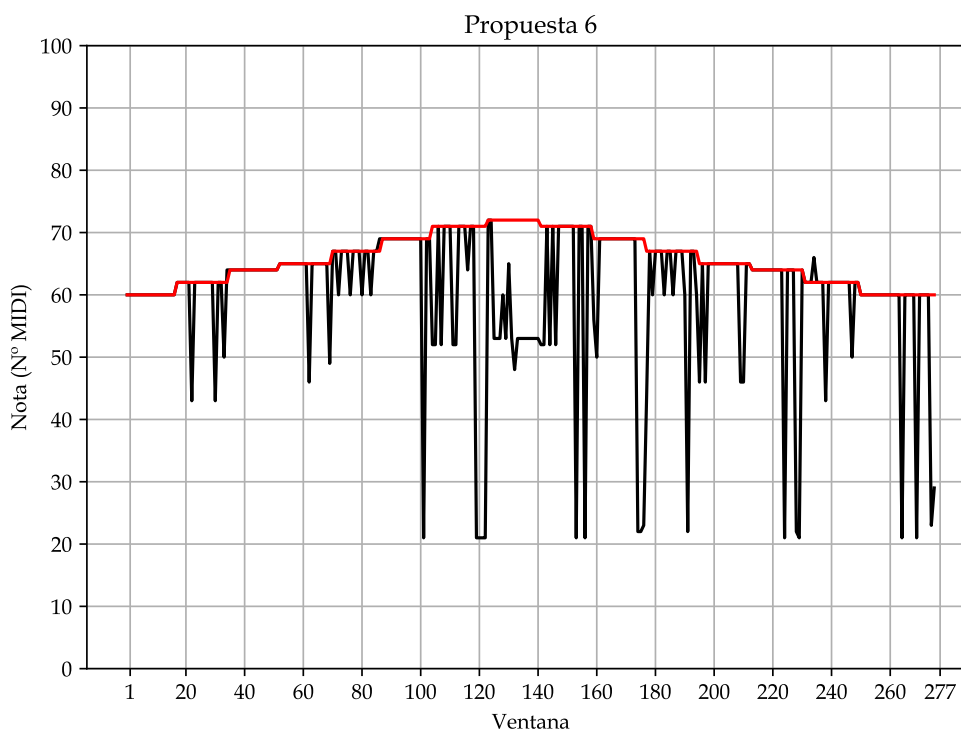


Figura 3.16.: Comparativa entre el desempeño de la propuesta sexta y la clasificación real.

A estas alturas, disponemos de un conjunto de alternativas aceptable con el que proceder. Sin embargo, deber escoger entre unos tiempos de ejecución deplorables y seleccionar un número máximo de armónicos con los que trabajar no es del todo aceptable, dado que no resulta trivial prefijar este valor en cualquier caso. Esta reflexión condujo a investigar alternativas en otros ámbitos.

Propuesta séptima: autocorrelación

Con el objetivo en mente de mejorar aún más la eficacia, cabe introducir el concepto de **autocorrelación**, técnica que guarda relación con la **convolución** ([Con23]) de funciones medibles, definida como sigue:

Definición 3.4. Sean $f, g \in \mathcal{L}(\mathbb{R})$ T -periódicas. Dado $x \in \mathbb{R}$ se define, si tiene sentido hacerlo, la **convolución de f y g en x** como

$$(f * g)(x) = \frac{1}{T} \int_0^T f(x-y)g(y) \, dy$$

Se puede probar, si las funciones f y g en las hipótesis son integrables en $[0, T]$, que $f * g$ está definida *c.p.d.* en \mathbb{R} y que, a su vez, resulta integrable en $[0, T]$ y T -periódica.

En cualquier caso, lo que verdaderamente nos ocupa es inteligir que la convolución consiste en la media del producto de una función con la versión desplazada de otra función a lo largo de un período.

No obstante, quizás realmente podríamos trazar su origen en el producto escalar de $L_2(\Omega)$ como medida de similitud entre pares de elementos o vectores de dicho espacio vectorial. Recordemos:

$$\langle [f], [g] \rangle := \int_{\Omega} f \bar{g} \quad \forall [f], [g] \in L_2(\Omega)$$

Inspirados en todo esto y, en particular, en la convolución de una función consigo misma, se tiene que

Definición 3.5. Dada una señal $f : \{0, 1, \dots, N-1\} \rightarrow \mathbb{R}$, con $N \in \mathbb{N}$, llamamos **autocorrelación de f** a la señal

$$r_f(n) = \sum_{k=0}^{N-1} f(k)f(k+n) \quad \forall n \in \{0, 1, \dots, N-1\}$$

Un apunte interesante que cabe realizar sobre la definición anterior en primer lugar es que, para mayor fidelidad con respecto a lo que conocemos como convolución, sería lógico pensar que debiéramos multiplicar la sumatoria por el factor $\frac{1}{N}$ con la finalidad de computar una media como tal. No obstante, la literatura suele obviar este factor, si bien realmente se trata de una homotecia constante que no aporta a nuestra tarea. De hecho, tiente hacia la definición del producto escalar.

Y, al margen de lo anterior, es razonable cuestionar el interés de esta nueva señal. Para introducir el contexto, la señal de partida f no será otra que alguna de las ventanas en que fragmentamos nuestro archivo de audio. Según se ha operado hasta el momento, se procedía al cálculo de la FFT y se aplicaba el algoritmo de turno. Ahora, se propone calcular la FFT sobre la autocorrelación de la ventana, lo cual se conoce como **densidad espectral de potencia**, pues las propiedades que presenta esta nueva señal podrían ser de utilidad. Hablamos, fundamentalmente, de servir, como se adelantaba, a modo de medida de similitud entre la señal y una versión desplazada de sí misma.

Intuitivamente, desplazar la misma señal en un período que corresponda a una frecuencia relevante en su descomposición espectral hará que r_f garantice una gran similitud para dicho desplazamiento y este patrón no pasará desapercibido por el análisis que ofrece la densidad espectral de potencia vía la FFT. Por el contrario, la similitud para otros desplazamientos será reducida, luego, en definitiva, tras la FFT se conservan las amplitudes de frecuencias relevantes y se aminoran o suavizan aquellas de frecuencias no relevantes con respecto a una aplicación directa sobre la señal de partida, como puede verse en las figuras que siguen.

3. Implementación del conversor WAV-MIDI

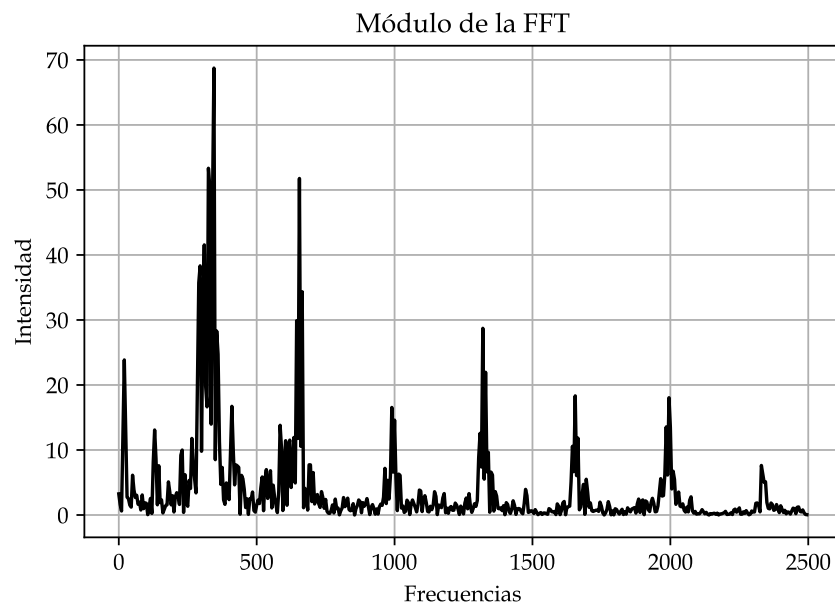


Figura 3.17.: Análisis espectral habitual de una ventana de transición entre un sonido y el siguiente.

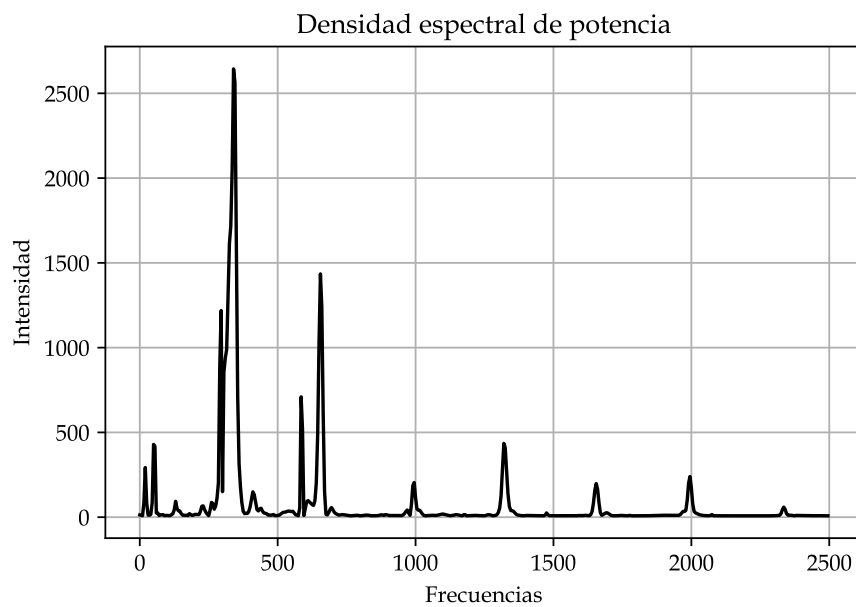


Figura 3.18.: Densidad espectral de potencia de la ventana mencionada en 3.17. ⁴

⁴Nótese el sorprendente suavizado que produce la técnica de autocorrelación en este ejemplo.

La aplicación de esta herramienta combinada con el enfoque quinto da como resultado la siguiente asignación:

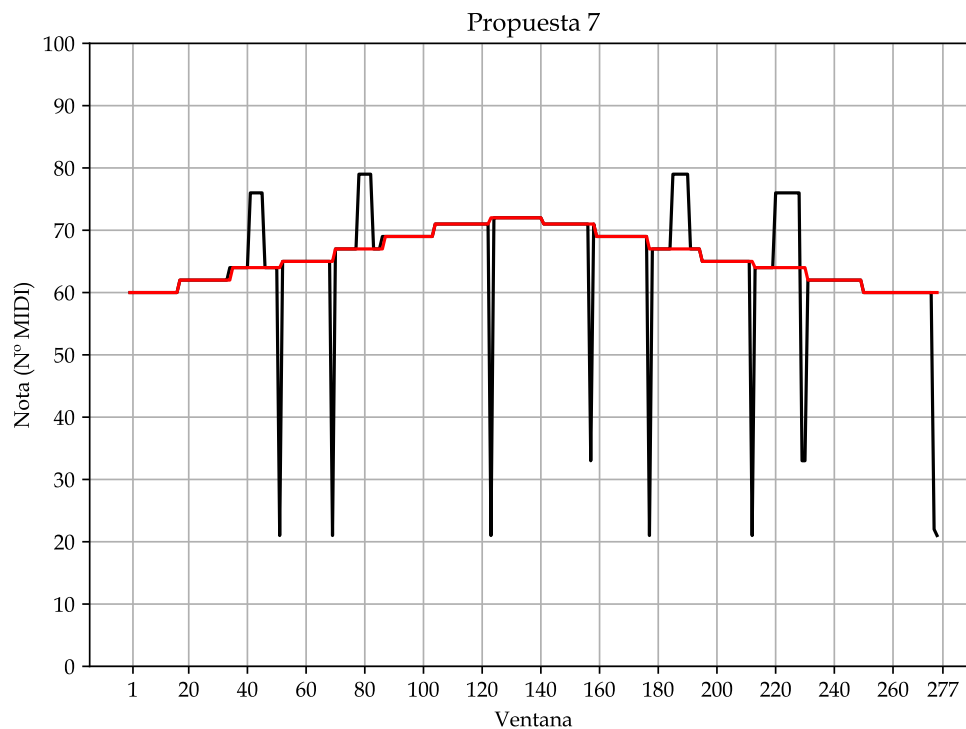
[illegible]

Figura 3.19.: Comparativa entre el desempeño de la propuesta séptima y la clasificación real.

Aún, como se observa, esencialmente para Mi_4 (E) y Sol_4 (G), existen frecuentes errores fundados en la confusión de la fundamental real y su segundo armónico, la octava.

Llegados a este punto, las ventanas mal clasificadas exhibían un nivel de patología tal, que las descompensaciones en cuestión de amplitud de armónicos superaban en algún momento de toda ejecución a cualquiera de las diferentes variaciones de parámetros empíricamente

3. Implementación del conversor WAV-MIDI

abordadas. No obstante, aunque dichos errores podrían considerarse, de cierta manera, casos extremos "despreciables", lo que impulsó el último cambio de dirección hacia el algoritmo final fue el hecho de que, aún con la vileza del análisis espectral, a la vista seguía siendo perceptible el valor de la frecuencia fundamental.

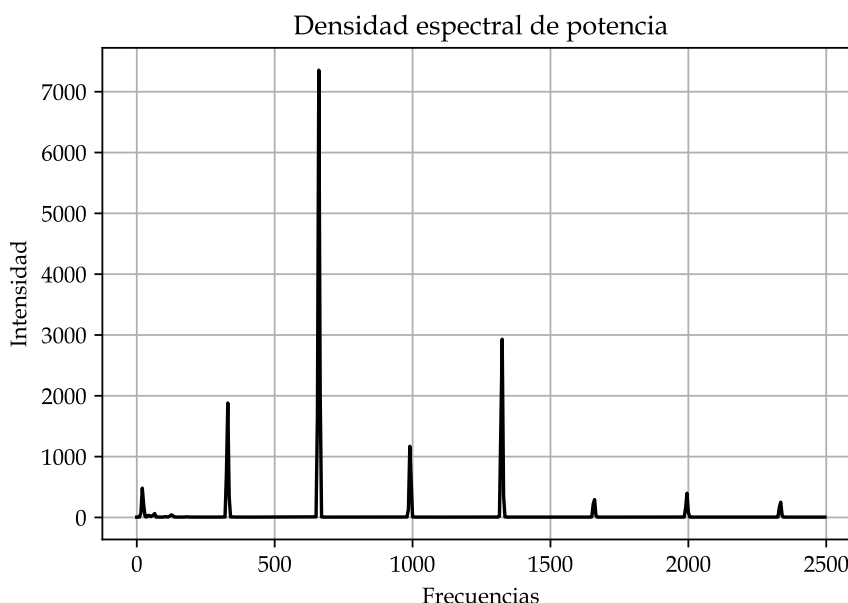


Figura 3.20.: Densidad espectral de potencia correspondiente a la primera ventana clasificada erróneamente como Mi_5 .⁵

Así, el impulso último a la investigación vino de la mano de un cambio radical de perspectiva: en lugar de focalizar los esfuerzos en tratar de descifrar e imitar el complejo proceso que al oído hace percibir un cierto sonido (lo cual, además, podría verse turbado por factores psico-acústicos), colocar la mira en **cómo a los ojos la frecuencia fundamental no pasa desapercibida en la gráfica de análisis espectral**. Y, para ello, una de las claves del algoritmo ha de residir en dar valor a la **forma** de la gráfica y no buscar métricas en base a las amplitudes de los armónicos.

Propuesta octava: aprendizaje no supervisado

Con la pretensión de atender a la forma de la FFT en mente, se regresa al problema abstracto inicial acerca de cómo dilucidar qué picos de la FFT son relevantes y cuáles no, y se elabora un plan en cuatro fases que consituye la base del algoritmo definitivo.

Etapas 1.- Clasificación de picos relevantes

⁵La verdadera fundamental en este caso es Mi_4 (329.63 Hz). No obstante, y pese a ser perceptible a la vista dado que existen picos en cada uno de sus armónicos, la mayor intensidad encontrada para los armónicos pares impide al algoritmo del enfoque quinto dar otra respuesta que no sea la octava, Mi_5 .

Ya la alternativa segunda había dejado patente que establecer un umbral sencillo no serviría de nada. Así, se decide recurrir a la artillería pesada que representa el aprendizaje no supervisado. Se utilizará un procedimiento de **clustering** con el objetivo de clasificar los picos de la FFT en dos grupos: relevantes y no relevantes.

Dado que, además, trabajaremos de manera serena sobre una dimensión porque tan solo interesa clasificar la imagen de la FFT, contenida en \mathbb{R} , es clara la elección del algoritmo que aplicaremos: *K*-Means, para $K = 2$. Y la claridad en la elección radica en las diversas ventajas que coloca encima de la mesa:

1. Conocemos el número de clusters en que pretendemos clasificar nuestro conjunto de datos, luego la principal exigencia de *K*-Means, el establecimiento del parámetro *K*, nos es indiferente.
2. *K*-Means es un algoritmo sencillo y, en honor a su simplicidad, rápido.
3. Pretendemos establecer un umbral para picos relevantes, lo que implica dividir \mathbb{R} en dos semirrectas, que son intervalos reales. Dado que en \mathbb{R} los únicos conjuntos convexos son los intervalos, trabajaremos en todo momento sobre conjuntos convexos. Afortunadamente, *K*-Means, por ser un algoritmo basado en centroides, solo opera adecuadamente con agrupaciones convexas, y es precisamente el entorno en que nos movemos.

Así, la etapa primera consiste en la búsqueda de máximos en la FFT (en clave discreta, serán los puntos para los que el anterior y el siguiente son menores o iguales en altura), para lo cual utilizaremos el método *find_peaks*, del módulo *scipy.signal*, y la aplicación de *K*-Means con $K = 2$ sobre el conjunto de máximos, tras lo cual se recupera el cluster que contiene al valor máximo de la FFT y se toman las frecuencias asociadas a esos picos.

Etapa 2.- Aproximación a temperamento igual y eliminación de duplicados

Las frecuencias obtenidas en la etapa primera, que se corresponden con las asociadas a picos relevantes, son armónicos, recordemos, de la frecuencia fundamental de la ventana. En consecuencia, es el momento idóneo para redondear cada una de ellas a temperamento igual, instante tras el cual ya podríamos barajar en nuestro conjunto la posesión de la fundamental real... o no. En cualquier caso, procede aplicar un método de eliminación de duplicados que puedan surgir en algún momento como consecuencia de haber realizado la aproximación.

Etapa 3.- Obtención de los verdaderos candidatos

Pero, como ya se advirtió, podría darse el caso de que la intensidad de la fundamental real audible no fuese lo suficientemente elevada como para que nuestro algoritmo de clustering la tomase por relevante. En tal caso, de no prever ningún procedimiento a la contra de estos escenarios, el algoritmo estaría siempre errado a su merced.

Es, por ello, que nos apoyaremos en una hipótesis extra que nos permitirá solventar este quiebro: **si la intensidad de la fundamental real no es lo suficientemente elevada cabe pensar, al menos, que se dispondrá de dos armónicos suyos consecutivos en el conjunto de picos relevantes**. Y, de acuerdo a esto, lo único que se habrá de emprender para hacer manar

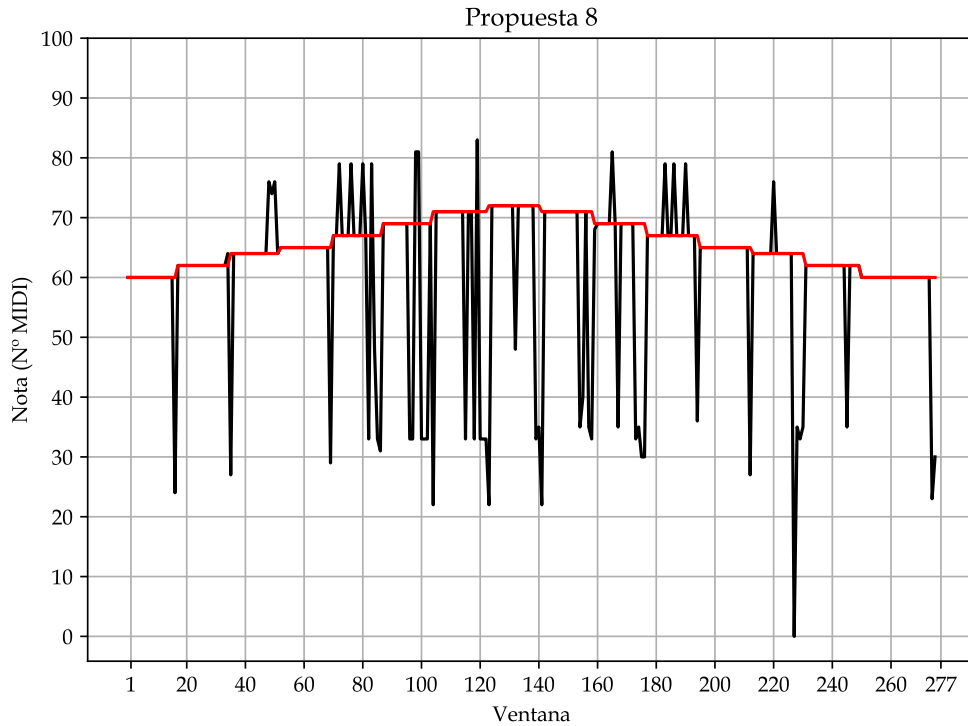


Figura 3.21.: Comparativa entre el desempeño de la propuesta octava y la clasificación real.

