Escuela Politécnica Superior

Composición musical asistida por ordenador

Grado en Ingeniería en Sonido e Imagen en Telecomunicación

Trabajo Fin de Grado

Autor:
Sergio Ribes Miró

Tutor/es:
Pedro José Ponce de León Amador

Septiembre 2017

Universitat d'Alacant
Universidad de Alicante

Resumen

La composición algorítmica es un área de trabajo de la música creada por computador que ha sido largamente estudiada desde los años cincuenta del pasado siglo, cuando los investigadores en inteligencia artificial comenzaron a abordar el reto de construir máquinas capaces de componer música.

Desde entonces se han desarrollado muchas técnicas. Una de ellas es la de componer secuencias de notas basadas en las probabilidades de que una nota determinada esté precedida de otra u otras, técnica conocida como Cadena de Markov. Estas probabilidades pueden estimarse a partir de ejemplos de composiciones.

Las versiones más recientes del sistema Csound incluyen herramientas cuyo uso para este fin aún no han sido exploradas.

Este proyecto se centra en estudiar esas herramientas, mediante el diseño e implementación de un sistema capaz de componer una secuencia musical simple utilizando cadenas de Markov, tras haber sido entrenado utilizando composiciones previas.

El sistema diseñado consta de dos partes, la primera recibe los archivos MIDI que el usuario quiere utilizar como ejemplos y 'entrena' el sistema, generando las matrices que se utilizan en el proceso de Markov. La segunda parte recupera dichas matrices y 'compone' una secuencia musical basándose en cadenas de Markov, almacenando la secuencia en un nuevo archivo MIDI.

Palabras clave: Cadenas de Markov, Csound, MIDI, composición musical

Abstract

Algorithmic composition is a field of computer generated music work area which has been studied since 1950's, when the artificial intelligence researchers started to try to build machines that could compose music.

Since then, a lot of techniques has been developed. One of them is to compose musical sequences based on the probabilities that one specific note is preceded by another one. This technique is known as Markov Chain. These probabilities can be estimated from other composition examples.

The most recent versions of the Csound system include tools that have not yet been explored for this purpose.

This project is focused on the study of these tools, by means of the design and implementation of a system capable of composing a simple musical sequence using Markov Chains, after being trained with previous compositions.

The designed system consists of two parts, the first one receives the MIDI files that the user wants to use as examples and 'trains' the system, generating the matrices that

are used in the Markov process. The second part retrieves these matrices and composes a musical sequence based on Markov chains, storing the sequence in a new MIDI file.

Key words: Markov Chains, Csound, MIDI, musical composition

Índice general

Índice de figuras

CAPÍTULO 1
Introducción

1.1 Presentación

Este proyecto nace de la exploración de la idea de utilizar la tecnología para ayudar en el proceso de creación de una composición musical. En un entorno de programación como Csound existen numerosas herramientas para diseñar un programa que genere una idea para que el compositor pueda desarrollarla y crear su propia composición.

En concreto, este proyecto se centrará en el uso de cadenas de Markov para ello, generando una composición musical a partir de composiciones previas que el usuario introducirá al sistema para entrenarlo.

1.2 Objetivos y Motivación

El objetivo principal del proyecto es estudiar las posibilidades de Csound para componer secuencias musicales utilizando cadenas de Markov, secuencias de símbolos musicales con dependencia estadística temporal.

Para ello se implementará un sistema que reciba secuencias musicales de ejemplo y posteriormente entregue una secuencia musical resultante, generada por procesos de Markov.

Esto viene motivado por la necesidad de muchos compositores de tener una idea a partir de la cual desarrollar su creación. Como una posible solución a esta necesidad, y aprovechando los avances tecnológicos actuales, se propone un sistema que permita crear estas ideas.

1.3 Estructura de la memoria

La memoria estará dividida en las siguientes partes:

Una primera parte de introducción teórica, donde se explicarán conceptos teóricos necesarios para entender el funcionamiento del sistema.

Una segunda parte de explicación del sistema diseñado, donde se explicará su estructura y funcionamiento.

Una tercera parte de análisis de resultados con una serie de ejemplos para visualizar y entender el funcionamiento del sistema.

Una cuarta parte con las conclusiones del proyecto.

Y, finalmente, una última parte con la bibliografía utilizada.

CAPÍTULO 2
Marco Teórico

Antes de comenzar a explicar el desarrollo del proyecto es importante tener ciertos conocimientos sobre el campo en el que se desarrolla.