Pero, recordemos que la motivación para este radical cambio de paradigma fueron los errores cometidos en ventanas cuya FFT promocionaba agresivamente algún armónico de la fundamental real, como es el caso de la octava. Pues, desgraciadamente, el presente enfoque es igualmente burlado por este tipo de ventanas, si bien la complejidad del asunto desafía hasta los límites de *K-Means* que, ante estos escenarios, construye un cluster de picos relevantes con un único pico, el del máximo absoluto, del que no se puede extraer la fundamental real.

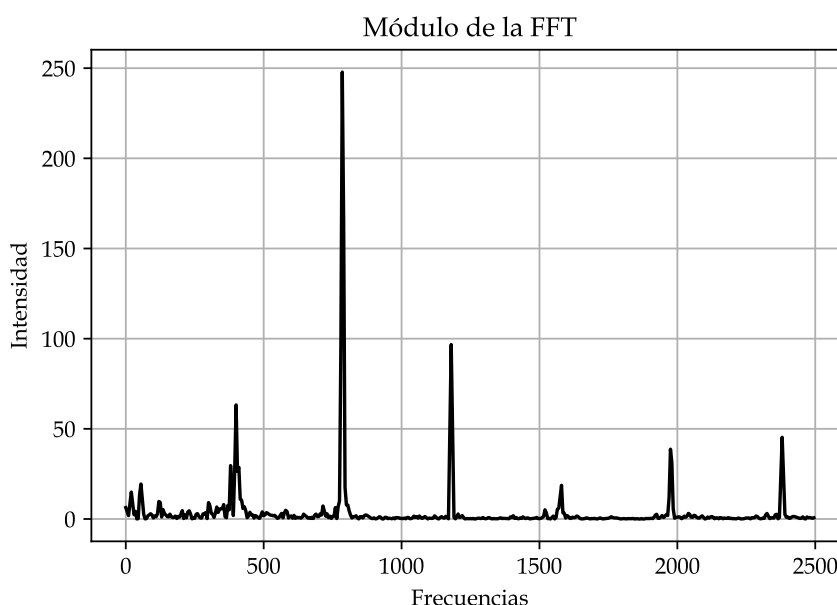


Figura 3.22.: Ventana erróneamente clasificada en esta propuesta. ⁶

Así, se propone una última vuelta de tuerca que configura el algoritmo final, aportando una solución definitiva a este conflictivo asunto, así como un armazón de mejoras que elevan la eficacia de manera ampliamente satisfactoria.

Propuesta novena: *K*-Means dinámico y otras mejoras

Para finalizar el algoritmo de detección de la frecuencia fundamental, fruto de todo el trabajo previo, tomamos la propuesta anterior y realizamos las siguientes modificaciones:

1. La mejora principal que apunta a mejorar la precisión del algoritmo radica en arreglar la ocasional poquedad de resultado de *K*-Means, fijo $K = 2$. Claro que, aumentar dicho valor sin más carece totalmente de sentido.

Sin embargo, se sugiere ahora utilizar un parámetro nuevo *MAX_REL_PEAKS* para gestionar, grosso modo, el número de picos que, como máximo, pretendemos tildar de relevantes. De esta manera, es posible proceder de forma iterativa, obligando a una primera clasificación con $K = 2$ y aumentar el valor de K si, para la clasificación $K + 1$, el número de picos relevantes, conjunto definido como la unión de todos los clusters salvo aquel que contiene al mínimo absoluto de la FFT, no supera el valor *MAX_REL_PEAKS* prefijado.

Con ello, suplimos el corto alcance del umbral primero gestionando un descenso progresivo del mismo hasta alcanzar la mayor división que no supere el valor máximo establecido de picos relevantes. Como medida de seguridad, se controla igualmente

⁶El umbral propuesto por *K*-Means con $K = 2$ tan solo abarca el pico más prominente, que se corresponde con la octava de la fundamental real, *Sol*₄.

que el número de iteraciones realizadas no supere un cierto valor, pero este detalle no es relevante en la discusión que nos ocupa.

2. Comprometidos, además, con la mejor clasificación de picos relevantes, podemos apoyar al método realizando previamente la autocorrelación de la ventana que, de acuerdo a lo explicado previamente, suavizará el ruido generado por frecuencias poco presentes al tiempo que mantiene los picos asociados a las frecuencias significativas.
3. En la etapa tercera, para ajustar más rigurosamente el procedimiento de generación de candidatos a la hipótesis propuesta basada en la existencia de armónicos consecutivos, tras ordenar la salida de la etapa segunda, bastará simplemente con añadir las diferencias entre cada par de elementos sucesivos, y no contar con las diferencias positivas entre cada par de frecuencias del conjunto.
4. En la etapa cuarta alteramos ligeramente el conteo para calcular, en lugar del número de armónicos de cada candidato en el conjunto de candidatos definitivo, el número de armónicos de cada candidato en el conjunto de picos relevantes, ya aproximados, obtenido en la etapa segunda (en el cual el conjunto de candidatos está contenido). Las razones son, en primer lugar, asegurar que, de por sí, los armónicos se cuentan entre los picos relevantes y, en segundo lugar, evitar que las frecuencias extrañas originadas en las diferencias a causa de la generación de candidatos obtengan conteos más altos que la frecuencia fundamental.

Con todas estas modificaciones, y para $MAX_REL_PEAKS = 12$ el resultado evoluciona significativamente, como puede observarse:

[illegible]

3. Implementación del conversor WAV-MIDI

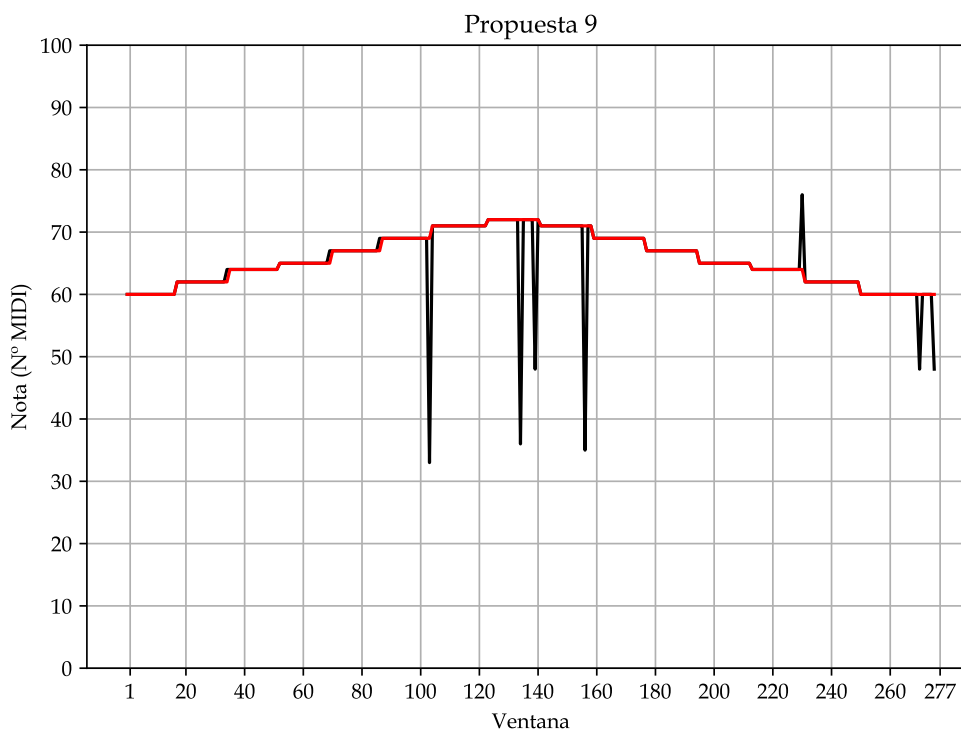


Figura 3.23.: Comparativa entre el desempeño de la propuesta novena y la clasificación real.

No obstante, este tan logrado resultado sobre el objetivo del capítulo, la detección de la frecuencia fundamental real para cada ventana, se ha edificado poco a poco tan solo con la información de que se dispone de la propia ventana y no, como bien cabe incorporar, con la información extra que puede ofrecer el entorno de ventanas vecinas a ella. En consecuencia, la detección de tonos admite un nuevo nivel de mejora, que se introduce ahora en el capítulo de corrección.

3.3.3. Corrección de tonos. Asignación de silencios

El último de los procedimientos, que trabajará de manera directa con el resultado obtenido en la detección de tonos alude, como ya se introducía, a utilizar la vecindad de un tono para corregirlo, en caso de ser necesario.

Y ciertamente es menester incorporar la información que se puede extraer del entorno de un sonido pues, como físicamente sucede, cada instrumento musical, en la interpretación de un sonido, imprime en él una forma característica de variación de las amplitudes de la onda sonora emitida, que dan lugar a discusiones sobre el **ADSR**: *Attack* (momento de ataque del sonido, ascenso hacia el primer pico de intensidad), *Decay* (decaimiento inicial tras el ataque), *Sustain* (etapa de estabilidad del sonido) y *Release* (momento final de disipación de la energía).

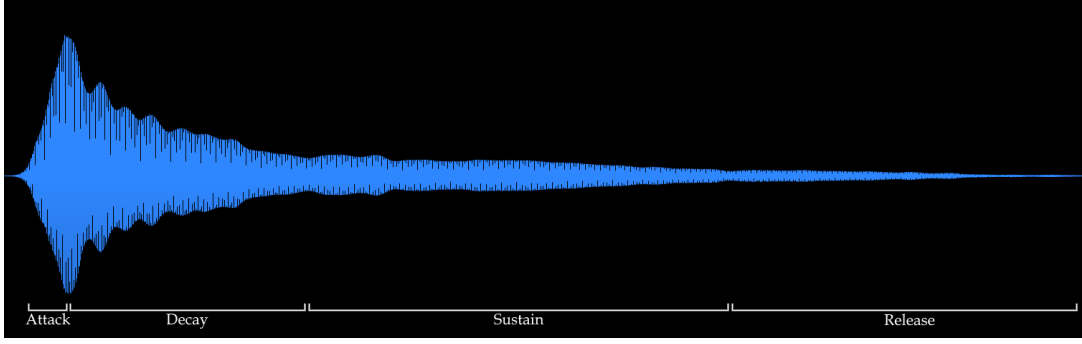


Figura 3.24.: Ejemplo de ADSR de una señal, extraída vía *iZotope RX8*

Y es que ciertas clasificaciones incorrectas podrían hallarse en el *Release* de un sonido, justo cuando la energía se reduce hasta tal punto que el algoritmo anterior no puede dilucidar correctamente lo que acontece en la ventana en cuestión. Así, la importancia de la corrección se halla en utilizar el pasado inmediato de un sonido para decidir si adaptarlo o no.

Motivada la introducción de este mecanismo, habremos de discutir el método empleado para la corrección. En primer lugar, se entiende razonable calcular alguna métrica de certeza, a este respecto, sobre la detección de cada ventana. Asimismo, una vez obtenida la fundamental estimada, se calculará **el máximo de las amplitudes de sus armónicos**, que hará las veces de dicha medida de fiabilidad o certeza (calculado estimando la amplitud desconocida de tales armónicos como se propuso en la alternativa quinta).

El porqué se encuentra en que, si la ventana abarca un momento de *Release*, la amplitud general de la FFT será baja, luego susceptible la nota de modificación en base a la medida propuesta. Y si, en un escenario más general, la frecuencia estimada resulta ser errónea, entonces cabe esperar que la máxima amplitud de los armónicos de esta no sea superior a la máxima amplitud de los armónicos de la fundamental real, siendo susceptible de modificación de nuevo para nuestro algoritmo de corrección.

El algoritmo implementado admite como parámetros el número de ventanas pasadas y futuras a considerar para la corrección ordenada de cada nota. Sin embargo, ni se encuentran motivos físicos reales para incorporar ventanas futuras, ni la puesta en práctica apoya su inclusión, desmejorando notablemente el resultado primero obtenido de la detección de frecuencias. Es por ello que se omitirá su uso de cara a la explicación teórica y se establecerá, por defecto, dicho número de ventanas futuras a cero.

Pues bien, dado nl el número de ventanas pasadas a utilizar y siendo k el índice de una ventana tal que $k \geq nl$ asignamos una ponderación, para cada ventana indizada en el conjunto ordenado $\{k - nl, k - nl + 1, \dots, k\}$, de acuerdo a la sucesión:

$$p_0 := 1, p_n := \sum_{m < n} p_m \quad \forall n \in \{1, \dots, nl\}$$

De esta manera, se asigna peso 1 a la primera ventana. Como la segunda ha de tener, al

3. Implementación del conversor WAV-MIDI

menos, el mismo peso que ella en la ponderación por ser más cercana a la ventana k , se le asigna igualmente peso 1. La contribución de la tercera, que es aún más próxima, ha de ser igual o superior a la de las dos anteriores juntas para no verse comprometida por la conjunción de ambas, que son anteriores a ella, luego se le asocia el peso 2. Y así, sucesivamente, se calculan las ponderaciones de cada ventana en la vecindad.

Finalmente, en aras de corregir la nota de la k -ésima ventana se realiza, para cada una de las notas diferentes detectadas en dicho conjunto, la suma de los productos de los pesos en cada ventana en que fueron detectadas (recordemos, el máximo de las amplitudes de sus armónicos) por las ponderaciones anteriormente definidas correspondientes; es decir, para una nota con Δ el conjunto de índices de ventanas a las que se asocia y M_i la medida de certeza de la detección en la ventana i ,

$$P := \sum_{i \in \Delta} p_{i-(k-nl)} M_i$$

tras lo cual se elegirá la nota con mayor valor de P asociado.

Una vez detallado el proceso, que se ejecuta secuencialmente sobre el conjunto de notas tras la detección, cabe mencionar que este puede encapsularse y repetirse tantas veces se quiera, con tal de definir nuevos pesos para las notas corregidas. No obstante, con los valores P calculados, se estimará fácilmente, para la k -ésima ventana, y hablando respecto de la nota finalmente elegida, como

$$M_k^{(2)} := \frac{P}{\sum_{i \in \Delta} p_{i-(k-nl)}} = \frac{\sum_{i \in \Delta} p_{i-(k-nl)} M_i}{\sum_{i \in \Delta} p_{i-(k-nl)}}$$

la media ponderada, en definitiva.

Por último, durante el proceso de corrección, en el que incorporamos la información referente a la intensidad máxima de la FFT en la ventana, es imperativo hablar por fin de un elemento crucial en música y que, hasta el momento, había pasado inadvertido: el silencio.

Para su gestión, incorporamos una nueva variable $SILENCE_THRESHOLD \in [0, 1]$, que representa la proporción del máximo valor hallado entre todas las FFTs calculadas y dará lugar al umbral de silencio. Durante el método de corrección, se hallará dicho umbral como el producto de $SILENCE_THRESHOLD$ por el máximo en el conjunto de los pesos de las notas detectadas, los M_i y, simplemente, para cada nota a corregir, si su peso es menor o igual que el umbral de silencio, será sustituida por la cadena 'S'. Huelga decir que, en base a la clasificación de silencio propuesta, las ventanas previamente asignadas como silencio **no podrán ser corregidas**; dicho de otra manera, serán ignoradas en futuros barridos del algoritmo de corrección. Y, con esto, se da conclusión a la estimación de notas y silencios para cada ventana.

Por fin, es la hora. Tras un tortuoso trayecto, el resultado final, habiendo realizado una única corrección y para $MAX_REL_PEAKS = 12$, $SILENCE_THRESHOLD = 0$ (sin silencios) y $nl = 4$, es el siguiente:

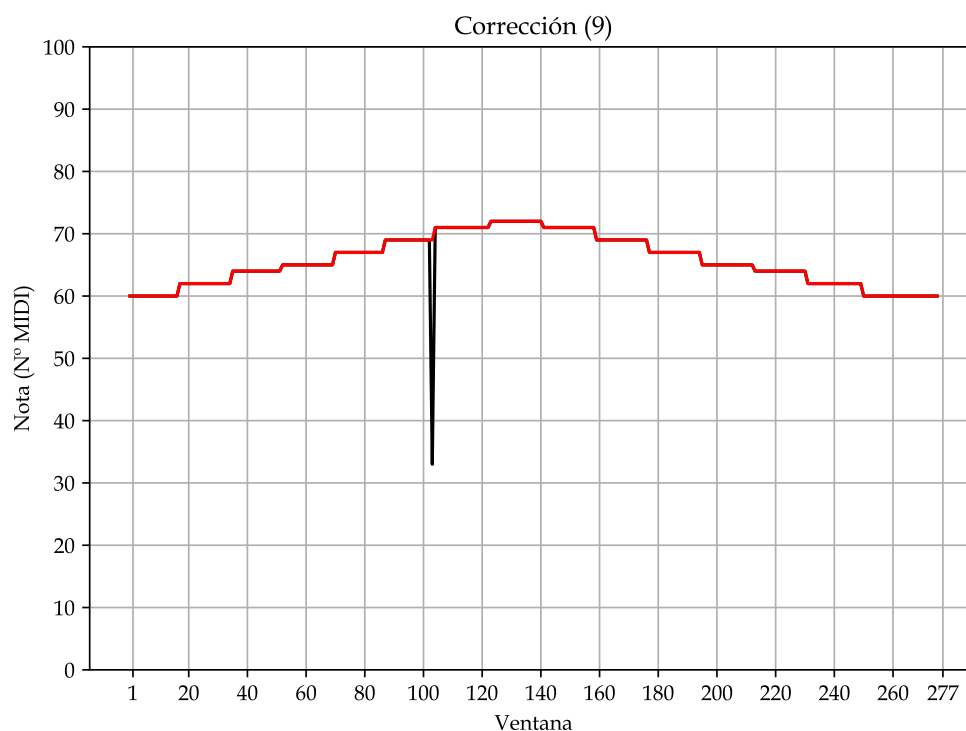
[illegible]

Figura 3.25.: Comparativa entre el desempeño de la propuesta novena con corrección y la clasificación real.

3.3.4. Estudio comparativo

Antes de proseguir con las secciones finales del algoritmo de conversión, se realiza el estudio comparativo de las precisiones de las diferentes alternativas comentadas en base al resultado final esperado para el ejemplo propuesto.

3. Implementación del conversor WAV-MIDI

	Precisión (%)
Propuesta 1	53.6231884057971
Propuesta 2	62.68115942028985
Propuesta 3	47.10144927536232
Propuesta 4	80.07246376811594
Propuesta 5	86.23188405797102
Propuesta 6	74.6376811594203
Propuesta 7	86.59420289855072
Propuesta 8	78.98550724637681
Propuesta 9	96.3768115942029
Corrección (g)	99.6376811594203

A continuación, se muestra un gráfico equivalente que resultará de mayor interés:

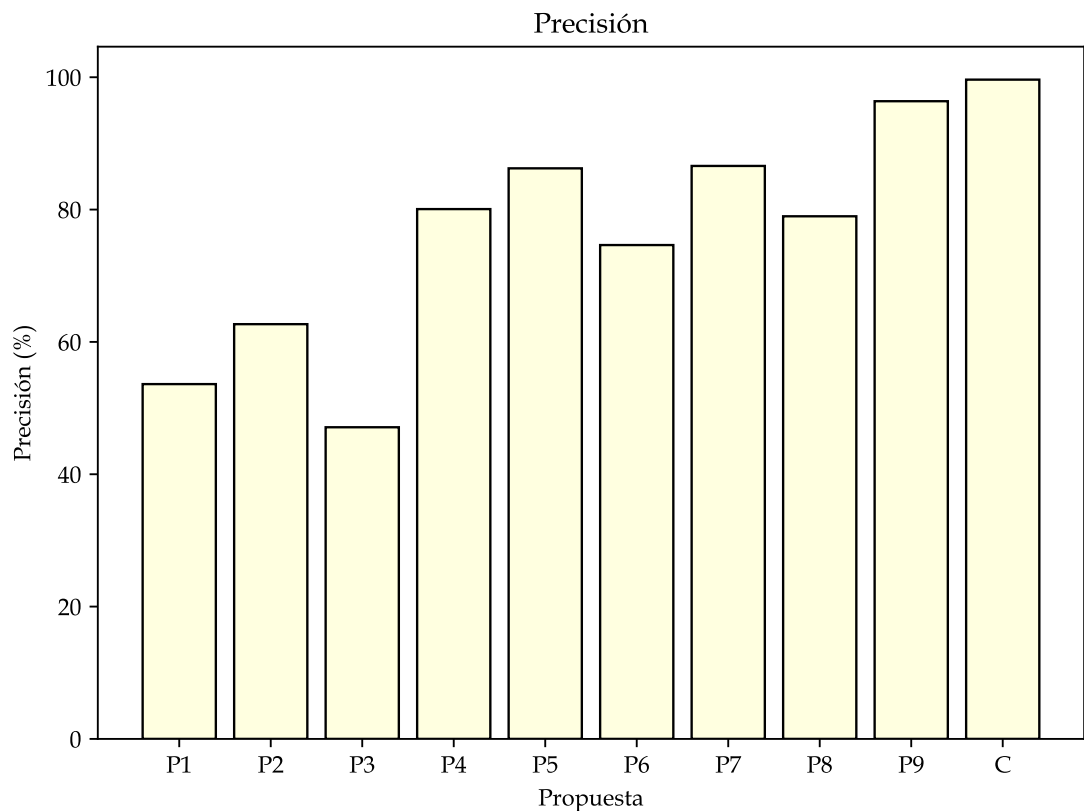


Figura 3.26.: Precisiones de las distintas propuestas aplicadas al fichero de ejemplo.

Como puede verse, en este caso la última de las versiones arroja el mejor valor de precisión, con un desempeño quasi perfecto.

3.3.5. Cálculo de duraciones

Ha llegado el momento de labrar el conjunto de asignaciones de notas a ventanas, principal materia prima de la sección que comienza. Para ello, habremos de conocer el *tempo* de la melodía, que se pedirá como variable de entrada en **negras por minuto** y se almacenará con el nombre de *QUARTER_PPM*. A partir de ella, se calcula el tiempo que dura una negra en segundos como

$$QUARTER_SECS = \frac{60}{QUARTER_PPM}$$

Ahora, se aplica de manera iterativa un proceso organizado en tres etapas.

Etapas 1.- Cálculo de duraciones en segundos

La primera etapa consiste en el recorrido del conjunto de notas y la asociación de subconjuntos de notas iguales consecutivas como una sola junto con el cálculo del tiempo o duración que le corresponde.

Así, aceptamos como parámetro de este procedimiento un conjunto de notas y un conjunto de duraciones asociadas.

Si este último es vacío, lo cual ocurrirá siempre en primera instancia, se asume que cada nota se relaciona con una ventana, luego se iterará sobre el conjunto de notas contabilizando, para cada nueva nota, el número de notas sucesivas iguales y, en base a esto, se calculará la duración pertinentemente a partir, claro está, de los ya conocidos *WINDOW_SIZE_SECS* y *OVERLAPPING_SECS*. En particular, cada nota tendrá una duración asignada de *WINDOW_SIZE_SECS - OVERLAPPING_SECS* segundos salvo la última, que será de *WINDOW_SIZE_SECS* segundos.

Si, por el contrario, el conjunto de duraciones no es vacío, entonces se supone que a cada nota le corresponde una duración determinada bajo el mismo índice, por lo que la agrupación de notas iguales consecutivas debe agregar los tiempos asignados a cada una de ellas mediante la suma.

El resultado de esta etapa consiste en un nuevo conjunto de notas, reducido tras la agrupación, y un conjunto de duraciones asociadas a cada una de ellas.

Etapas 2.- Cálculo de duraciones en negras

Puesto que será obligatorio dar las duraciones en negras para formalizar el paso a partitura, habremos de tomar el conjunto de duraciones y calcular sus equivalentes en negras.

Sin embargo, resulta interesante cuantizar, musicalmente hablando, en caso de conocer el valor temporal más pequeño de que hará uso la melodía que se pretende transformar. En definitiva, se trata de redondear los tiempos a dicho valor temporal mínimo con el objetivo de garantizar una mayor limpieza de la partitura a costa de abandonar una exactitud, junto con los errores que pueda acarrear que, según se asume de hecho, no interesa consumir.

3. Implementación del conversor WAV-MIDI

Por tanto, sea s valor de subdivisión tal que 1 implica que la unidad mínima es la negra, 2 representa a la corchea (una negra equivale a dos corcheas), 4 la semicorchea (una negra equivale a cuatro semicorcheas), etc. Para cada duración t , se cuantiza y evalúa en negras mediante la fórmula

$$\frac{\text{redon}(s \frac{t}{\text{QUARTER_SECS}})}{s} = \frac{\lfloor s \frac{t}{\text{QUARTER_SECS}} + 0.5 \rfloor}{s}$$

En lo sucesivo, se tomará $s = \text{QUANTIZATION_FACTOR}$ parámetro de entrada al programa.

Etapas 3.- Eliminación de duraciones nulas

Tras la cuantización, las duraciones lo suficientemente pequeñas serán aproximadas por cero negras, lo que indica que, para el paso a partitura de la melodía en proceso, resultan despreciables. Así, eliminamos tanto del conjunto de notas, como de los de duraciones en segundos y en negras todos aquellos elementos cuyo índice apunte a un cero en este último conjunto.

Sin embargo, se propone introducir un valor booleano en esta funcionalidad con el objetivo de manejar la eliminación o no de silencios, pese a su brevedad. La intención en esto alude a poder utilizar silencios, por cortos que sean, como separadores entre ataques de notas iguales.

Proceso iterativo

El algoritmo completo plantea iterar indefinidamente estas tres etapas hasta que se produzca la estabilización del conjunto de notas pues, a causa de la cuantización y la eliminación de duraciones nulas, podrían ocasionarse nuevas secuencias consecutivas de notas iguales. Como se hizo mención, a lo largo de estas iteraciones, no se permitirá cuantizar silencios con el fin de separar ataques sucesivos de notas iguales pero, una vez alcanzada la estabilización (que está garantizada por la definición del método), se procede a realizar la última eliminación de ceros que, ya sí, abre paso al borrado de silencios con el fin, ya mencionado, de separar ataques consecutivos de notas iguales.

El resultado de esta fase de la conversión es el conjunto de notas a representar y sus duraciones correspondientes, tanto en segundos (dato que, realmente, no utilizaremos más) como en negras.

Para el ejemplo que se venía mostrando en el apartado de detección, cuantizando a la corchea ($\text{QUANTIZATION_FACTOR} = 2$), obtendríamos la lista de notas que se presenta a continuación:

[C4, D4, E4, F4, G4, A4, B4, C5, B4, A4, G4, F4, E4, D4, C4]

que viene a representar, simple y llanamente, la escala de *DoM* recorrida en sentido ascendente y luego descendente.

A esta secuencia corresponden las duraciones que se adjuntan a continuación, primero los tiempos reales en segundos:

[0.85, 0.9, 0.85, 0.9, 0.85, 0.8, 0.95, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.95, 1.2]

y luego en negras, tras cuantización, para un tempo de negra igual a 60 PPM:

[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]

3.3.6. Paso a partitura

Con los ingredientes necesarios para escribir una partitura, o un archivo *MIDI* equivalentemente (ahóndese en materia del formato *MIDI* en [IM91] y [MID23]), procedemos a apoyarnos en una librería de Python que haga realidad la escritura musical de la melodía procesada. Sin más preámbulos, nos beneficiaremos de *music21* (documentación en [MSAC23]), una herramienta ideada en planteamiento para la computación de estadísticas y métricas interesantes al ámbito de la musicología. Sin embargo, para la empresa que nos ocupa, se precisa únicamente del soporte de escritura sobre pentagrama y la exportación a *MIDI*, así como la impresión de partituras por pantalla.

Definido el compás como un par extra de variables de entrada, numerador y denominador, procedemos a la creación de un objeto *Score*, al que indicamos la métrica del compás, y dos objetos *Part* y *Voice*.

A continuación, se introducen, uno a uno, las notas y los silencios ordenados en el objeto *Voice*, al tiempo que se guarda control sobre la clave actual para, en el momento en que las alturas pasen a ser superiores a Do_4 , se introduzca un cambio a clave de Sol mientras que, durante el tiempo en que sean inferiores, se disponga una clave de Fa.

Finalmente, puesto que *music21* no autocompleta el último compás con silencios, este proceso se realiza manualmente de manera sencilla.

Por último, se añade el objeto *Voice* al objeto *Part* y este, a su vez, al objeto *Score*, lo que pone fin a la confección de la partitura.

Con este último elemento, *Score*, tenemos potestad, bien para imprimir la partitura obrada, bien para su exportación al formato *MIDI*, tal y como se quería.

Así, para el ejemplo que se acarrea desde el inicio del presente capítulo, la partitura correspondiente calculada por el programa de conversión es la que sigue, para un compás de 4/4:

Do M

WAV2MIDIConverter



Figura 3.27.: Escala de *DoM* en sentido ascendente y descendente.

3.3.7. Otros ejemplos de ejecución

Veamos ahora algunos ejemplos de ejecución extra.

Himno a la alegría

Para un tamaño de ventana de 0.1 segundos, un solapamiento de 0.05 segundos, un umbral de silencio (*SILENCE_THRESHOLD*) de 0.0001, una corrección que tiene en consideración un total de ocho ventanas hacia el pasado, un tempo de negra igual a 80, cuantizando a la negra, y en compás de 4/4, la grabación manual en piano de pared, mediante un dispositivo móvil, en una habitación relativamente pequeña, del tema clásico conocido como el *Himno a la alegría* queda retratado como sigue:

Himno a la alegría

WAV2MIDIConverter



Figura 3.28.: Himno a la alegría.

Himno de España

Con un tamaño de ventana de 0.2 segundos, un solapamiento de 0.15 segundos, el mismo umbral de silencio, una corrección que emplea cuatro ventanas, tempo de negra igual a 80, cuantizado a la semicorchea y compás de 4/4, la grabación de guitarra exportada de *Garage-Band* del *Himno de España* es convertida en la partitura que se muestra a continuación:

Himno de España

WAV2MIDConverter



Figura 3.29.: Himno de España.

Marcha imperial

Para un tamaño de ventana de 0.1 segundos, un solapamiento de 0.05 segundos, idéntico umbral de silencio, una corrección que emplea cinco ventanas, tempo de negra igual a 65, cuantizado a la semicorchea y compás de 2/4, la grabación manual en piano de pared, mediante un dispositivo móvil, en una habitación relativamente pequeña, de la *Marcha imperial* queda transformada en la partitura siguiente:

Marcha imperial

WAV2MIDConverter



Figura 3.30.: Marcha imperial.

3.3.8. Eficiencia

Para finalizar, se propone el estudio empírico de la eficiencia del algoritmo de conversión, dada la extensión y complejidad del algoritmo al completo.

Asimismo, puesto que lo más razonable es expresar el tiempo transcurrido (elapsed time) durante la ejecución del algoritmo en función del número de ventanas procesadas, y teniendo en mente la estrecha relación, claramente lineal, existente entre el número total de ventanas utilizado y la frecuencia fundamental de la ventana, definida en base a su inversa, que representa la duración en segundos de una ventana, se ha decidido representar el tiem-

3. Implementación del conversor WAV-MIDI

po transcurrido en función de dicha frecuencia fundamental prefijada, que se calcula como $\frac{1}{\text{WINDOW_SIZE_SECS}}$.

El resto de parámetros de entrada quedan fijos a los valores previamente descritos, salvo por aquellos que verdaderamente contribuyen a modificar el número de ventanas a analizar, como son, evidentemente:

- **La duración del archivo de audio:** Se utilizará un mismo archivo de audio a lo largo de la experiencia, que no será otro que *DoM-piano.wav*, de unos 14 segundos de duración.
- **El tamaño del solapamiento entre ventanas:** Se fijará, para cada fundamental de ventana, a la mitad de la duración de dichas ventanas.

Tomadas estas hipótesis iniciales, obtenemos la gráfica siguiente:

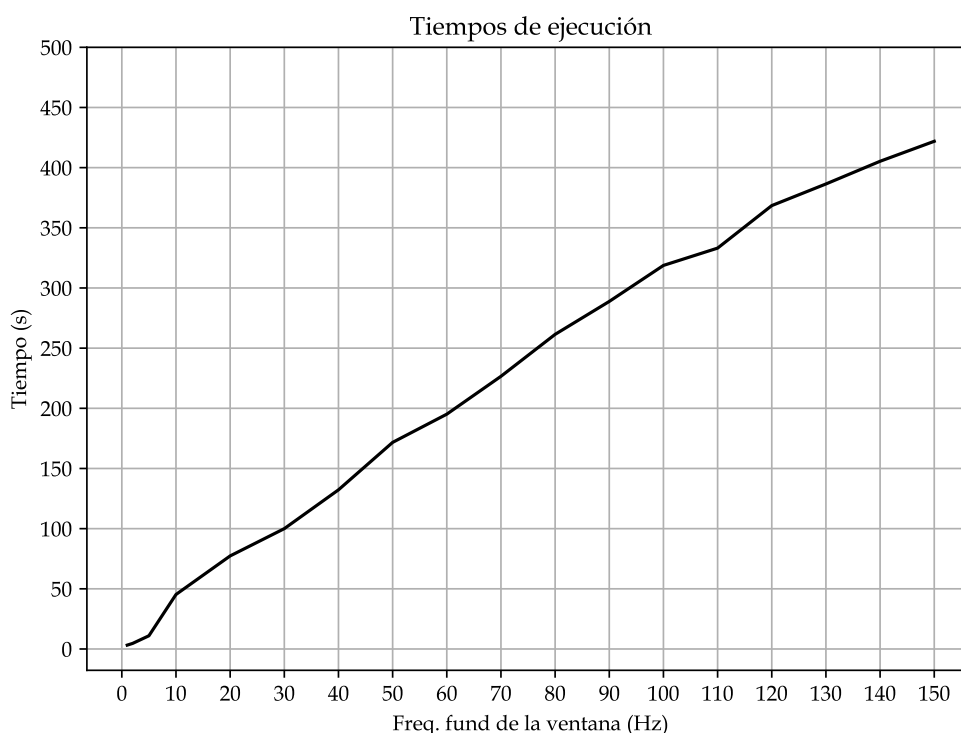


Figura 3.31.: Tiempos de ejecución en función de la fundamental de la ventana, en base a las suposiciones previas.

Lógicamente, las magnitudes variarán de formas inimaginables en base a diferentes configuraciones de los parámetros. Sin embargo, lo que realmente nos interesa de la gráfica es su forma aproximadamente **lineal**, lo que es indicador que nos encontramos ante un algoritmo aparentemente $O(f)$ con f la frecuencia fundamental de la ventana e, igualmente por tanto,

$O(n)$ en el número de ventanas procesadas por el algoritmo, un desempeño altamente aceptable, sobre todo si se tiene en cuenta la complejidad global del algoritmo de que se dispone entre manos.

Observación 3.1. Nótese, en efecto, que si el solapamiento es nulo (por simplificar), y si f y n representan la fundamental de la ventana y el número de ventanas totales, respectivamente, entonces para un fichero de duración t segundos, n se calcula como sigue:

$$n = \frac{t}{1/f} = ft$$

siendo lineal la relación entre ambas variables.

3.3.9. Estado del arte en la detección de tonos

El algoritmo de detección de tonos desarrollado en el presente trabajo halla un desempeño supeditado, en parte, al temperamento igual, en aras de generar una partitura en el marco de la práctica musical común. Algorítmicamente, la propuesta de un conjunto al que redondear los valores de frecuencia posibilita un tratamiento determinista del error derivado de la predicción.

No obstante, muchas de las herramientas descritas son comunes a implementaciones generales de la detección que, como ya se introdujo, se trata de un problema de alta complejidad y para el cual no se dispone de métodos de resolución perfectos. De este modo, se propone una breve sección final en la que debatir acerca de las bases que fundamentan algoritmos como los de programas como *Melodyne* y *Shazam*.

Se distinguen, usualmente, métodos de detección en el **dominio del tiempo** y en el **dominio de la frecuencia**, siempre dentro del ambiente digital discreto, que precisamente supone la principal fuente de error. En el primer caso, se realiza un proceso de ventaneo, se procesan las ventanas resultantes vía filtros y, finalmente, se aplica un algoritmo cuyo objetivo es localizar muestras periódicas para así obtener el período asociado a la frecuencia fundamental real. En el segundo, volvemos a operar con ventanas pero, esta vez, se pretende trabajar con frecuencias (FFT) con objeto de dilucidar la frecuencia fundamental real.

Asimismo, dos de las metodologías más utilizadas actualmente residen en el grupo de trabajo en el dominio de la frecuencia, y son la autocorrelación y el algoritmo YIN.

La primera, ya conocida, se emplea vía la densidad espectral de potencia, como se ejercita aquí, o bien esperando, idealmente, que se produzcan máximos en un retardo igual al período asociado a la fundamental y en sus múltiplos, de forma que tan solo baste con calcular el primer máximo. Ni que decir tiene que el procedimiento se embarra en condiciones normales, en las que el ruido, los armónicos y otros factores afectan a la forma de onda de manera flagrante. Como apunte, uno de los métodos clásicos para la mayor robustez de este procedimiento es el *center-clipping*, que trabaja ejecutando autocorrelación sobre una transformación de la señal que, dado $C \in \mathbb{R}^+$, anula las intensidades comprendidas entre $-C$ y C , rebaja en C las superiores a C y aumenta en C las inferiores a $-C$. Esto es, dada una señal digital $f : \{0, 1, \dots, N-1\} \rightarrow \mathbb{R}$, con $N \in \mathbb{N}$, se define una nueva señal

3. Implementación del conversor WAV-MIDI

$g : \{0, 1, \dots, N-1\} \rightarrow \mathbb{R}$ donde

$$g(k) = \begin{cases} f(k) - C & \text{si } f(k) > C \\ 0 & \text{si } |f(k)| \leq C \\ f(k) + C & \text{si } f(k) < -C \end{cases} \quad \forall k \in \{0, 1, \dots, N-1\}$$

Por otro lado, el algoritmo YIN comporta un importante paralelismo con la propia autocorrelación. Como sabemos,

$$r_f(n) = \sum_{k=0}^{N-1} f(k)f(k+n) \quad \forall n \in \{0, 1, \dots, N-1\}$$

representa la autocorrelación de f . Pues bien,

Definición 3.6. Diremos *Average Magnitude Difference Function* o **AMDF** de f a la señal

$$d(n) = \frac{1}{N} \sum_{k=0}^{N-1} |f(k) - f(k+n)| \quad \forall n \in \{0, 1, \dots, N-1\}$$

Inspirada en la AMDF que, en contrapunto a lo que sucedía con la autocorrelación, alcanza valores mínimos para traslaciones de correlación máxima, surge la función del algoritmo YIN por suma de cuadrados, de propiedades similares a la anterior:

$$d(n) = \frac{1}{N} \sum_{k=0}^{N-1} (f(k) - f(k+n))^2 \quad \forall n \in \{0, 1, \dots, N-1\}$$

A esta se le aplica un proceso de normalización acumulativa con el fin de reducir los efectos del posible cambio de amplitud de la señal a lo largo del tiempo, dando lugar a

$$d'(n) = \begin{cases} 1 & \text{si } n = 0 \\ \frac{d(n)}{\frac{1}{n} \sum_{k=0}^n d(k)} & \text{si } n \in \{1, 2, \dots, N-1\} \end{cases}$$

A continuación, en un enfoque similar al del algoritmo de conversión propuesto, se calcularía un umbral bajo el cual los mínimos pasarían a considerarse candidatos a período de la frecuencia fundamental.

Este tipo de técnicas se utilizan en programas de detección punteros como *Melodyne*. Por su parte, *Shazam* no implementa una detección de tonos como tal sino que, habiendo almacenado de antemano un conjunto inmenso de espectrogramas de canciones populares, define métricas de similitud entre espectrogramas para estudiar las soluciones de un problema de clasificación dado un espectrograma (correspondiente a un fichero de audio) nuevo.

Profundícese al respecto de estas y más técnicas en [AA21], [Kuh90], [Muh10], [BG] y [GW].

4. Transformaciones digitales

Hasta el momento se ha discutido profusamente lo concerniente al análisis de señales de audio y, en especial, a lo que al cálculo de frecuencias respecta. Es por ello que conviene, finalmente, debatir la aplicación del modelo teórico propuesto en el fundamento amén de la síntesis de nuevas señales a partir de otras dadas lo que, en definitiva, puede considerarse transformación digital.

4.1. Filtrado de señales digitales

Recuperemos primeramente el marco de trabajo utilizado durante el estudio de la discretización de señales:

Sea una señal T -periódica $f : \mathbb{R} \rightarrow \mathbb{C}$, con $T > 0$, y tal que $f|_{[0,T]} \in L_2([0,T])$, de la que se conocen un número finito $N \in \mathbb{N}$ de evaluaciones, para cada una de las abscisas que conforman la partición equiespaciada $P_N = \{0 < \frac{T}{N} < \frac{2T}{N} < \dots < T\}$ del intervalo $[0, T]$. Sea, por alusión, $\{f_k^N : k \in \{1, 2, \dots, N\}\}$ el conjunto de evaluaciones.

Observación 4.1. Desde el principio, el contexto impele a la concepción de T , en tanto que responde a una magnitud temporal, como una cifra en segundos, por ejemplo. Sin embargo, pudiera en ocasiones ser clarificador tomar $T = N$ el número de muestras como nueva medida de tiempo, conocida de antemano la frecuencia de muestreo, por supuesto, de donde entonces se tendría $P_N = \{0 < 1 < 2 < \dots < N\}$. Dado que en lo sucesivo las abscisas no obtendrán mención en el asunto, entiéndase como un mero apunte a bien de manifestarse en la literatura.

Recordemos que el objeto principal de la discretización, y su producto, como se ha probado, resulta el cálculo de unos coeficientes de Fourier aproximados vía operaciones sencillas sobre las evaluaciones f_k^N y, venturosamente, la recuperación de dichas evaluaciones previo conocimiento de los coeficientes aproximados por medio, como es natural, de la aplicación del proceso inverso. Por supuesto, la utilidad conocida de dichos coeficientes aproximados, que de ahora en adelante notaremos como $\widehat{f}_k^N := \widehat{f}_N(k-1) \quad \forall k \in \{1, 2, \dots, N\}$, radica en servir a modo de medida de intensidad, amplitud o notabilidad de la frecuencia $\frac{k-1}{T}$ de acuerdo a las conclusiones extraídas en capítulos precedentes.