En primer lugar, se explicarán conceptos básicos sobre la teoría musical y su representación. A continuación, se abordará el estándar MIDI, utilizado para la representación musical en entornos informáticos. Seguidamente, se explicará el concepto de composición algorítmica y la teoría relacionada con las cadenas de Markov. Y, finalmente, se explicarán algunos conceptos sobre el entorno donde se ha desarrollado el proyecto, Csound.

2.1 Teoría Musical

En esta sección se abordarán conceptos musicales básicos necesarios para entender el funcionamiento del sistema diseñado.

Las características que definen un sonido son la altura, relacionada con la frecuencia del mismo, la duración, la dinámica, relacionada con la intensidad del mismo; el timbre, relacionado con la sensación que el oyente recibe del sonido, y la situación, relacionada con la posición de la fuente del sonido respecto del oyente.

Los conceptos de timbre y situación tienen relación con el instrumento que interpreta la música y dónde se sitúa el mismo, por tanto escapan del campo del proyecto.

La música tiene una representación llamada partitura basada en dos direcciones, la vertical, para la altura; y la horizontal, para la duración y el tiempo. El conjunto de 5 líneas equidistantes donde se representan las notas musicales se denomina pentagrama, según en qué linea o espacio de encuentre la nota se conoce su altura (dirección vertical) y según su posición horizontal se conoce en qué momento tiene lugar. Además según el que símbolo represente la nota se conoce su duración. En la figura 2.1 se puede ver un ejemplo de pentagrama con algunas notas.

2.1.1. Altura

El nombre de una nota viene dado por su altura: *DO, RE, MI, FA, SOL, LA y SI* (*C,D,E,F,G,A y B*, respectivamente en el sistema de notación musical inglés o *denominación literal*). Entre todas ellas hay una distancia de un tono, excepto entre las notas MI y FA y las notas SI y DO, donde hay un semitono. Los términos de sostenido (♯) y bemol (♭) se utilizan para indicar aquellas notas que están un semitono entre aquellas separadas por un tono, por ejemplo, un FA♯ está un semitono por encima de FA y un semitono por debajo de SOL. El símbolo del becuadro, ♮, se utiliza para devolver una nota a su estado natural cuando se encuentra alterada previamente en la partitura.

Un concepto de especial relevancia es el de octava, se trata de la distancia entre dos notas cuyas frecuencias son el doble o la mitad la una de la otra. La primera nota de cada octava es el DO, así la frecuencia un DO_2 (DO de la segunda octava) es la mitad que la de un DO_3 (DO de la tercera octava).

Otro elemento importante referente a la altura es la clave. Este elemento permite conocer la frecuencia fundamental de las notas que se representan en el pentagrama. Las claves más utilizadas son la clave de SOL, que indica que la nota situada en la segunda línea es un SOL_4, y la clave de FA, que indica que la nota en la cuarta línea es un FA_3

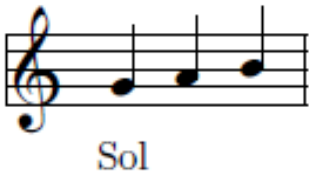

Figura 2.1: Pentagrama con Clave de SOL y tres notas: SOL, LA y SI]

El siguiente concepto interesante respecto a la altura es la afinación, consistente en la asignación de una frecuencia concreta en Hz a cada nota. Se trata de un problema importante, que se intentó resolver desde la antigua Grecia pero no se logró un sistema aceptado hasta el siglo VIII. Con los sistemas de afinación Pitagórica y Física (o Justa), el problema consistía en que cuando se calculaba el valor de la frecuencia de una nota se debía tomar una altura de referencia, sin embargo, dependiendo de la referencia tomada, el resultado no siempre era el mismo. Este problema era consecuencia de que las relaciones entre las frecuencias de las diferentes notas no eran iguales en todos los casos (ver figura 2.2).

La solución fue la afinación Bien Temperada, que consistía en desafinar levemente cada nota de manera que la relación entre las frecuencias de todas las notas fuera la misma. Para ello se divide la octava es sus 12 notas de manera igualmente proporcional entre todas ellas y obteniendo una relación para cada semitono de $2^{1/12} = 1,059463$.