Pertinentemente, cabe recopilar la notación utilizada para referir a los vectores de \mathbb{C}^N que representan, por un lado a las evaluaciones y, por otro, a las aproximaciones:

$$F_N = (f_1^N, f_2^N, \dots, f_N^N)^T, \quad \widehat{F}_N = (\widehat{f}_1^N, \widehat{f}_2^N, \dots, \widehat{f}_N^N)^T$$

así como a la denominación empleada para invocar a las matrices de análisis, A_N , y de síntesis, A_N^{-1} , que verificaban:

$$\widehat{F}_N = A_N F_N, \quad F_N = A_N^{-1} \widehat{F}_N$$

4. Transformaciones digitales

Consúltense acerca de la señal digital en [Smig99].

Ya nos encontramos en condiciones para presentar transformaciones que operan inmediatamente sobre los coeficientes aproximados para, a través de su modificación y artesanía, gestar determinados efectos en la señal f de partida.

Definición 4.1. Llamaremos **filtro** a una transformación $T : \mathbb{C}^N \longrightarrow \mathbb{C}^M$, con $N, M \in \mathbb{N}$.

Un filtro, de manera general sobre las hipótesis en que nos movemos, es toda correspondencia arbitraria entre dos espacios en los que habitan configuraciones de espectros de frecuencia (listas de coeficientes aproximados) y su modo de empleo, como se presta a intuir, es el siguiente:

1. Tomamos la señal original f dada por sus evaluaciones F_N .
2. Aplicamos la matriz de análisis, A_N , para obtener \widehat{F}_N :

$$\widehat{F}_N = A_N F_N$$

3. Transformamos los coeficientes aproximados mediante el filtro, calculando así el espectro de frecuencias de la señal modificada f^* :

$$\widehat{F}_M^* := T(\widehat{F}_N)$$

4. Extraemos la señal modificada, dada por el vector de evaluaciones F_M^* aplicando la matriz de síntesis apropiada, A_M^{-1} sobre sus coeficientes aproximados:

$$F_M^* = A_M^{-1} \widehat{F}_M^*$$

En definitiva, se tiene que

$$F_M^* = A_M^{-1} T(A_N F_N)$$

Observación 4.2. Adviértanse los siguientes hechos en lo concerniente al filtro:

- Que N y M pueden ser iguales o distintos, luego se podrá jugar con espacios vectoriales de diferente dimensión.
- Que, independientemente de si $N = M$ o no, la transformación bien podría no ser biyectiva y, en consecuencia, no invertible.
- Que, por supuesto, T no tiene por qué ser lineal aunque, de serlo, por trabajar en espacios lineales de dimensión finita, T sería equivalente a multiplicar por una matriz en $\mathcal{M}_{M \times N}(\mathbb{C})$.

Observación 4.3. Nada nos impide utilizar cualquier tipo de matriz de análisis A_N con tal de que cumpla con su función de acuerdo a la base teórica presentada. En particular, la matriz de análisis B_N asociada a la FFT es perfectamente intercambiable con A_N en el proceso de transformación expuesto.

Veamos ahora una serie de ejemplos de filtros clásicos en el procesamiento de señales digitales.

Como caso práctico de aplicación, dispondremos del archivo de ejemplo utilizado para ilustrar la precisión del conversor que, recordemos, se trataba de una escala de *DoM*, ascendente y descendente, interpretada en piano.

La visualización de frecuencia en función del tiempo no encaja con ninguna de las tipologías de gráfico previamente mostradas: el osciloscopio (tiempo \rightarrow intensidad) y el análisis espectral (frecuencia \rightarrow intensidad). Introducimos, en consecuencia, un nuevo formato de visualización de archivos de audio más adecuado en lo sucesivo para la materia que nos ocupa: el **espectrograma** ([Car97], [SRU]).

Un espectrograma se trata, como se adelantaba, de una representación del contenido en frecuencias de una señal en función del tiempo. Se dispone, por tanto, de la estructura de frecuencias que componen dicha señal en cada instante. La forma de presentar el contenido en frecuencias en un gráfico bidimensional requiere de una variable más, la intensidad, a incluir mediante el color o el brillo de los puntos del espectrograma, siendo que a mayor brillo, mayor intensidad de la frecuencia correspondiente en el instante dado.

Véase, a continuación, el espectrograma correspondiente al archivo de ejemplo, *DoM-piano.wav*:

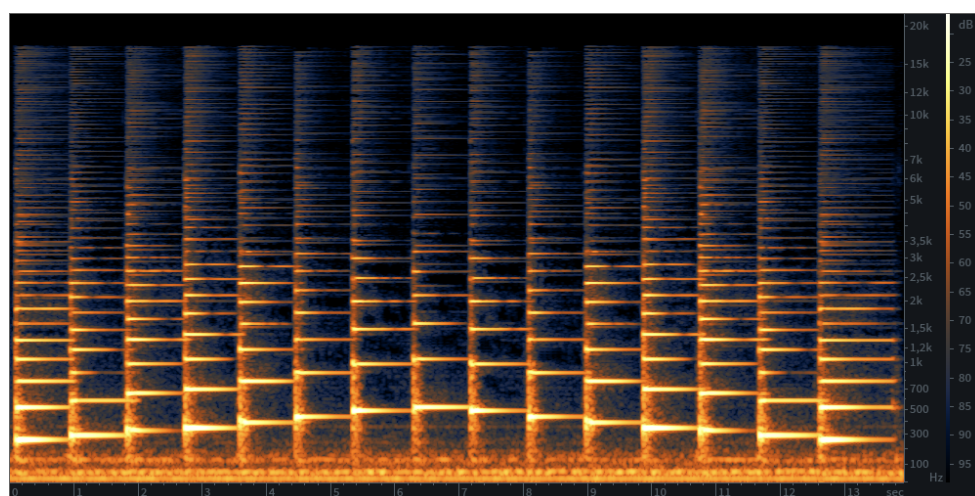


Figura 4.1.: Espectrograma del archivo *DoM-piano.wav* exportado con *iZotope RX8*.¹

¹Nótese, de acuerdo a la explicación aportada, que las columnas representan notas de la escala de *DoM*, cuyas frecuencias fundamentales y armónicos quedan retratados como segmentos horizontales a lo largo de su duración en segundos.

A la derecha, se incluye el eje vertical en Hz y en escala logarítmica (tal y como el oído los trata), detalle a percibir dada la mayor compresión de los segmentos a medida que se asciende en la gráfica.

Finalmente, y también a la derecha, se adjunta la leyenda para la variable intensidad, en decibelios.

4. Transformaciones digitales

4.1.1. Ecualizador

Un ecualizador se trata de una transformación que, para cada frecuencia de manera independiente, asocia una nueva amplitud ya sea amplificada o reducida respecto a la amplitud original, sin alterar la dimensión de la señal de partida, luego $T : \mathbb{C}^N \rightarrow \mathbb{C}^N$.

Puede caracterizarse, por tanto, como una lista de factores complejos $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_N) \in \mathbb{R}^N$ asociados a cada frecuencia (que, de hecho, definen una poligonal o una curva ajustable comúnmente por el usuario en programas de edición de audio), siendo así

$$T(F_N) := (\alpha_1 f_1^N, \alpha_2 f_2^N, \dots, \alpha_N f_N^N)$$

Al ser T lineal, se identifica con una matriz en $\mathcal{M}_N(\mathbb{C})$. Concretamente, con la matriz

$$\begin{pmatrix} \alpha_1 & 0 & \dots & 0 \\ 0 & \alpha_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \alpha_N \end{pmatrix}$$

En materia de edición digital de audio, los ecualizadores son muy comúnmente empleados con el fin de realizar ajustes en calidad y timbre del sonido, ya sea en clave creativa o de restauración/simulación de ambientes sonoros diferentes, en base a la respuesta que cada frecuencia ofrece tan solo por emitirse en un medio particular. Por ejemplo, se utiliza para la eliminación de respiraciones de cantantes o cualquier tipo de *popping* en las frecuencias graves, o de resonancias concretas de un instrumento musical, pero también posibilita resaltar frecuencias interesantes a juicio de la composición musical.

Indudablemente, la identificación del filtro como una función real que a cada frecuencia asigna un factor o peso multiplicativo eleva las posibilidades y los casos de uso. Así, surgen una serie de transformaciones derivadas del ecualizador que resulta preciso presentar, sea por su habitualmente considerable frecuencia de empleo: los filtros de paso bajo, paso alto, paso banda y banda rechazada o eliminada.

4.1.2. Filtro de paso bajo

El filtro de paso bajo respeta las amplitudes de las frecuencias más graves y actúa atenuando las frecuencias agudas. Para ello, define un umbral a partir del cual la nueva amplitud asociada será cero. Por tanto, existirá un cierto $m \in \{1, 2, \dots, N-1\}$ tal que $\alpha_n = 0 \ \forall n > m$, mientras que $\alpha_n \neq 0 \ \forall n \leq m$.

Teóricamente, el filtro se escribirá como

$$T(F_N) := (\alpha_1 f_1^N, \alpha_2 f_2^N, \dots, \alpha_m f_m^N, 0, \dots, 0)$$

Este subtipo de ecualización encuentra utilidad en el equilibrado de una mezcla musical al evitar determinados choques sonoros que puedan producirse con frecuencias agudas, la eliminación de ruido y la simulación de distancia a la fuente sonora o la absorción de mate-

riales y entornos para la imitación de paisajes sonoros.

También se conoce como filtro *antialiasing* por su contribución a mitigar el *efecto moiré* en fotografía, y reduce interferencias con la disminución del ancho de banda en señales de comunicación. Contamos igualmente con el interés en la reducción de ruido de señales biomédicas y de radiofrecuencia.

Con ánimo de ilustrar tanto el funcionamiento de un filtro de paso bajo ideal como el desempeño de la implementación de filtros incluida en el anexo, se adjunta a continuación el espectrograma del resultado de aplicación de un filtro paso bajo (*Low-Pass Filter* o *LPF*, en inglés) sobre el archivo *DoM-piano.wav*, cuyos coeficientes toman valores en el conjunto $\{0, 1\}$, establecido el umbral a 400 Hz:

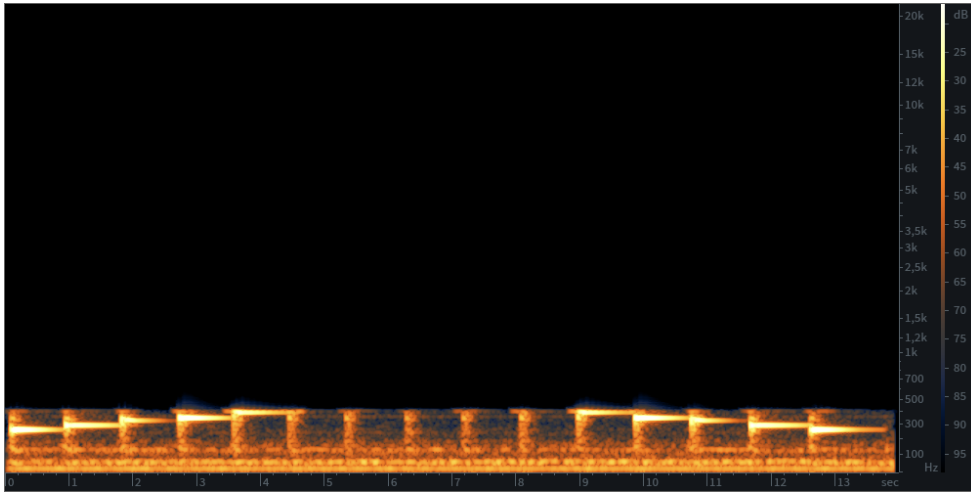


Figura 4.2.: Espectrograma del archivo *DoM-pianoLPE.wav* exportado con *iZotope RX8*.²

4.1.3. Filtro de paso alto

En contraposición al filtrado de paso bajo, el paso alto plantea la anulación de las amplitudes de frecuencias graves, de forma que se dejen pasar tan solo las frecuencias agudas de una señal. El marco teórico define la transformación por medio de un umbral, al igual que sucedía para el filtro anterior, existiendo un cierto $m \in \{2, 3, \dots, N\}$ tal que $\alpha_n = 0 \quad \forall n < m$, mientras que $\alpha_n \neq 0 \quad \forall n \geq m$.

Y así,

$$T(F_N) := (0, \dots, 0, \alpha_m f_m^N, \alpha_{m+1} f_{m+1}^N, \dots, \alpha_N f_N^N)$$

Entre los usos relevantes y particulares de este filtro destacan la eliminación de zumbidos y ruidos de baja frecuencia (en la red eléctrica, o el efecto del viento en micrófonos, por ejemplo), la mejora en la claridad del sonido en música, la separación de frecuencias para su envío a los correspondientes altavoces (agudos a *tweeters* y graves a *woofers*), el resaltado

²Adviértase el vacío generado sobre las frecuencias agudas eliminadas en la sección superior del espectrograma.

4. Transformaciones digitales

de bordes y detalles finos en fotografía, y otros de carácter similar en las disciplinas de comunicaciones, radiofrecuencia y biomedicina.

Nuevamente, comprobemos el resultado de aplicar un filtro de paso alto (*High-Pass Filter* o *HPF*, en inglés) ideal, cuyos coeficientes pertenecen al conjunto $\{0, 1\}$, establecido el umbral a 400 Hz:

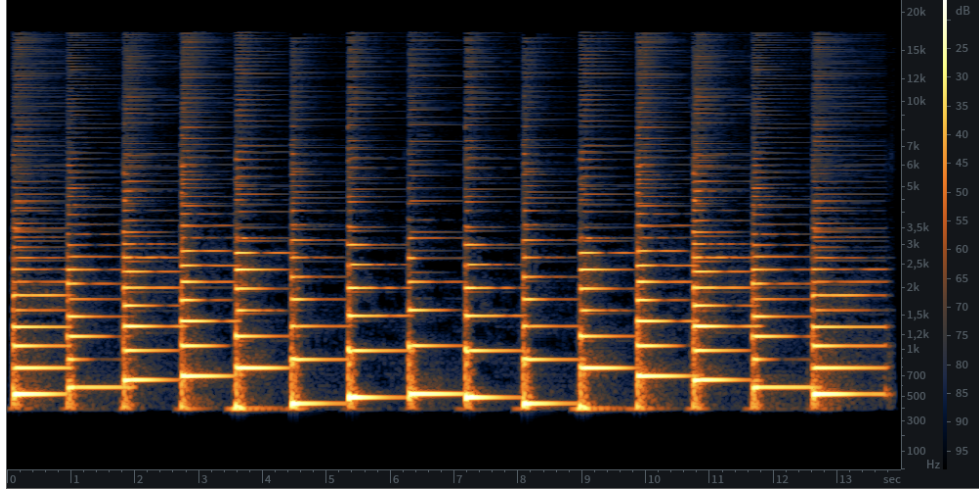


Figura 4.3.: Espectrograma del archivo *DoM-pianoHPF.wav* exportado con *iZotope RX8*.³

4.1.4. Filtro de paso banda

Nacido de la combinación de los dos anteriores, el filtro de paso banda permite la definición de un intervalo de frecuencias prevalentes, al tiempo que el resto son canceladas. Existirán, por tanto, $p, q \in \{2, 3, \dots, N-1\}$ verificando $p \leq q$ tales que $\alpha_n = 0$ si $n < p$ ó $n > q$, y $\alpha_n \neq 0$ si $p \leq n \leq q$.

Luego

$$T(F_N) := (0, \dots, 0, \alpha_p f_p^N, \alpha_{p+1} f_{p+1}^N, \dots, \alpha_q f_q^N, 0, \dots, 0)$$

Además de la elusión de ruido en todo tipo de señales, tanto de altas como de bajas frecuencias, el filtro de paso banda permite, en materia de telecomunicaciones, seleccionar una banda de frecuencias concreta con el fin de seleccionar un determinado canal de comunicación. En general, goza de las utilidades combinadas descritas anteriormente para los filtros paso bajo y paso alto.

Transformamos seguidamente el archivo de ejemplo por medio de un filtro de paso banda (*Band-Pass Filter* o *BPF*, en inglés) ideal, cuyos coeficientes pertenecen al conjunto $\{0, 1\}$, tomando el primero de los umbrales en 400 Hz y el segundo, en 600 Hz:

³Adviértase el vacío generado sobre las frecuencias graves eliminadas en la base del espectrograma.

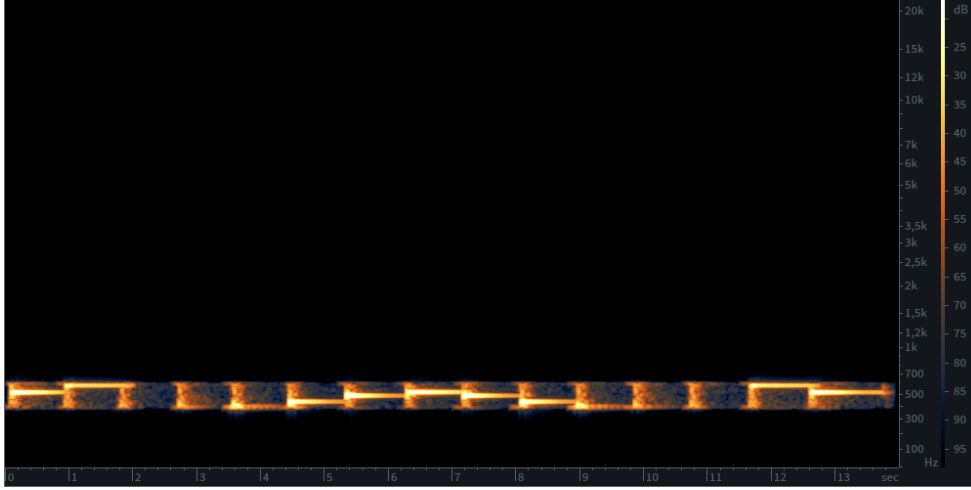


Figura 4.4.: Espectrograma del archivo *DoM-pianoBPF.wav* exportado con *iZotope RX8*.⁴

4.1.5. Filtro de banda rechazada o eliminada

Finalmente, se encuentra el complementario del filtro de paso banda, que propone un par de umbrales, existiendo $p, q \in \{1, 2, 3, \dots, N\}$ con $p < q$ tales que $\alpha_n = 0$ si $p < n < q$, y $\alpha_n \neq 0$ si $n \leq p$ ó $n \geq q$.

Consecuentemente,

$$T(F_N) := (\alpha_1 f_1^N, \dots, \alpha_p f_p^N, 0, \dots, 0, \alpha_q f_q^N, \dots, \alpha_N f_N^N)$$

Posibilita la eliminación de interferencias (como el zumbido en la red eléctrica) y resonancias específicas en sistemas de sonido.

Por último, ejemplificamos el fruto de la aplicación de un filtro de banda rechazada o eliminada (*Band-Stop Filter* o *BSF*, en inglés) ideal sobre el archivo de ejemplo, cuyos coeficientes se encuentran, de nuevo, en $\{0, 1\}$, eligiendo idénticos umbrales que en el caso anterior, a los 400 Hz y a los 600 Hz:

⁴Se desecha todo el contenido de frecuencias ajeno a la banda entre los 400 y los 600 Hz.

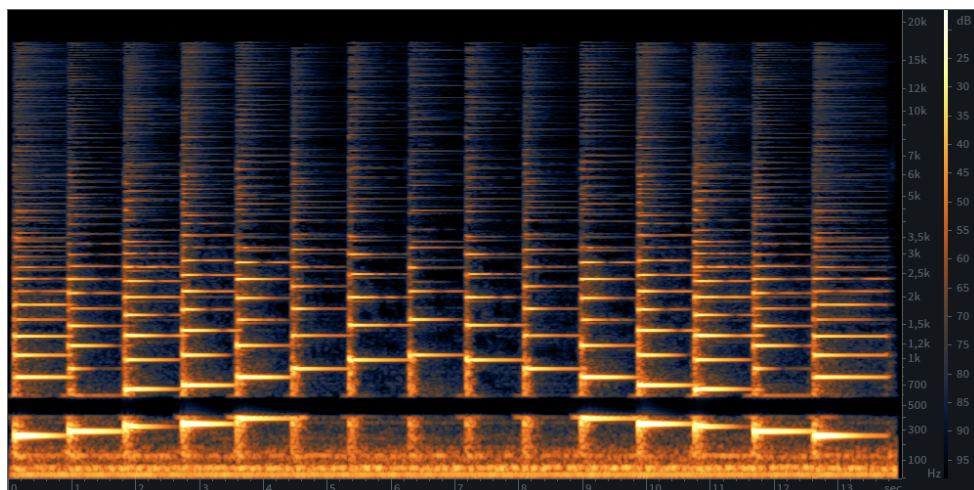


Figura 4.5.: Espectrograma del archivo *DoM-pianoBSF.wav* exportado con *iZotope RX8*.⁵

4.1.6. Implementación de un filtro

Cabe asumir, por supuesto, que el modo de empleo de filtros establecido en la primera de las secciones del presente capítulo refiere a un tratamiento general de señales que, a priori, nada tiene que ver con el ventaneo con solapamiento utilizado, por ejemplo, para el mecanismo de conversión de *WAV* a *MIDI* expuesto en el capítulo tercero.

Sin embargo, es a fin de cuentas crucial la técnica de ventaneo para manejar un mayor control de la resolución en frecuencia y tiempo, como ya se ha explicado, pero igualmente es necesario para garantizar una mejor representación de señales que no son periódicas (recordemos que tratamos, desde el primer momento, de adaptar la teoría de funciones periódicas a funciones que no lo son), así como la menor introducción de artefactos indeseables en la señal de partida.

Por tanto, parece lógico pensar que se debe dar cuenta del modo de implementación de un filtro, que ya realiza un proceso de síntesis, y su conjunción con el ventaneo y el hanning, ya conocidos.

Para el conversor, la aplicación de la ventana de hanning antes de la síntesis permitía dilatar el perfil de la curva de análisis espectral con el fin de una mejor detección de picos, en contra de la cual jugaba la magnitud de la frecuencia fundamental de la ventana.

Ahora no se pretende la detección de picos, ni razonar sobre la forma de la FFT, sino trabajarla en bruto, motivo por el cual no se aplicará hanning antes del proceso de síntesis. En su lugar, trabajaremos con la función de hanning para conformar una **partición de la unidad** amén de asegurar una agregación pertinente de las ventanas en la síntesis.

Téngase en cuenta que

⁵Se anulan las intensidades de las frecuencias localizadas en la banda 400-600 Hz.

Definición 4.2. Sea X un espacio topológico. Diremos que una familia \mathfrak{F} de funciones con dominio X e imagen contenida en $[0, 1]$ es una **partición de la unidad** si satisface las dos propiedades siguientes:

- Para cada $x \in X$, existe $U_x \subset X$ entorno de x en el que se anulan todas salvo un número finito de funciones en \mathfrak{F} .
- Para cada $x \in X$, se verifica que $\sum_{f \in \mathfrak{F}} f(x) = 1$.

Con esto en mente, es posible construir una partición de la unidad con funciones de la forma de la función de hanning utilizada para la conversión. Definida sobre el intervalo $[0, T]$, para $T > 0$,

$$h(x) = \frac{1}{2} (1 - \cos(\frac{2\pi}{T}x)) \quad \forall x \in [0, T]$$

Pues bien, es sencillo comprobar que la familia

$$\mathfrak{H}_T = \{h_T^k : k \in \mathbb{Z}\}$$

donde

$$\begin{aligned} h_T^k(x) &= \begin{cases} \frac{1}{2} (1 - \cos(\frac{2\pi}{T}(x - \frac{kT}{2}))) & \text{si } x \in [\frac{kT}{2}, \frac{(k+2)T}{2}] \\ 0 & \text{si } x \notin [\frac{kT}{2}, \frac{(k+2)T}{2}] \end{cases} \\ &= \begin{cases} \frac{1}{2} (1 - \cos(\frac{2\pi}{T}x - \pi k)) & \text{si } x \in [\frac{kT}{2}, \frac{(k+2)T}{2}] \\ 0 & \text{si } x \notin [\frac{kT}{2}, \frac{(k+2)T}{2}] \end{cases} \quad \forall k \in \mathbb{Z} \end{aligned}$$

es una partición de la unidad ⁶.

En efecto,

- h_T^k está definida en \mathbb{R} y $|h_T^k(x)| \leq \frac{1}{2} (1 - \cos(\frac{2\pi}{T}x - \pi k)) \leq \frac{1}{2} (1 + 1) = 1 \quad \forall x \in \mathbb{R}$.
- Dado $x \in \mathbb{R}$, existen, a lo sumo, tres valores de k consecutivos tales que $x \in [\frac{kT}{2}, \frac{(k+2)T}{2}]$.
- Para cada $x \in [\frac{kT}{2}, \frac{(k+2)T}{2}] \cap [\frac{(k+1)T}{2}, \frac{(k+3)T}{2}]$ se tiene

$$\begin{aligned} h_T^k(x) + h_T^{k+1}(x) &= \frac{1}{2} (1 - \cos(\frac{2\pi}{T}x - \pi k)) + \frac{1}{2} (1 - \cos(\frac{2\pi}{T}x - \pi(k+1))) \\ &= \frac{1}{2} (2 - \cos(\frac{2\pi}{T}x - \pi k) - \cos(\frac{2\pi}{T}x - \pi k - \pi)) \\ &= \underbrace{\frac{1}{2} (2 - \cos(y) - \cos(y - \pi))}_{y = \frac{2\pi}{T}x - \pi k} \\ &= \frac{1}{2} (2 - \cos y + \cos y) = 1 \end{aligned}$$

⁶Obsérvese que, ciertamente, la familia \mathfrak{H}_T se compone de las versiones desplazadas de h en múltiplos enteros de $\frac{T}{2}$.

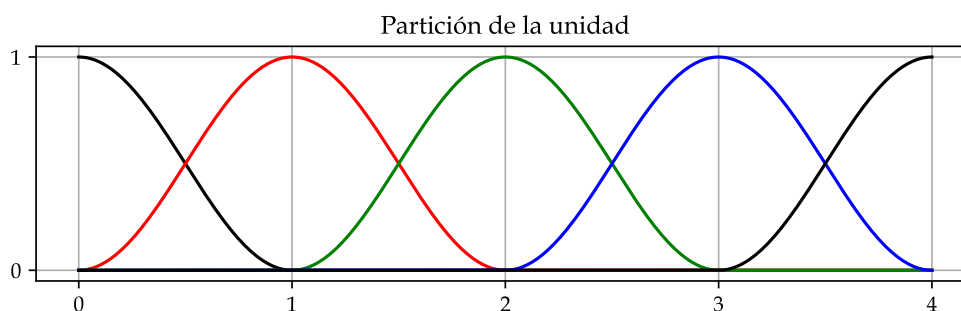


Figura 4.6.: Representación de algunos elementos de la partición de la unidad elegida, para $T = 2$.

Con ánimo de construir un filtro, esta partición de la unidad servirá a modo de combinación convexa con la que agregar los valores de las ventanas en las zonas de solapamiento; esto se detallará más adelante. No obstante, conviene asimilar que, consideradas ventanas de tamaño T , con objeto de aplicar la partición especificada a la manera que se propone, el solapamiento entre ventanas vendrá determinado por una duración de $\frac{T}{2}$.

Asimismo, una vez definidos el tamaño de la ventana T en segundos, por ejemplo; el filtro a aplicar, los umbrales pertinentes o cualquier otro parámetro que fuera necesario, junto con la función de hanning seleccionada, se procede como sigue:

1. Leer el archivo de audio y fragmentar en ventanas tal y como, valiéndose de modelo, se realizó en el conversor. Para cada una de las ventanas, realizar los procesos de:
 - a) Análisis: Computar la FFT de la ventana (sin ser precisa una sola modificación, pues se pretende evadir toda pérdida de datos).
 - b) Filtrado: Aplicar del filtro, que conlleva modificar las amplitudes de la FFT.
 - c) Síntesis: Transmutar la FFT filtrada en una nueva señal por medio de la transformada rápida inversa, IFFT, recuperando así el dominio del tiempo.
2. Se dispone ahora de un conjunto nuevo de ventanas que almacenan la señal transformada, pero la síntesis no cesa hasta componer la nueva señal, que requiere la integración del conjunto respetando el solapamiento.

Para ello, se toman las ventanas, una a una, se les aplica hanning (el producto por la función definida) y se procede a combinarlas en las zonas de solapamiento meramente mediante la suma, para cada muestra. Nótese que, realmente, se está realizando una media ponderada de las dos ventanas (a efectos prácticos) que se combinan en cada punto dadas las características que posee por definición una partición de la unidad (los pesos que multiplican los valores reales de la ventana suman 1).

En definitiva, este proceso de aplicación de hanning permite suavizar las transiciones de una ventana a otra para lograr un solapamiento imperceptible y, para ello, se hace uso, ciertamente, de una combinación convexa de los valores de las ventanas que se unen en cada punto.

3. Finalmente, se erige un nuevo fichero de audio utilizando como materias primas el conjunto de evaluaciones obtenido tras la síntesis y la frecuencia de muestreo del archivo original.

En relación al filtrado de señales y su utilidad práctica, profundícese en [Ele20], [Engo4], [MK11], [MM21] y [PM96].

4.2. Compresión de audio

Finalmente, cruzar el sendero del filtrado y la transformación de señales conduce, irrevocablemente, a la discusión sobre la compresión de señales de audio, procedimiento que encuentra su meta en la reducción, tanto mejor cuanto más significativa, del tamaño de los ficheros encargados de almacenar audio de naturaleza digital.

Por supuesto, se emplea en ello la herramienta clave ya más que mencionada a lo largo del presente trabajo: el análisis del espectro de frecuencias; especialmente, en la forma de la FFT. Y es que la aplicación de la matriz de análisis transforma la señal de partida, dada por F_N , en una nueva lista de valores, \widehat{F}_N , de la misma longitud a priori y, por consiguiente, del mismo tamaño en disco.

Notemos que el mero descarte de elementos en \widehat{F}_N haría, en la síntesis, que se obtuviera una señal nueva de un tamaño inferior a la original y, por tanto, no se estaría hablando de compresión en ningún caso.

Con objeto de preservar la duración de la señal de origen y comprimir simultáneamente, es preciso descartar un conjunto de frecuencias considerándolas nulas en la práctica, lo cual implica almacenar tan solo las amplitudes de las frecuencias consideradas relevantes y, previo a la recreación en la síntesis, embutir ceros en el espacio faltante hasta cumplir con la longitud inicialmente dada.

No obstante, un detalle fundamental que no puede obviarse recae en que, por razones de eficiencia y precisión, un archivo de audio habrá de manipularse por medio de ventanas con solapamiento, tal y como se realizó durante la implementación del conversor, y en base a las mismas necesidades.

Otro apunte imprescindible subyace en la elección del conjunto de frecuencias a preservar o a descartar, y el criterio se encuentra, como no podría ser de otra manera, en características psico-acústicas del oído humano; más concretamente, en el hecho de la mayor percepción de frecuencias medias frente a las graves y, especialmente, las agudas. Así, se pretende engañar al oído descartando las frecuencias que proveen, de acuerdo a su comportamiento, el "detalle" en una audición compleja.

4. Transformaciones digitales

Los pasos, en definitiva, que guían la compresión de audio estándar (MP3) de un archivo WAV se detallan a continuación:

1. Fragmentar en archivo en un conjunto de ventanas con solapamiento.
2. Aplicar, para cada una de ellas, el proceso de análisis condensado, para mayor eficiencia, en la FFT.
3. Transformar cada una de las ventanas de acuerdo a un filtro de paso banda que anule las frecuencias extremas, tanto agudas como graves, y que mantenga vivas las frecuencias medias-graves, prevalentes al oído.
4. Almacenar en un nuevo fichero la banda no anulada de los espectros de cada una de las ventanas, junto con la información necesaria en lo que respecta al filtro para garantizar su recuperación.

Observación 4.4. Dado que se propone una eliminación radical del contenido en frecuencias de cada ventana del archivo de audio, se espera, lógicamente, que el archivo recreado durante la síntesis sea distinto al original, en tanto que no se puede recuperar la información desechada; hablamos, por tanto, de una **compresión con pérdidas**. En otro caso, si la síntesis arrojara un archivo idéntico al original, se hablaría de **compresión sin pérdidas**.

Finalmente, su síntesis, dado un fichero comprimido, se realizaría de la siguiente forma:

1. Leer los espectros de cada una de las ventanas almacenadas.
2. Añadir tantos ceros, a izquierda y a derecha (*cero padding*) como sea preciso, en función del filtro aplicado durante la compresión, para cada ventana.
3. Aplicar, a cada espectro ampliado, la transformada rápida inversa, IFFT, para retornar al dominio del tiempo en el tamaño o duración original.
4. Sintetizar una nueva versión del fichero inicial unificando las ventanas obtenidas vía la IFFT respetando el solapamiento establecido.

Conviene citar, como complemento a la mención acerca de la compresión de datos y, especialmente, al formato MP3, a [Seloo], [Say18] y [Guc12]. También, en cuanto a las transformaciones digitales con base en el Análisis de Fourier, se alude a [FS22], [IG19] y [SG23].

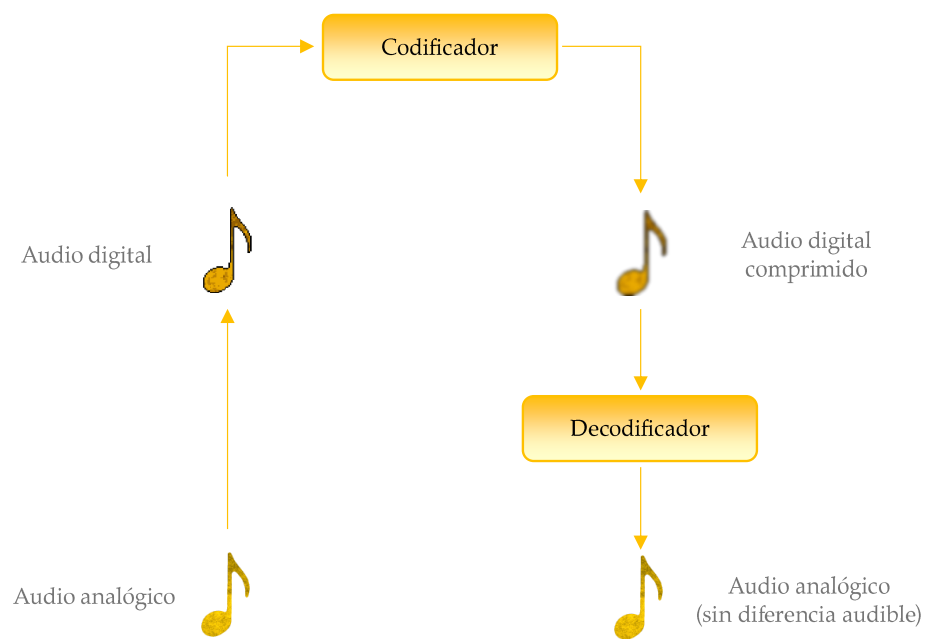


Figura 4.7.: Esquema básico del funcionamiento de un filtro *MP3* para compresión de ficheros de audio.

5. Planificación

Se propone a continuación el detalle acerca del proceso de composición del trabajo, fundamentalmente en lo concerniente a su vertiente práctica.

5.1. Marco de desarrollo

El modelo de desarrollo de software elegido en aras de implementar las aplicaciones producto del presente trabajo, donde fundamentalmente cabe reseñar el conversor, se trata del modelo en cascada con retroalimentación, un formato de actividad cuyo modus operandi viene retratado en la imagen siguiente:

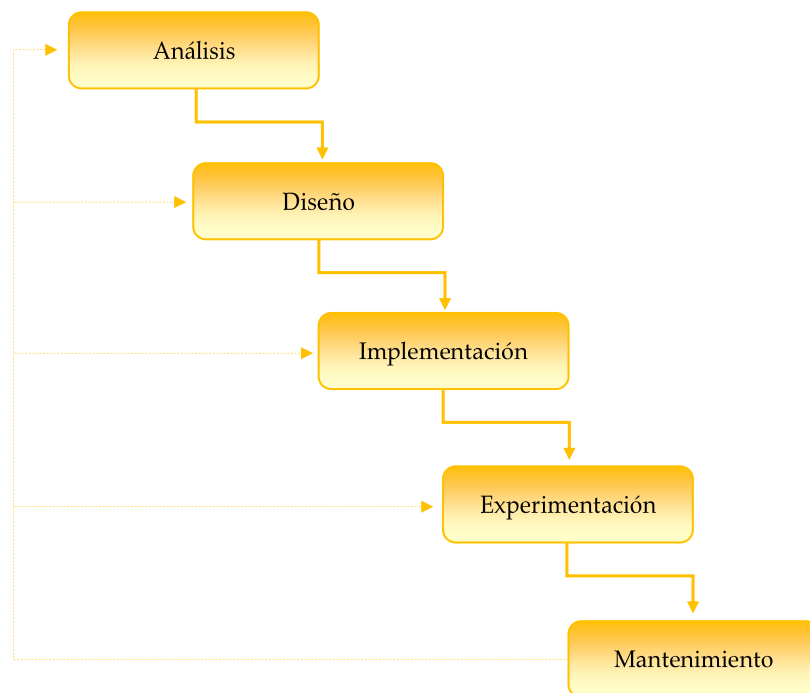


Figura 5.1.: Esquema general del modelo en cascada con retroalimentación.

Considera, como puede verse, cinco etapas bien diferenciadas verificando que, dada una etapa arbitraria, se admite descender a aquella inmediatamente inferior o bien regresar a cualquiera de las a ella superiores.

5. Planificación

A continuación, se presentan:

1. **Análisis:** Condensa la investigación requerida en el ámbito de interés, el planteamiento del problema a resolver sumado a la relevancia de su resolución, un análisis previo del estado del arte y, si cabe, revisión de soluciones existentes.
2. **Diseño:** Constitución del plan de acciones a ejecutar en pos de la solución al problema; fundamentalmente, consiste en la composición de algoritmos que aproximen a la consecución del objetivo.
3. **Implementación:** Se trata de llevar los diseños obtenidos en la etapa anterior a código ejecutable.
4. **Experimentación:** Se trata de una etapa de pruebas de ejecución del código implementado, así como de obtención de resultados. Es un momento clave, en que valorar muy concienzudamente el regreso a etapas previas del proceso de vida del software.
5. **Mantenimiento:** Una vez obtenida una primera versión completa de la solución, se procede a su mantenimiento y, si cabe, se impulsa el desarrollo volviendo atrás en el ciclo.

5.2. Programación temporal

Considerando que al proyecto se le asigna un total de 18 créditos ECTS, y que cada crédito implica 25 horas de trabajo, el número de horas requeridas para su compleción se eleva a 450, de las que la mitad, 225, corresponden con la dedicación diputada a la parte práctica al completo.

Habiéndome ocupado de ello desde enero del año actual, 2024, en total han transcurrido, aproximadamente, 22 semanas de trabajo, lo que arroja una cifra, redondeada al alza, de unas 11 horas semanales, que se consideran completamente realizadas compensadas las de unas semanas con las de otras.

Asimismo, el diagrama de Gantt que sigue ofrece una perspectiva sobre el proceso de desarrollo de software previamente descrito:

	Enero					Febrero					Marzo					Abril					Mayo					Junio		
Etap	1	8	15	22	29	5	12	19	26	4	11	18	25	1	8	15	22	29	6	13	20	27	3	10	17	24	31	
Análisis																												
Diseño																												
Implementación																												
Experimentación																												
Mantenimiento																												

Figura 5.2.: Diagrama de Gantt asociado al modelo en cascada aplicado.

Obsérvese el período de tiempo que suponen, de manera aproximada, los meses de febrero y marzo, en que se alternan las fases intermedias a gran velocidad, como consecuencia de la implementación de las diferentes versiones del conversor, hasta la estabilización última que llega alrededor de comienzos de abril.

Finalmente, el mes de junio supone un momento de mantenimiento del conversor, de fin de elaboración de la aplicación de filtros y, en última instancia, de la confección de una interfaz gráfica para ambas aplicaciones producto del trabajo.

Así, en definitiva, el cálculo semanal por etapas resulta tal como sigue:

Etapas	Duración
Análisis	5 semanas
Diseño	10 semanas
Implementación	11 semanas
Experimentación	12 semanas
Mantenimiento	2 semanas

Figura 5.3.: Tabla de duración temporal de ejecución de las etapas del modelo en cascada.

Finalmente, asumiendo un salario fijado a los 30 €/hora para un desarrollador software especializado en análisis de señales, y teniendo en consideración los gastos propiciados por consumo energético y de red (electricidad y conexión a internet), así como el ordenador portátil utilizado, el coste total del proyecto asciende a los 7877 €. Se suponen, para ello, un consumo aproximado de 0.4 kW/h y un coste de 0.1579 €/kW en lo que a electricidad respecta, y una tarifa de red de 47 €/mes.

A continuación se adjunta una tabla que muestra el desglose de costes de realización:

Recursos	Coste
Salario	6750 €
Ordenador portátil	1100 €
Electricidad	12 €
Conexión a internet	15 €
Total	7877 €

Figura 5.4.: Desglose de costes del proyecto.

6. Implementación

Se describen brevemente ahora los aspectos más relevantes en relación a la escritura y ejecución de código a lo largo de trabajo, ya no únicamente con miras al desarrollo de las aplicaciones principales, sino también en cuanto a la redacción de la presente memoria.

Comenzamos, en primer lugar, exponiendo el contexto de trabajo a niveles hardware y software.

6.1. Entorno de desarrollo

Este trabajo de fin de grado ha sido elaborado utilizando los lenguajes de programación que ahora se nombran:

- **Python:** Lenguaje de alto nivel interpretado y multiparadigma, uno de los más conocidos y empleados actualmente, no solo por su estilo sencillo, sino también por su enorme versatilidad, especialmente en el ámbito de APIs importables de manera gratuita.
- **LaTeX:** Lenguaje dedicado a la redacción de textos de alta calidad tipográfica, de código abierto.

En cuanto a los entornos de desarrollo software y utilidades específicas utilizadas, cabe mencionar las que siguen:

- **Jupyter Notebook (Anaconda):** Entorno de desarrollo interactivo para Python, que posibilita la alternancia de ventanas de texto plano y de código ejecutable de manera modular, extremadamente útil a la confección de código pertinentemente documentado y la ejecución por ventanas o células en pos de unas mejores depuración y fase de pruebas. Producto de la fundación *Proyecto Jupyter*, se ha manejado en el seno del gestor de entornos Anaconda.
- **Visual Studio Code:** Entorno clásico de desarrollo multilenguaje desarrollado por Microsoft para Linux, Windows, macOS y Web.
- **TeXstudio:** IDE específico de código LaTeX, ejecutable en Linux, Windows y macOS.
- **iZotope RX 8:** Se trata de una poderosa herramienta que permite la edición profunda y minuciosa de señales de audio, además de proveer tanto de interesantes utilidades de visualización o análisis, como de síntesis de audio. Desarrollado por la compañía iZotope.

6. Implementación

- **Musescore 4:** Programa clásico de edición de partituras y notación musical para Linux, Windows y macOS. Software libre y de código abierto.
- **Sibelius:** Igualmente, herramienta mundialmente utilizada en el ámbito de la composición musical como editor de partituras de cabecera, actualmente en manos de *Avid Technology*.

Resulta adecuado mencionar, además, el uso de librerías de Python cruciales en el desarrollo del proyecto, como son: *numpy*, para cálculo numérico; *sympy*, para cálculo simbólico; *scipy.io*, para la lectura de ficheros WAV; *scipy.signal*, para la importación de métodos sobre señales; *pandas*, para el trabajo con *DataFrames*; *sklearn.cluster*, para la ejecución de *K-Means*; *music21*, crucial para la composición de la partitura; y *tkinter*, para el diseño de la interfaz gráfica.

Por otro lado, la máquina local en que se ha realizado la totalidad del trabajo, y que previamente se reveló ser un ordenador portátil, posee las siguientes características:

- **Sistema operativo:** Microsoft Windows 11 Pro.
- **Versión (SO):** 23H2.
- **Modelo:** ASUS ZenBook 15 UX533FD.
- **Procesador:** Intel(R) Core(TM) i7-8565U @ 1.80GHz 1.99 GHz.
- **RAM:** 16 GB.

Finalmente, se ofrece el enlace web del repositorio de GitHub ¹ en el cual se localizan los archivos principales de los que se compone el proyecto software asociado al trabajo. En él, anexo al presente documento, se encuentran además dos directorios, cada uno de los cuales guarda tanto los ejecutables, como los ficheros de código y Notebooks de Jupyter, así como las imágenes que conforman su interfaz gráfica, uno de los audios de ejemplo utilizados y un *readme* para su compilación independiente.

¹Enlace al repositorio: <https://github.com/Lhesare1/DGIIM-TFG-23-24>

7. Tutorial de uso

Finalmente, se ofrece un breve tutorial para la utilización de las utilidades resultado de este trabajo de fin de grado.

Veremos, en definitiva, lo que ofrecen las interfaces gráficas de ambos programas y qué le cabe esperar al usuario de aplicación. En cualquier caso, sendas aplicaciones son autónomamente explicativas, y tan solo requieren los parámetros de entrada adecuados para operar y emitir la salida correspondiente.

7.1. WAV2MIDIConverter

Comenzamos con el conversor, de nombre *WAV2MIDIConverter*. El programa se constituye de un único archivo ejecutable, siendo absolutamente *standalone*.

Hacer click en el archivo accionará la apertura de la siguiente ventana:

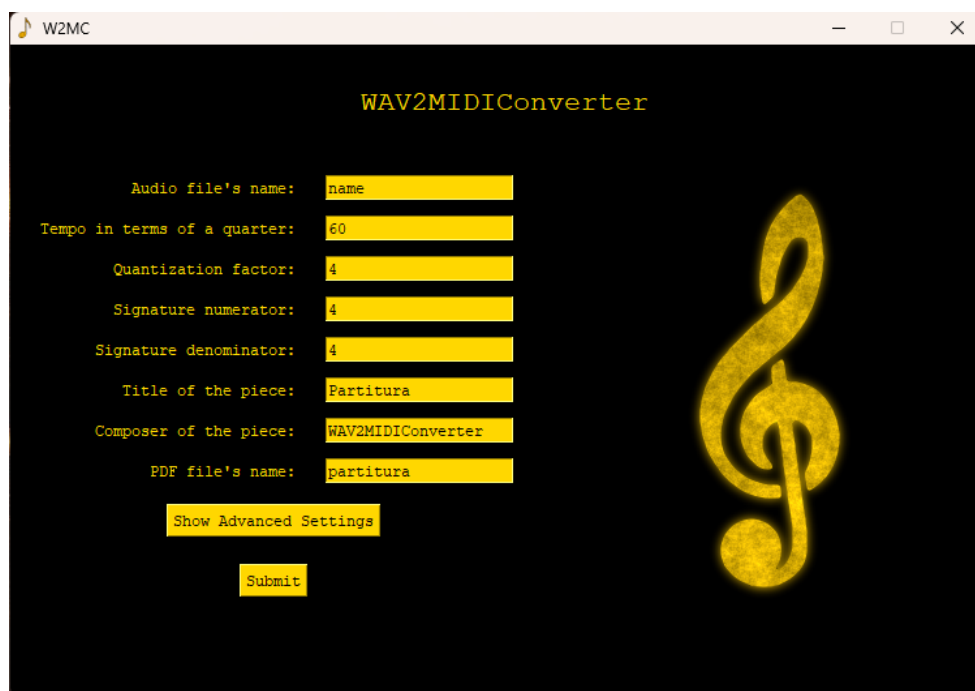


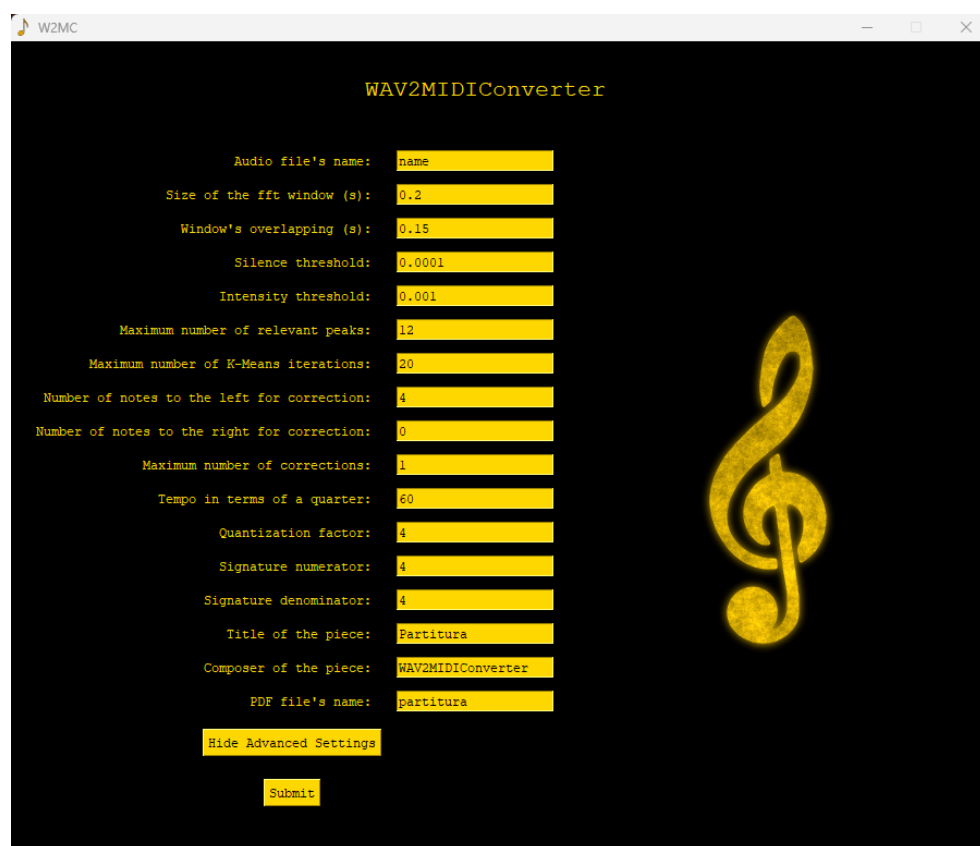
Figura 7.1.: Interfaz del programa de aplicación *WAV2MIDIConverter*.

Disponemos de los siguientes parámetros de entrada, inicializados previamente a valores por defecto, la mayoría de los cuales se han fijado de manera puramente experimental:

7. Tutorial de uso

- **Nombre del archivo WAV:** Nombre del fichero ubicado al nivel del ejecutable, en el mismo directorio.
- **Tempo de negra:** Según la indicación metronómica de la melodía.
- **Factor de cuantización:** Indica la unidad mínima de subdivisión (1 para negra, 2 para corchea, 4 para semicorchea y 8 para fusa).
- **Numerador y denominador del compás**
- **Título de la pieza**
- **Nombre del compositor**
- **Nombre del fichero de salida**

Ahora bien, si pulsamos el botón *"Show Advanced Settings"*, se mostrarán parámetros que operan sobre variables de más bajo nivel en el proceso de conversión:



The screenshot shows a window titled 'W2MC' with a black background and yellow text. The title 'WAV2MIDIConverter' is centered at the top. Below it, a list of settings is displayed, each with a label and a corresponding input field containing a value. The settings are: Audio file's name: name, Size of the fft window (s): 0.2, Window's overlapping (s): 0.15, Silence threshold: 0.0001, Intensity threshold: 0.001, Maximum number of relevant peaks: 12, Maximum number of K-Means iterations: 20, Number of notes to the left for correction: 4, Number of notes to the right for correction: 0, Maximum number of corrections: 1, Tempo in terms of a quarter: 60, Quantization factor: 4, Signature numerator: 4, Signature denominator: 4, Title of the piece: Partitura, Composer of the piece: WAV2MIDIConverter, and PDF file's name: partitura. At the bottom, there are two buttons: 'Hide Advanced Settings' and 'Submit'. A large yellow treble clef is positioned on the right side of the window.

Parameter	Value
Audio file's name:	name
Size of the fft window (s):	0.2
Window's overlapping (s):	0.15
Silence threshold:	0.0001
Intensity threshold:	0.001
Maximum number of relevant peaks:	12
Maximum number of K-Means iterations:	20
Number of notes to the left for correction:	4
Number of notes to the right for correction:	0
Maximum number of corrections:	1
Tempo in terms of a quarter:	60
Quantization factor:	4
Signature numerator:	4
Signature denominator:	4
Title of the piece:	Partitura
Composer of the piece:	WAV2MIDIConverter
PDF file's name:	partitura

Buttons: Hide Advanced Settings, Submit

Figura 7.2.: Parámetros avanzados de *WAV2MIDIConverter*.

Surgen, además de los anteriores,

- **Tamaño de ventana en segundos**
- **Solapamiento entre ventanas en segundos**
- **Umbral de silencio:** Proporción del máximo de las FFTs bajo el cual se considera que una ventana es de silencio.
- **Umbral de intensidad:** Umbral para picos demasiado bajos.
- **Máximo número de picos relevantes:** A considerar como criterio de parada para K-Means.
- **Máximo número de iteraciones para K-Means:** Criterio de parada extra.
- **Número de ventanas pasadas a utilizar en la corrección**
- **Número de ventanas pasadas a utilizar en la corrección**
- **Máximo número de correcciones:** Criterio de parada para la corrección iterativa.

Una vez configurados, tan solo se habrá de pulsar el botón *"Submit"* para lanzar la conversión de manera automática. Transcurrido el tiempo suficiente, aparecerán los archivos resultado en el directorio en que se localizan tanto el ejecutable como el archivo de partida.

7.2. 2MANYFILTERS

Finalmente, exhibimos la aplicación de filtros, llamada *2MANYFILTERS*. De nuevo, contamos con un programa *standalone* independiente.

Tras hacer click en el archivo, se manifestará una ventana de corte similar a la del programa anterior:

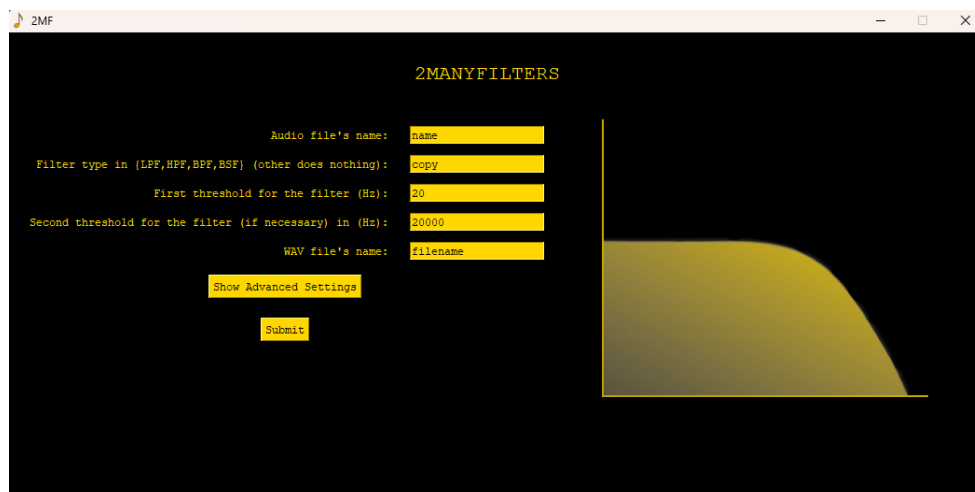


Figura 7.3.: Interfaz del programa de aplicación *2MANYFILTERS*.

7. Tutorial de uso

Ahora se presenta un menor número de parámetros de entrada, inicializados de igual manera a valores por defecto:

- **Nombre del archivo WAV:** Nombre del fichero ubicado al nivel del ejecutable, en el mismo directorio.
- **Tipo de filtro:** A elegir entre LPF, HPF, BPF y BSF. Cualquier otra cadena producirá una copia exacta del archivo original.
- **Primer umbral en Hz:** Primer o único umbral utilizado por el filtro.
- **Segundo umbral en Hz:** Segundo umbral, si procede, empleado por el filtro.
- **Nombre del fichero de salida**

Pulsado el botón *"Show Advanced Settings"* se mostrarán, análogamente, los ajustes avanzados; en este caso, solo tenemos uno: el tamaño de la ventana en segundos.



Figura 7.4.: Parámetros avanzados de *2MANYFILTERS*.

Pulsaremos el botón *"Submit"* una vez configurada la ejecución para lanzar el filtro a la manera del conversor. Rápidamente se originarán los archivos resultado en el directorio en que se localizan tanto el ejecutable como el archivo de partida.

A. Conversor

Se presenta a continuación el código final que responde al programa de aplicación *WAV2MIDIConverter*, que incluye igualmente la construcción de la interfaz gráfica, contenido en un único archivo Python: *w2mc.py*.

A.1. WAV2MIDIConverter

```
1      # WAV TO MIDI CONVERTER
2      # Javier Granados López
3      # TFG 2023/2024
4
5      # -----Imports-----
6      import sys
7      from scipy.io import wavfile
8      import os
9      import numpy as np
10     from scipy.signal import find_peaks, correlate
11     import pandas as pd
12     from sklearn.cluster import KMeans
13
14     from music21 import *
15
16     import tkinter as tk
17     from tkinter import messagebox
18     from PIL import Image, ImageTk
19
20     # -----Inputs-----
21     AUDIO_FILE = 'name' # Audio file's name
22     WINDOW_SIZE_SECS = 0.2 # Size of the fft
23     window in seconds
24     OVERLAPPING_SECS = 0.15 # Window's
25     overlapping in seconds
26     SILENCE_THRESHOLD = 0.0001 # Intensity
27     threshold for silence in [0,1]
28     INTENSITY_THRESHOLD = 0.001 # Intensity
29     (relevance) threshold for frequencies
30     MAX_REL_PEAKS = 12 # Maximum number of
31     peaks in the cluster of relevant peaks
32     MAX_KM_ITERATIONS = 20 # Maximum number of
33     K-Means iterations
34     N_NOTES_CORRECTION_L = 4 # Number of notes
35     to the left to consider for correcting a note
36     N_NOTES_CORRECTION_R = 0 # Number of notes
37     to the right to consider for correcting a note
38     MAX_CORRECTIONS = 1 # Maximum number of
39     corrections
40
41     QUARTER_PPM = 60 # Tempo in terms of
42     a quarter
43     QUANTIZATION_FACTOR = 4 # Briefest musical
44     figure to consider
45
46     SIGNATURE_NUM = 4 # Numerator of the
47     signature
48     SIGNATURE_DEN = 4 # Denominator of
49     the signature
50     TITLE = 'Partitura' # Title of the piece
51     COMPOSER = 'WAV2MIDIConverter' # Composer of the
52     piece
```

A. Conversor

```

39  FILENAME = 'partitura'                                # The pdf file's
    name
40
41  # -----Global variables-----
42  WDIMENSIONS = {'little': '750x500', 'large': '900x750'}
43  DEFAULTS = {                                          # Default values
44      "AUDIO_FILE" : AUDIO_FILE,
45      "WINDOW_SIZE_SECS": WINDOW_SIZE_SECS,
46      "OVERLAPPING_SECS": OVERLAPPING_SECS,
47      "SILENCE_THRESHOLD": SILENCE_THRESHOLD,
48      "INTENSITY_THRESHOLD": INTENSITY_THRESHOLD,
49      "MAX_REL_PEAKS": MAX_REL_PEAKS,
50      "MAX_KM_ITERATIONS": MAX_KM_ITERATIONS,
51      "N_NOTES_CORRECTION_L": N_NOTES_CORRECTION_L,
52      "N_NOTES_CORRECTION_R": N_NOTES_CORRECTION_R,
53      "MAX_CORRECTIONS": MAX_CORRECTIONS,
54      "QUARTER_PPM": QUARTER_PPM,
55      "QUANTIZATION_FACTOR": QUANTIZATION_FACTOR,
56      "SIGNATURE_NUM": SIGNATURE_NUM,
57      "SIGNATURE_DEN": SIGNATURE_DEN,
58      "TITLE": TITLE,
59      "COMPOSER": COMPOSER,
60      "FILENAME": FILENAME
61  }
62  PATH = os.path.dirname(os.path.realpath(sys.argv[0]))    # Path of
    project's directory
63  NOTES = ["C", "C#", "D", "D#", "E", "F", "F#", "G", "G#", "A", "A#", "B"] # The twelve notes'
    names
64
65  SAMPLE_RATE = None                                     # Sample rate
    (samples per second)
66  SIGNAL = None                                          # Signal data
67  WINDOW_SIZE_SAMPLES = None                             # Size of the fft
    window in samples
68  OVERLAPPING_SAMPLES = None                             # Size of
    overlapping in samples
69  AUDIO_SIZE_SECS = None                                 # Size of the audio
    file in seconds
70
71  QUARTER_SECS = None                                    # Seconds of a
    quarter
72
73  # -----Functions-----
74  # Notes' detection
75
76  def freq_to_number(f):                                  # Transforms
    any note's frequency into its midi number
77      return 69 + 12*np.log2(f/440.0)
78
79  def number_to_freq(n):                                  # Transforms
    any note's midi number into its frequency
80      return 440 * 2.0**((n-69)/12.0)
81
82  def note_name(n):                                       # Gets the
    note's name given its midi number
83      return NOTES[n % 12] + str(int(n/12 - 1))
84
85  def extract_window(audio, window_number):               # Returns
    samples of window number <window-number> and true or false whether it's the last window
86      begin = window_number * (WINDOW_SIZE_SAMPLES - OVERLAPPING_SAMPLES)
87      end = begin + WINDOW_SIZE_SAMPLES
88
89      if end < len(SIGNAL): # Commonly
90          return False, audio[begin:end]
91      else: # The window surpasses the audio data => Complete last elements of the window
        with zeros
92          return True,
            np.concatenate([audio[begin:len(SIGNAL)-1], np.zeros(end-len(SIGNAL)+1, dtype=float)])
93
94  def autocorrelation(window):                             #
    Autocorrelation of a given window

```



```

95     ac = correlate(window,window,mode='full')
96     return ac[int(len(ac)/2):]
97
98     def indexes(freqs,i1,i2,harmonic): # Returns h1
99     and h2 indexes of the nearest two #
100     if i2-i1 == 1: #
101     harmonics of window's fund. to harmonic or # h1 the
102     if harmonic == freqs[i1]: #
103     index of harmonic in freqs and h2<0
104     return i1,-1
105     elif harmonic == freqs[i2]:
106     return i2,-1
107     else:
108     return i1,i2
109     else:
110     isplit = int(i1 + np.ceil((i2-i1)/2.0))
111     if harmonic < freqs[isplit]:
112     return indexes(freqs,i1,isplit,harmonic)
113     elif harmonic > freqs[isplit]:
114     return indexes(freqs,isplit,i2,harmonic)
115     else:
116     return isplit,-1
117
118     def remove_duplicates(seq): # Remove duplicates preserving order
119     seen = set()
120     seen_add = seen.add
121     return [x for x in seq if not (x in seen or seen_add(x))]
122
123     def detect_peaks(freqs,F): # Returns the array of freqs where fft has relevant peaks
124     peaks_before = []
125     peaks_after = []
126     pindex = find_peaks(F)
127     if not len(pindex[0]):
128     return []
129     num_clusters = 2
130     nIterations = 0
131     P = pd.DataFrame(data=[F[i] for i in
132     pindex[0]],index=pindex[0],columns=['Intensity'])
133
134     kmeans = KMeans(n_clusters=num_clusters,n_init=5,random_state=123456)
135     clusters = kmeans.fit_predict(P) # Detect two clusters: peaks and non peaks
136     cluster_id = clusters[np.argmax([F[i] for i in pindex[0]])] # Cluster of non
137     relevant peaks id
138     rpindex = np.where(clusters != cluster_id)[0] # Indexes of relevant peaks
139     peaks_after = [freqs[i] for i in [pindex[0][j] for j in rpindex]]
140     num_clusters += 1
141     peaks_before = peaks_after.copy()
142     while len(peaks_after) <= MAX_REL_PEAKS and nIterations < MAX_KM_ITERATIONS:
143     peaks_before = peaks_after.copy()
144     nIterations += 1
145     kmeans = KMeans(n_clusters=num_clusters,n_init=5,random_state=123456)
146     clusters = kmeans.fit_predict(P) # Detect <num_clusters> clusters
147     cluster_id = clusters[np.argmax([F[i] for i in pindex[0]])] # Cluster of non
148     relevant peaks id
149     rpindex = np.where(clusters != cluster_id)[0] # Indexes of relevant peaks
150     peaks_after = [freqs[i] for i in [pindex[0][j] for j in rpindex]]
151     num_clusters += 1
152
153     return peaks_before
154
155     def find_candidates(cset): # Finds candidates for fundamental by subtracting elements
156     in cset
157     aux_cset = [c for c in cset if c >= 27.5].copy() # Remove too low freqs
158     aux_cset.sort() # Order
159     candidates = aux_cset.copy()
160     for i in range(0,len(aux_cset)-1):
161     candidate = number_to_freq(int(round(freq_to_number(aux_cset[i+1] -
162     aux_cset[i])))) # Round to equal temperament
163     if candidate not in candidates:
164     candidates.append(candidate)
165     return [c for c in candidates if c >= 27.5] # Remove too low freqs

```

```

158
159     def count_harmonics(peaks,candidates,m): # Count the number of harmonics in peaks for
each candidate
160         nharmonics = np.zeros(len(candidates),dtype=float)
161         for i in range(0,len(candidates)):
162             for j in range(0,len(peaks)):
163                 if peaks[j] >= candidates[i]:
164                     div = np.modf(peaks[j]/candidates[i])[0]
165                     if np.abs(div-round(div)) < 0.01: # Check if harmonic
166                         nharmonics[i] += 1
167         for i in range(0,len(candidates)):
168             samples = max(1,round(m/candidates[i]))
169             nharmonics[i] /= np.power(samples,0.1)
170
171         return nharmonics
172
173     def count_differences(peaks): # Count the number of harmonics in peaks for each candidate
174         aux_peaks = peaks.copy()
175         aux_peaks.sort()
176         aux_peaks.insert(0,0.0);
177         differences = []
178         counts = []
179
180         for i in range(0,len(aux_peaks)):
181             for j in range(i+1,len(aux_peaks)):
182                 diff = number_to_freq(int(round(freq_to_number(aux_peaks[j] -
aux_peaks[i]))))
183                 if diff not in differences:
184                     differences.append(diff)
185                     counts.append(1)
186                 else:
187                     counts[differences.index(diff)] += 1
188
189         return differences[np.argmax(counts)]
190
191     def max_amplitude(fund,freqs,F): # Compute the maximum of amplitudes in fund harmonics
as weight of the note
192         max_amp = 0
193         num_harmonic = 1
194         harmonic = num_harmonic * fund
195         top_freq = (len(F)-1) * freqs[1]
196         while harmonic <= top_freq:
197             # Compute the indexes of the nearest two harmonics of window's fund. to harmonic
198             h1,h2 = indexes(freqs,0,len(freqs)-1,harmonic)
199             if h2 < 0:
200                 max_amp = max([max_amp,F[h1]])
201             else:
202                 # Weighted mean of F[h1] and F[h2] by distance of freqs[h1] and freqs[h2] to
the harmonic
203                 if freqs[h1] != 0:
204                     max_amp = max([max_amp,(F[h1]*np.log2(freqs[h2]/harmonic) +
F[h2]*np.log2(harmonic/freqs[h1])) / np.log2(freqs[h2]/freqs[h1])])
205
206             num_harmonic += 1
207             harmonic = num_harmonic * fund
208
209         return max_amp
210
211     def detect_note(fft):
212         freqs = np.fft.rfftfreq(WINDOW_SIZE_SAMPLES, 1/SAMPLE_RATE) # The array of
frequencies to evaluate in the fft
213         F = np.abs(fft.real) # Evaluations of those frequencies
214
215         peaks = [number_to_freq(round(freq_to_number(i))) for i in detect_peaks(freqs,F)] #
Round to equal temperament
216         if not len(peaks):
217             return ('S',0)
218         peaks = remove_duplicates(peaks) # Remove duplicates
219
220         candidates = find_candidates(peaks)
221         nharmonics = count_harmonics(peaks,candidates,freqs[-1])

```

```

222     pred_fund = candidates[np.argmax(nharmonics)]
223
224
225     return
226     (note_name(int(round(freq_to_number(pred_fund)))), max_amplitude(pred_fund, freqs, F))
227
228 def correct_notes_iteration(notes, weights, nl, nr): # Correct each note according to its
229 nl previous and nr following ones' weights
230     cnotes = notes.copy() # Necessary for keeping the first and last notes
231     cweights = weights.copy() # Necessary for keeping the first and last weights
232
233     silence_threshold = SILENCE_THRESHOLD * max(cweights)
234     n = max(nl, nr)
235     w = [1] # New weights for the window, based on proximity to the note to be corrected
236     for k in range(1, n+1):
237         w.append(sum(w))
238
239     for i in range(0, len(notes)-nr):
240         if cnotes[i] == 'S': # Avoid correcting silence
241             continue
242         if cweights[i] <= silence_threshold: # Correct as silence and keep the weight
243             cnotes[i] = 'S'
244             continue
245
246         if i in range(0, nl) or i in range(len(notes)-nr, len(notes)): # Skip if cannot be
247             corrected for being out of the range
248             continue
249
250         nsubset = []
251         wsubset = []
252         nsums = []
253         for j in range(i-nl, i+nr+1):
254             if notes[j] not in nsubset:
255                 nsubset.append(notes[j])
256                 if j <= i:
257                     wsubset.append(w[j-(i-n)] * weights[j])
258                     nsums.append(w[j-(i-n)])
259                 else:
260                     wsubset.append(w[i+n-j] * weights[j] / 2) # Little penalization to
261                     future notes
262                     nsums.append(w[i+n-j] / 2)
263             else:
264                 if j <= i:
265                     wsubset[nsubset.index(notes[j])] += w[j-(i-n)] * weights[j]
266                     nsums[nsubset.index(notes[j])] += w[j-(i-n)]
267                 else:
268                     wsubset[nsubset.index(notes[j])] += w[i+n-j] * weights[j] / 2
269                     nsums[nsubset.index(notes[j])] += w[i+n-j] / 2
270
271         index = len(wsubset) - wsubset[::-1].index(max(wsubset)) - 1 # Index of last
272         maximum of wsubset
273         cnotes[i] = nsubset[index]
274         cweights[i] = wsubset[index] / nsums[index]
275
276     return cnotes, cweights
277
278 def correct_notes(notes, weights, nl, nr): # Correct the notes
279     count = 0
280
281     notes_before = notes.copy()
282     notes_after, cweights = correct_notes_iteration(notes, weights, nl, nr)
283     while not np.array_equal(notes_before, notes_after) and count < MAX_CORRECTIONS:
284         notes_before = notes_after.copy()
285         count += 1
286         notes_after, cweights = correct_notes_iteration(notes_before, cweights, nl, nr)
287
288     return notes_before
289
290 def notes_per_window():
291     hanning = 0.5 * (1 - np.cos(np.linspace(0, 2*np.pi, WINDOW_SIZE_SAMPLES, False))) #
292     The hanning window function

```

```

287     notes = []
288     weights = []
289     window_number = 0
290     last_window = False
291     while not(last_window):
292         last_window, window = extract_window(SIGNAL, window_number)
293         window_number += 1
294         fft = np.fft.rfft(autocorrelation(window * hanning))
295         note, weight = detect_note(fft)
296         notes.append(note)
297         weights.append(weight)
298
299     return correct_notes(notes, weights, N_NOTES_CORRECTION_L, N_NOTES_CORRECTION_R)
300
301 # Computing the durations
302
303 def durations(n_prev, d_prev): # Compute the arrays of notes and durations (in seconds)
304     from previous array or not
305     notes = []
306     durations = []
307
308     cn = n_prev[0] # Current note
309     cnd = 0 # Current note's duration
310     if not d_prev:
311         for n in n_prev:
312             if n == cn:
313                 cnd += WINDOW_SIZE_SECS - OVERLAPPING_SECS
314             else:
315                 notes.append(cn)
316                 durations.append(cnd)
317                 cn = n
318                 cnd = WINDOW_SIZE_SECS - OVERLAPPING_SECS
319         notes.append(cn)
320         durations.append(cnd + OVERLAPPING_SECS)
321     else:
322         for i in range(0, len(n_prev)):
323             if n_prev[i] == cn:
324                 cnd += d_prev[i]
325             else:
326                 notes.append(cn)
327                 durations.append(cnd)
328                 cn = n_prev[i]
329                 cnd = d_prev[i]
330         notes.append(cn)
331         durations.append(cnd)
332
333     return notes, durations
334
335 def to_quarters(durations, subdivision): # Compute the array of durations (in quarters)
336     return np.round(subdivision * np.array(durations) / QUARTER_SECS) / subdivision
337
338 def remove_zeros(n, d, q, remove_rests):
339     notes = []
340     durations = []
341     quarters = []
342     for i in range(0, len(n)):
343         if q[i] > 0 or (n[i] == 'S' and not remove_rests):
344             notes.append(n[i])
345             durations.append(d[i])
346             quarters.append(q[i])
347
348     return notes, durations, quarters
349
350 def final_notes_durations(npw, subdivision): # Compute the final arrays of notes and
351     durations (in seconds and quarters)
352     n_before = npw.copy()
353     d = []
354     q = []
355     n_after, d = durations(n_before, d)

```

```

356     q = to_quarters(d, subdivision)
357     n_after, d, q = remove_zeros(n_after, d, q, False)
358     while not np.array_equal(n_before, n_after):
359         n_before = n_after.copy()
360         n_after, d = durations(n_before, d)
361         q = to_quarters(d, subdivision)
362         n_after, d, q = remove_zeros(n_after, d, q, False)
363
364     return remove_zeros(n_before, d, q, True) # To considering little rests as separators
        between equal notes
365
366 # Conversion
367
368 def music_sheet(notes, quarters):
369     score = stream.Score() # Score
370     part = stream.Part() # Staff
371     voice = stream.Voice() # Voice
372
373     voice.append(meter.TimeSignature(str(SIGNATURE_NUM) + '/' + str(SIGNATURE_DEN))) #
        Define time signature
374
375     first_note = False # We allow clef changes after insertion of the first note (not
        rest)
376     for i in range(0, len(notes)):
377         if notes[i] == 'S':
378             voice.append(note.Rest(quarters[i]))
379             continue
380         n = note.Note(notes[i])
381         n.duration = duration.Duration(quarters[i])
382
383         cclef = None
384         if voice.getElementsByClass('Clef'): # Get current clef
385             cclef =
                voice.getElementsByClass('Clef')[len(voice.getElementsByClass('Clef'))-1]
386         else:
387             cclef = clef.bestClef(voice)
388
389         if first_note and n.octave < 4 and cclef == clef.TrebleClef(): # Condition to
            change to F clef
390             voice.append(clef.BassClef())
391         if first_note and n.octave > 3 and cclef == clef.BassClef(): # Condition to
            change to G clef
392             voice.append(clef.TrebleClef())
393
394         voice.append(n)
395         first_note = True
396
397
398     quarters_in_measure = 4 * SIGNATURE_NUM / SIGNATURE_DEN
399     last_silence_quarters = quarters_in_measure - (sum(quarters) % quarters_in_measure)
400     voice.append(note.Rest(last_silence_quarters)) # Last silence duration to end last
        measure
401
402     part.append([voice])
403     score.insert(0, part)
404
405     score.insert(0, metadata.Metadata())
406     score.metadata.title = TITLE
407     score.metadata.composer = COMPOSER
408
409     score.write('musicxml.pdf', fp=FILENAME)
410     score.write('midi', fp=FILENAME+'.midi')
411
412 # Conversion
413
414 def convert():
415     global SAMPLE_RATE
416     global SIGNAL
417     global WINDOW_SIZE_SAMPLES
418     global OVERLAPPING_SAMPLES
419     global AUDIO_SIZE_SECS

```

A. Conversor

```

420     global QUARTER_SECS
421
422     SAMPLE_RATE, data = wavfile.read(os.path.join(PATH, AUDIO_FILE)) # Get sample
423     rate (samples per second) and signal data
424     SIGNAL = data if data.ndim == 1 else data.T[0] # Get the first
425     channel
426     WINDOW_SIZE_SAMPLES = int(SAMPLE_RATE * WINDOW_SIZE_SECS) # Size of the
427     fft window in samples
428     OVERLAPPING_SAMPLES = int(SAMPLE_RATE * OVERLAPPING_SECS) # Size of
429     overlapping in samples
430     AUDIO_SIZE_SECS = len(SIGNAL) / SAMPLE_RATE # Size of the
431     audio file in seconds
432
433     QUARTER_SECS = 60.0 / QUARTER_PPM # Seconds of a
434     quarter
435
436     # -----Files' statistics-----
437     messagebox.showinfo("File's info", "Sample rate: " + str(SAMPLE_RATE) + " Hz\n" +
438     "Window size: " + str(WINDOW_SIZE_SECS) + " s = " + str(WINDOW_SIZE_SAMPLES) + "
439     samples\n" +
440     "Overlapping: " + str(OVERLAPPING_SECS) + " s = " + str(OVERLAPPING_SAMPLES) + "
441     samples\n" +
442     "Audio length: " + str(AUDIO_SIZE_SECS) + " s\n")
443
444     npw = notes_per_window()
445     notes, _, quarters = final_notes_durations(npw, QUANTIZATION_FACTOR)
446     music_sheet(notes, quarters)
447
448     # Input reading and settings
449
450     def submit(): # Validation of inputs and converter's execution
451     try:
452         global AUDIO_FILE
453         global WINDOW_SIZE_SECS
454         global OVERLAPPING_SECS
455         global SILENCE_THRESHOLD
456         global INTENSITY_THRESHOLD
457         global MAX_REL_PEAKS
458         global MAX_KM_ITERATIONS
459         global N_NOTES_CORRECTION_L
460         global N_NOTES_CORRECTION_R
461         global MAX_CORRECTIONS
462         global QUARTER_PPM
463         global QUANTIZATION_FACTOR
464         global SIGNATURE_NUM
465         global SIGNATURE_DEN
466         global TITLE
467         global COMPOSER
468         global FILENAME
469
470         AUDIO_FILE = entry_audio_file.get()
471         WINDOW_SIZE_SECS = float(entry_window_size_secs.get())
472         OVERLAPPING_SECS = float(entry_overlapping_secs.get())
473         SILENCE_THRESHOLD = float(entry_silence_threshold.get())
474         INTENSITY_THRESHOLD = float(entry_intensity_threshold.get())
475         MAX_REL_PEAKS = int(entry_max_rel_peaks.get())
476         MAX_KM_ITERATIONS = int(entry_max_km_iterations.get())
477         N_NOTES_CORRECTION_L = int(entry_n_notes_correction_l.get())
478         N_NOTES_CORRECTION_R = int(entry_n_notes_correction_r.get())
479         MAX_CORRECTIONS = int(entry_max_corrections.get())
480         QUARTER_PPM = int(entry_quarter_ppm.get())
481         QUANTIZATION_FACTOR = int(entry_quantization_factor.get())
482         SIGNATURE_NUM = int(entry_signature_num.get())
483         SIGNATURE_DEN = int(entry_signature_den.get())
484         TITLE = entry_title.get()
485         COMPOSER = entry_composer.get()
486         FILENAME = entry_filename.get()
487
488         # Validation checks
489         if not os.path.isfile(AUDIO_FILE):
490             raise ValueError("The audio file must exist in the working directory.")

```

```

483         if WINDOW_SIZE_SECS <= 0:
484             raise ValueError("WINDOW_SIZE_SECS must be greater than 0.")
485         if OVERLAPPING_SECS < 0:
486             raise ValueError("OVERLAPPING_SECS must be greater than or equal to 0.")
487         if WINDOW_SIZE_SECS <= OVERLAPPING_SECS:
488             raise ValueError("WINDOW_SIZE_SECS must be greater than OVERLAPPING_SECS.")
489         if not (0 <= SILENCE_THRESHOLD <= 1):
490             raise ValueError("SILENCE_THRESHOLD must be between 0 and 1 (inclusive).")
491         if INTENSITY_THRESHOLD <= 0:
492             raise ValueError("INTENSITY_THRESHOLD must be greater than 0.")
493         if MAX_REL_PEAKS <= 0:
494             raise ValueError("MAX_REL_PEAKS must be greater than 0.")
495         if MAX_KM_ITERATIONS <= 0:
496             raise ValueError("MAX_KM_ITERATIONS must be greater than 0.")
497         if N_NOTES_CORRECTION_L < 0:
498             raise ValueError("N_NOTES_CORRECTION_L must be greater than or equal to 0.")
499         if N_NOTES_CORRECTION_R < 0:
500             raise ValueError("N_NOTES_CORRECTION_R must be greater than or equal to 0.")
501         if MAX_CORRECTIONS < 0:
502             raise ValueError("MAX_CORRECTIONS must be greater than or equal to 0.")
503         if QUARTER_PPM <= 0:
504             raise ValueError("QUARTER_PPM must be greater than 0.")
505         if QUANTIZATION_FACTOR not in {1, 2, 4, 8}:
506             raise ValueError("QUANTIZATION_FACTOR must be one of {1, 2, 4, 8}.")
507         if SIGNATURE_NUM <= 0:
508             raise ValueError("SIGNATURE_NUM must be greater than 0.")
509         if SIGNATURE_DEN not in {1, 2, 4, 8, 16}:
510             raise ValueError("SIGNATURE_DEN must be one of {1, 2, 4, 8, 16}.")
511         if not FILENAME:
512             raise ValueError("FILENAME must not be empty.")
513
514         messagebox.showinfo("Success", "Parameters received correctly.")
515
516         convert()
517     except ValueError as e:
518         messagebox.showerror("Error", f"Invalid input:{e}")
519
520 def toggle_advanced_params(entries, advanced_entries, labels, toggle_button, root):
521     for key, (_, entry) in entries.items():
522         if key in advanced_entries:
523             if entry.winfo_ismapped(): # If visible, hide
524                 entry.grid_remove()
525                 labels[key].grid_remove()
526                 toggle_button.config(text="Show Advanced Settings")
527                 root.geometry(WDIMENSIONS['little'])
528             else: # Si hidden, show
529                 entry.grid()
530                 labels[key].grid()
531                 toggle_button.config(text="Hide Advanced Settings")
532                 root.geometry(WDIMENSIONS['large'])
533
534 # Auxiliary
535
536 def resource_path(relative_path):
537     try:
538         base_path = sys._MEIPASS
539     except Exception:
540         base_path = os.path.abspath(".")
541
542     return os.path.join(base_path, relative_path)
543
544 # Main
545
546 def main():
547     # Create the main window
548     custom_font = ('Courier', 9)
549     root = tk.Tk()
550     root.title('W2MC')
551
552     root.geometry(WDIMENSIONS['little']) # Set fixed size for the window
553     root.resizable(False, False)

```

A. Conversor

```

554 icon = ImageTk.PhotoImage(file=resource_path('icon.ico'))
555 root.tk.call('wm','iconphoto',root._w,icon) # Program's icon
556
557
558 root.config(bg='black')
559
560 title_frame = tk.Frame(root,bg='black') # Container for a title
561 title_frame.grid(row=0,column=0,columnspan=3,pady=30)
562 title_label =
563 tk.Label(title_frame,text="WAV2MIDIConverter",font=('Courier',16),bg='black',fg='gold')
564 title_label.pack()
565
566 entry_frame = tk.Frame(root,bg='black') # Container for entries and labels
567
568 entries = { # Create the widgets
569     "AUDIO_FILE": ("Audio file's name:",tk.Entry(entry_frame)),
570     "WINDOW_SIZE_SECS": ("Size of the fft window (s):",tk.Entry(entry_frame)),
571     "OVERLAPPING_SECS": ("Window's overlapping (s):",tk.Entry(entry_frame)),
572     "SILENCE_THRESHOLD": ("Silence threshold:",tk.Entry(entry_frame)),
573     "INTENSITY_THRESHOLD": ("Intensity threshold:",tk.Entry(entry_frame)),
574     "MAX_REL_PEAKS": ("Maximum number of relevant peaks:",tk.Entry(entry_frame)),
575     "MAX_KM_ITERATIONS": ("Maximum number of K-Means
576     iterations:",tk.Entry(entry_frame)),
577     "N_NOTES_CORRECTION_L": ("Number of notes to the left for
578     correction:",tk.Entry(entry_frame)),
579     "N_NOTES_CORRECTION_R": ("Number of notes to the right for
580     correction:",tk.Entry(entry_frame)),
581     "MAX_CORRECTIONS": ("Maximum number of corrections:",tk.Entry(entry_frame)),
582     "QUARTER_PPM": ("Tempo in terms of a quarter:",tk.Entry(entry_frame)),
583     "QUANTIZATION_FACTOR": ("Quantization factor:",tk.Entry(entry_frame)),
584     "SIGNATURE_NUM": ("Signature numerator:",tk.Entry(entry_frame)),
585     "SIGNATURE_DEN": ("Signature denominator:",tk.Entry(entry_frame)),
586     "TITLE": ("Title of the piece:",tk.Entry(entry_frame)),
587     "COMPOSER": ("Composer of the piece:",tk.Entry(entry_frame)),
588     "FILENAME": ("PDF file's name:",tk.Entry(entry_frame))
589 }
590
591 advanced_entries = [ # Define the advanced settings
592     "WINDOW_SIZE_SECS",
593     "OVERLAPPING_SECS",
594     "SILENCE_THRESHOLD",
595     "INTENSITY_THRESHOLD",
596     "MAX_REL_PEAKS",
597     "MAX_KM_ITERATIONS",
598     "N_NOTES_CORRECTION_L",
599     "N_NOTES_CORRECTION_R",
600     "MAX_CORRECTIONS"
601 ]
602
603 entry_frame.grid(row=1,column=0,rowspan=len(entries),columnspan=2,padx=10,pady=5,sticky="nsew")
604
605 # Set default values and place labels and entries in the window
606 labels = {}
607 for i,(key,(label_text,entry)) in enumerate(entries.items()):
608     labels[key] =
609     tk.Label(entry_frame,text=label_text,font=custom_font,bg='black',fg='gold')
610     labels[key].grid(row=i,column=0,padx=10,pady=5,sticky="e")
611     entry.config(font=custom_font,bg='gold')
612     entry.grid(row=i,column=1,padx=10,pady=5)
613     entry.insert(0,DEFAULTS[key])
614     globals()[f"entry_{key.lower()}"] = entry # Create dynamic entry variables
615     if key in advanced_entries:
616         entry.grid_remove()
617         labels[key].grid_remove()
618
619 side_image = Image.open(resource_path('clef.png')) # Side image
620 side_image = ImageTk.PhotoImage(side_image)
621 side_label = tk.Label(root,image=side_image,bg='black')
622 side_label.grid(row=1,column=2,rowspan=len(entries),padx=0,pady=0)
623
624 # Create a button to showing/hiding the advanced settings

```



```

620     toggle_button = tk.Button(entry_frame, text="Show Advanced
        Settings", font=custom_font, bg='gold', command=lambda:
621         toggle_advanced_params(entries, advanced_entries, labels, toggle_button, root))
622     toggle_button.grid(row=len(entries), columnspan=2, pady=10)
623
624     button_submit =
        tk.Button(entry_frame, text="Submit", font=custom_font, bg='gold', command=submit)
625     button_submit.grid(row=len(entries)+1, columnspan=2, pady=10)
626
627     root.mainloop() # Run the main application loop
628
629     # -----Execution-----
630
631     if __name__ == "__main__":
        main()

```


B. Filtrado

Se presenta a continuación el código final que implementa el programa de aplicación *2MANY-FILTERS*, incluyendo de la misma manera la definición de su interfaz, y contenido en un único archivo *Python*: *zmf.py*.

B.1. 2MANYFILTERS

```
1  # 2MANYFILTERS
2  # Javier Granados López
3  # TFG 2023/2024
4
5  # -----Imports-----
6  import sys
7  from scipy.io import wavfile
8  import os
9  import numpy as np
10
11  import tkinter as tk
12  from tkinter import messagebox
13  from PIL import Image, ImageTk
14
15  # -----Inputs-----
16  AUDIO_FILE = 'name' # Audio file's name
17  WINDOW_SIZE_SECS = 0.2 # Size of the fft
18  window in seconds
19  FILTER_TYPE = 'copy' # Filter type in
20  [LPF,HPF,BPF,BSF]; other does nothing
21  THRESHOLD1 = 20 # First threshold
22  for the filter in Hz
23  THRESHOLD2 = 20000 # Second threshold
24  for the filter (if necessary) in Hz
25  FILENAME = 'filename' # The WAV file's
26  name
27
28  # -----Global variables-----
29  WDIMENSIONS = '1050x500'
30  DEFAULTS = { # Default values
31      "AUDIO_FILE": AUDIO_FILE,
32      "WINDOW_SIZE_SECS": WINDOW_SIZE_SECS,
33      "FILTER_TYPE": FILTER_TYPE,
34      "THRESHOLD1": THRESHOLD1,
35      "THRESHOLD2": THRESHOLD2,
36      "FILENAME": FILENAME
37  }
38  PATH = os.path.dirname(os.path.realpath(sys.argv[0])) # Path of project's
39  directory
40
41  OVERLAPPING_SECS = None # Window's
42  overlapping in seconds
43  SAMPLE_RATE = None # Sample rate
44  (samples per second)
45  SIGNAL = None # Signal data
46  WINDOW_SIZE_SAMPLES = None # Size of the fft
47  window in samples
48  OVERLAPPING_SAMPLES = None # Size of
49  overlapping in samples
50  AUDIO_SIZE_SECS = None # Size of the audio
51  file in seconds
```

B. Filtrado

```
41 HANNING = None # The hanning
42 window function
43
44 # -----Functions-----
45 # Filtering
46
47 def extract_window(audio, window_number): # Returns
48     samples of window number <window-number> and true or false whether it's the last window
49     begin = window_number * (WINDOW_SIZE_SAMPLES - OVERLAPPING_SAMPLES)
50     end = begin + WINDOW_SIZE_SAMPLES
51
52     if end < len(SIGNAL): # Commonly
53         return False, audio[begin:end]
54     else: # The window surpasses the audio data => Complete last elements of the window
55         with zeros
56         return True,
57         np.concatenate([audio[begin:len(SIGNAL)-1], np.zeros(end-len(SIGNAL)+1, dtype=float)])
58
59 def analysis(window): # Compute the FFT's module curve and return x and y axes in a tuple
60     freqs = np.fft.rfftfreq(WINDOW_SIZE_SAMPLES, 1/SAMPLE_RATE) # The array of
61     frequencies to evaluate in the fft
62     fft = np.fft.rfft(window) # Evaluations of those frequencies
63
64     return (freqs, fft)
65
66 def filter_window(fft): # Filter the FFT's module function according to the selected
67     filter and thresholds
68     if FILTER_TYPE == 'LPF':
69         filtered = [fft[1][i] if fft[0][i] <= THRESHOLD1 else 0 for i in
70                     range(0, len(fft[0]))]
71     elif FILTER_TYPE == 'HPF':
72         filtered = [fft[1][i] if fft[0][i] >= THRESHOLD1 else 0 for i in
73                     range(0, len(fft[0]))]
74     elif FILTER_TYPE == 'BPF':
75         filtered = [fft[1][i] if fft[0][i] >= THRESHOLD1 and fft[0][i] <= THRESHOLD2
76                     else 0 for i in range(0, len(fft[0]))]
77     elif FILTER_TYPE == 'BSF':
78         filtered = [fft[1][i] if fft[0][i] <= THRESHOLD1 or fft[0][i] >= THRESHOLD2 else
79                     0 for i in range(0, len(fft[0]))]
80     else:
81         filtered = fft[1].copy()
82
83     return filtered
84
85 def synthesis(ffts): # Synthesize the new signal via the list of windows considering
86     overlapping
87     new_signal = (np.real(np.fft.irfft(ffts[0])) * HANNING).astype(SIGNAL.dtype)
88     for i in range(1, len(ffts)):
89         window = (np.real(np.fft.irfft(ffts[i])) * HANNING).astype(SIGNAL.dtype)
90         for j in range(0, OVERLAPPING_SAMPLES):
91             new_signal[-OVERLAPPING_SAMPLES+j] = new_signal[-OVERLAPPING_SAMPLES+j] +
92             window[j]
93         new_signal = np.append(new_signal, window[OVERLAPPING_SAMPLES:])
94
95     return new_signal
96
97 def filtering():
98     filtered_windows = []
99     window_number = 0
100     last_window = False
101     while not(last_window):
102         last_window, window = extract_window(SIGNAL, window_number)
103         window_number += 1
104         filtered_windows.append(filter_window(analysis(window)))
105
106     filtered_signal = synthesis(filtered_windows)
107     wavfile.write(os.path.join(PATH, FILENAME+".wav"), SAMPLE_RATE, filtered_signal)
108
109 # Input reading and settings
110
111 def submit(): # Validation of inputs and converter's execution
```

```

100     try:
101         global AUDIO_FILE
102         global WINDOW_SIZE_SECS
103         global FILTER_TYPE
104         global THRESHOLD1
105         global THRESHOLD2
106         global FILENAME
107
108         AUDIO_FILE = entry_audio_file.get()
109         WINDOW_SIZE_SECS = float(entry_window_size_secs.get())
110         FILTER_TYPE = entry_filter_type.get()
111         THRESHOLD1 = float(entry_threshold1.get())
112         THRESHOLD2 = float(entry_threshold2.get())
113         FILENAME = entry_filename.get()
114
115         # Validation checks
116         if not os.path.isfile(AUDIO_FILE):
117             raise ValueError("The audio file must exist in the working directory.")
118         if WINDOW_SIZE_SECS <= 0:
119             raise ValueError("WINDOW_SIZE_SECS must be greater than 0.")
120         if THRESHOLD1 <= 0:
121             raise ValueError("THRESHOLD1 must be greater than 0.")
122         if THRESHOLD2 <= 0:
123             raise ValueError("THRESHOLD2 must be greater than 0.")
124         if THRESHOLD2 < THRESHOLD1:
125             raise ValueError("THRESHOLD1 must be greater than or equal to THRESHOLD2.")
126         if not FILENAME:
127             raise ValueError("FILENAME must not be empty.")
128
129         messagebox.showinfo("Success", "Parameters received correctly.")
130
131         global OVERLAPPING_SECS
132         global HANNING
133         global SAMPLE_RATE
134         global SIGNAL
135         global WINDOW_SIZE_SAMPLES
136         global OVERLAPPING_SAMPLES
137         global AUDIO_SIZE_SECS
138
139         OVERLAPPING_SECS = WINDOW_SIZE_SECS / 2 # Mandatory because of the fixed hanning
140         function
141         SAMPLE_RATE, data = wavfile.read(os.path.join(PATH, AUDIO_FILE)) # Get
142         sample rate (samples per second) and signal data
143         SIGNAL = data if data.ndim == 1 else data.T[0] # Get the
144         first channel
145         WINDOW_SIZE_SAMPLES = int(SAMPLE_RATE * WINDOW_SIZE_SECS) # Size of
146         the fft window in samples
147         OVERLAPPING_SAMPLES = int(SAMPLE_RATE * OVERLAPPING_SECS) # Size of
148         overlapping in samples
149         AUDIO_SIZE_SECS = len(SIGNAL) / SAMPLE_RATE # Size of
150         the audio file in seconds
151         HANNING = 0.5 * (1 - np.cos(np.linspace(0, 2*np.pi, WINDOW_SIZE_SAMPLES, False)))
152
153         # -----Files' statistics-----
154         messagebox.showinfo("File's info", "Sample rate: " + str(SAMPLE_RATE) + " Hz\n" +
155         "Window size: " + str(WINDOW_SIZE_SECS) + " s = " + str(WINDOW_SIZE_SAMPLES) + "
156         samples\n" +
157         "Overlapping: " + str(OVERLAPPING_SECS) + " s = " + str(OVERLAPPING_SAMPLES) + "
158         samples\n" +
159         "Audio length: " + str(AUDIO_SIZE_SECS) + " s\n")
160
161         filtering()
162     except ValueError as e:
163         messagebox.showerror("Error", f"Invalid input:{e}")
164
165     def toggle_advanced_params(entries, advanced_entries, labels, toggle_button, root):
166         for key, (_, entry) in entries.items():
167             if key in advanced_entries:
168                 if entry.winfo_ismapped(): # If visible, hide
169                     entry.grid_remove()
170                     labels[key].grid_remove()

```

```

163         toggle_button.config(text="Show Advanced Settings")
164         root.geometry(WDIMENSIONS)
165     else: # Si hidden, show
166         entry.grid()
167         labels[key].grid()
168         toggle_button.config(text="Hide Advanced Settings")
169         root.geometry(WDIMENSIONS)
170
171 # Auxiliary
172
173 def resource_path(relative_path):
174     try:
175         base_path = sys._MEIPASS
176     except Exception:
177         base_path = os.path.abspath(".")
178
179     return os.path.join(base_path, relative_path)
180
181 # Main
182
183 def main():
184     # Create the main window
185     custom_font = ('Courier',9)
186     root = tk.Tk()
187     root.title('2MF')
188
189     root.geometry(WDIMENSIONS) # Set fixed size for the window
190     root.resizable(False, False)
191
192     icon = ImageTk.PhotoImage(file=resource_path('icon.ico'))
193     root.tk.call('wm', 'iconphoto', root._w, icon) # Program's icon
194
195     root.config(bg='black')
196
197     title_frame = tk.Frame(root,bg='black') # Container for a title
198     title_frame.grid(row=0,column=0,columnspan=3,pady=30)
199     title_label =
200     tk.Label(title_frame,text="2MANYFILTERS",font=('Courier',16),bg='black',fg='gold')
201     title_label.pack()
202
203     entry_frame = tk.Frame(root,bg='black') # Container for entries and labels
204
205     entries = { # Create the widgets
206         "AUDIO_FILE": ("Audio file's name:",tk.Entry(entry_frame)),
207         "WINDOW_SIZE_SECS": ("Size of the fft window (s):",tk.Entry(entry_frame)),
208         "FILTER_TYPE": ("Filter type in {LPF,HPF,BPF,BSF} (other does
209         nothing):",tk.Entry(entry_frame)),
210         "THRESHOLD1": ("First threshold for the filter (Hz):",tk.Entry(entry_frame)),
211         "THRESHOLD2": ("Second threshold for the filter (if necessary) in
212         (Hz):",tk.Entry(entry_frame)),
213         "FILENAME": ("WAV file's name:",tk.Entry(entry_frame))
214     }
215
216     advanced_entries = ["WINDOW_SIZE_SECS"] # Define the advanced settings
217
218     entry_frame.grid(row=1,column=0,rowspan=len(entries),columnspan=2,padx=10,pady=5,sticky="nsew")
219
220     # Set default values and place labels and entries in the window
221     labels = {}
222     for i,(key,(label_text,entry)) in enumerate(entries.items()):
223         labels[key] =
224         tk.Label(entry_frame,text=label_text,font=custom_font,bg='black',fg='gold')
225         labels[key].grid(row=i,column=0,padx=10,pady=5,sticky="e")
226         entry.config(font=custom_font,bg='gold')
227         entry.grid(row=i,column=1,padx=10,pady=5)
228         entry.insert(0,DEFAULTS[key])
229         globals()[f"entry_{key.lower()}"] = entry # Create dynamic entry variables
230     if key in advanced_entries:
231         entry.grid_remove()
232         labels[key].grid_remove()

```

```

230     side_image = Image.open(resource_path('filter.png')) # Side image
231     side_image = ImageTk.PhotoImage(side_image)
232     side_label = tk.Label(root, image=side_image, bg='black')
233     side_label.grid(row=1, column=2, rowspan=len(entries), padx=40, pady=0)
234
235     # Create a button to showing/hiding the advanced settings
236     toggle_button = tk.Button(entry_frame, text="Show Advanced
Settings", font=custom_font, bg='gold', command=lambda:
toggle_advanced_params(entries, advanced_entries, labels, toggle_button, root))
237     toggle_button.grid(row=len(entries), columnspan=2, pady=10)
238
239     button_submit =
240     tk.Button(entry_frame, text="Submit", font=custom_font, bg='gold', command=submit)
241     button_submit.grid(row=len(entries)+1, columnspan=2, pady=10)
242
243     root.mainloop() # Run the main application loop
244
245     # -----Execution-----
246
247     if __name__ == "__main__":
248         main()

```


Bibliografía

- [AA21] John Awa-Abuon. How Does Shazam Recognize Music Accurately? <https://www.makeuseof.com/how-does-shazam-work/>, 2021.
- [AD] Analog Devices. The Scientist and Engineer's Guide to Digital Signal Processing Applications. https://www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_Ch9.pdf.
- [BB08] S. Allen Broughton and Kurt Bryan. Discrete Fourier Analysis and Wavelets: Applications to Signal and Image Processing. 2008.
- [BG] Franco S. Caspe; Adrián H. Laiuppa; Oscar A. Rodriguez; Miguel A. Banchieri and Christian L. Galasso. Estimador de Fo en tiempo real basado en el algoritmo de YIN.
- [Car97] Tim Carmell. Spectrogram Reading: What are spectrograms? <https://web.archive.org/web/19991008000849/http://cslu.cse.ogi.edu/tutordemos/SpectrogramReading/spectrogram.html>, 1997.
- [CB16] Luis Colomer Blasco. *Acústica Musical*. 2016.
- [CM91] Antonio Calvo Manzano. *Acústica físico-musical*. Real Musical, 1991.
- [Con23] Convolution. <https://en.wikipedia.org/wiki/Convolution>, 2023.
- [DO93] Tirso De Olazábal. *Acústica Musical y Organología*. Ricordi Americana, 1993.
- [dP14] Dionisio de Pedro. *Teoría completa de la música*, volume I,II. Real Musical, 2014.
- [Duo03] Javier Duoandikoetxea. Lecciones sobre las series y transformadas de Fourier. 2003.
- [Ele20] What is Low Pass Filter and its applications? <https://electrowebs.com/what-is-low-pass-filter-and-its-applications/>, 2020.
- [Eng04] Tomi Engdahl. ADSL filters explained. https://www.epanorama.net/documents/telecom/adsl_filter.html, 2004.
- [ETe23] Equal Temperament. https://en.wikipedia.org/wiki/Equal_temperament, 2023.
- [Fol92] Gerald B. Folland. *Fourier Analysis and Its Applications*. American Mathematical Society, 1992.
- [FS22] David Fernández and Diego Salas. Sistema de corrección de pitch. 2022.
- [Gan97] Kyle Gann. Just Intonation Explained. <https://www.kylegann.com/tuning.html>, 1997.
- [Guc12] John Guckert. The Use of FFT and MDCT in MP3 Audio Compression. <http://www.math.utah.edu/~gustafso/s2012/2270/web-projects/Guckert-audio-compression-svd-mdct-MP3.pdf>, 2012.
- [GW] Sarah Chen; Jean Yves Ishimwe; Fátima Villa González and Henry Wang. Note detection. <https://notedetection.weebly.com/amdf.html>.

- [IG19] Elena Ibáñez and José Luis Gámez. Series de Fourier en funciones de varias variables. Transformada discreta de Fourier. Aplicaciones al tratamiento de imágenes y vídeos. 2019.
- [IM91] IBM and Microsoft. Multimedia Programming Interface and Data Specifications 1.0. <https://www.aelius.com/njh/wavemetatools/doc/riffmci.pdf>, 1991.
- [Kuh90] William B. Kuhn. A Real-Time Pitch Recognition Algorithm for Music Applications. *Journal of Humanistic Mathematics (Claremont McKenna College)*, 14(3):60–71, 1990.
- [LN14] Nathan Lenssen and Deanna Needell. An Introduction to Fourier Analysis with Applications to Music. *Journal of Humanistic Mathematics (Claremont McKenna College)*, 4:72–91, 2014.
- [Mac09] Barbara MacCluer. *Elementary Functional Analysis*. Springer Science, 2009.
- [MID23] MIDI. <https://en.wikipedia.org/wiki/MIDI>, 2023.
- [MK11] Paul M. Mather and Magaly Koch. *Computer Processing of Remotely-Sensed Images*. John Wiley & Sons, 2011.
- [MM21] Sunny Moolchand Moryani Moryani. Diseño de un ecualizador de cuatro bandas. 2021.
- [MSAC23] Michael Scott Asato Cuthbert. Spectrograph. <https://web.mit.edu/music21/doc/index.html>, 2023.
- [Muh10] Ghulam Muhammad. Noise-Robust Pitch Detection using Auto-correlation Function with Enhancements. *Journal of King Saud University - Computer and Information Sciences*, 22:13–28, 2010.
- [NI] National Instruments. Understanding FFTs and Windowing. <https://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf>.
- [Ohm22] Ohm’s acoustic law. https://en.wikipedia.org/wiki/Ohm%27s_acoustic_law, 2022.
- [PA08] Rafael Payá Albert. Web personal. <https://www.ugr.es/~rpayá/cursosanteriores.htm>, 2008.
- [Pis12] Walter Piston. *Armonía*. Idea Música, 2012.
- [PM96] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications*. Prentice-Hall, 1996.
- [Say18] Khalid Sayood. *Introduction to Data Compression*. Elsevier, 2018.
- [Sel00] Paul Sellars. Perceptual Coding: How Mp3 Compression Works. <https://web.archive.org/web/20150731055521/http://www.soundonsound.com/sos/may00/articles/mp3.htm>, 2000.
- [SG23] Pedro Saavedra and José Luis Gámez. Modelización y tratamiento de señales digitales mediante el Análisis de Fourier. 2023.
- [Smi99] Grahame Smillie. *Analogue and Digital Communication Techniques*. Elsevier, 1999.
- [Smi07] Julius O. Smith. Mathematics of the Discrete Fourier Transform with audio applications. <https://ccrma.stanford.edu/~jos/mdft/>, 2007.
- [SRU] Sonic Research University. Spectrograph. <https://www.sfu.ca/sonic-studio-webdav/handbook/Spectrograph.html>.

- [Sun01] D. Sundararajan. *The Discrete Fourier Transform: Theory, Algorithms and Applications*. World Scientific, 2001.
- [You39] Robert Young. Terminology for Logarithmic Frequency Units. *Journal of the Acoustical Society of America*, 11(1), 1939.
- [Zam97] Joaquín Zamacois. *Tratado de armonía*, volume I. SpanPress Universitaria, 1997.