Para relacionar las notas con una frecuencia real, la referencia más común es establecer la frecuencia de LA_4 igual al 440 Hz.

grado del intervalo	relaciones pitagóricas	relaciones justas	ejemplo con notas
unísono	1 : 1	1 : 1	DO – DO
segunda menor	256 : 243	16 : 15	DO – DO♯
segunda mayor	9 : 8	9 : 8	DO – RE
tercera menor	32 : 27	6 : 5	DO – MI♭
tercera mayor	81 : 64	5 : 4	DO – MI
cuarta justa	4 : 3	4 : 3	DO – FA
cuarta aumentada	729 : 512	45 : 32	DO – FA♯
quinta disminuida	1024 : 729	64 : 45	DO – SOL♭
quinta justa	3 : 2	3 : 2	DO – SOL
sexta menor	128 : 81	8 : 5	DO – LA♭
sexta mayor	27 : 16	5 : 3	DO – LA
séptima menor	16 : 9	7 : 4	DO – SI♭
séptima mayor	243 : 128	15 : 8	DO – SI
octava	2 : 1	2 : 1	DO_N – DO_{N+1}

Figura 2.2: Cuadro con las relaciones de frecuencias para las afinaciones pitagórica y justa.

2.1.2. Duración

En la notación de las notas musicales no se especifica como duración absoluta en segundos, sino que se expresa de manera relativa a una nota de referencia definida por el autor y que recibe el nombre de *tiempo* o *beat*. La notación gráfica para las duraciones define diferentes símbolos para expresar cada duración relativa siendo múltiplos y divisores entre ellos (ver figura 2.3).

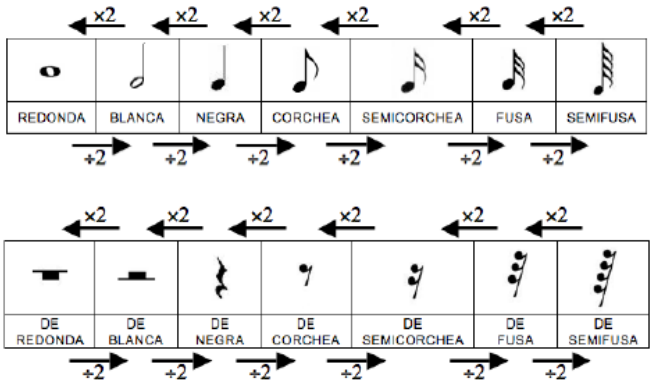

Figura 2.3: Notación gráfica de las duraciones musicales más comunes.

El concepto que relaciona las duraciones relativas con la duración real es el *tempo*, que indica la frecuencia con la que se emite la duración de referencia (en *tiempos* o *beats* por minuto, "bpm").

La métrica define el modo en el que se agrupan los tiempos dentro de una composición, en los denominados compases. La notación de la métrica se sitúa al inicio de la partitura (o al inicio del compás si hay un cambio de métrica) e indica la nota patrón, y cuantas notas patrones (y/o combinaciones de sus múltiplos o divisores) hacen falta para completar un compás.La métrica más habitual es la de 4/4

2.1.3. Dinámica

La dinámica o sonoridad son una sensación subjetiva y está fundamentalmente relacionada con la potencia o intensidad sonora, aunque también influyen en esta sensación el espectro del sonido y la duración.

La potencia sonora que llega al oyente se mide en nivel de presión sonora, NPS, siendo función de la amplitud de la onda de presión medida en newtons por metro cuadrado (N/m^2) o pascales (Pa).

Generalmente, al hablar de intensidades percibidas se hace tomando como referencia el umbral auditivo y expresándolas en dB SPL, decibelios de nivel de presión sonora. La potencia, W, es proporcional al cuadrado de la amplitud, A, de la onda de presión. La proporción de una intensidad respecto al umbral auditivo (potencia umbral,W_U, o amplitud umbral,A_U) de referencia, viene dada por:

$$dBSPL = 10log_{10}(W/W_U) = 10log_{10}(A^2/A_U^2) = 20log_{10}(A/A_U)$$

El umbral auditivo se encuentra en una amplitud de ($A_U = 2 * 10^{-5} N/m^2$), por debajo de este umbral se encuentra el silencio absoluto. Por encima de un NPS = 120 dBSPL aparece una sensación de dolor y una exposición prolongada a este NPS o superior puede ocasionar graves daños en el sistema auditivo.

2.2 Estándar MIDI

A finales de los años 70, la disponibilidad de microprocesadores baratos permitió el desarrollo de sintetizadores híbridos (analógicos en los métodos de síntesis pero controlados digital mente), sin embargo los que existían eran incompatibles entre sí: la programación de uno no servía para otro y tampoco había forma de sincronizarlos.

Hacía falta un sistema universal de comunicación entre diversos equipos, lo cual fue posible en 1981 cuando se publicó el USI (Universal System Interface) que no se impuso entre los fabricantes, por lo que estos (principalmente Oberheim, Sequential Circuits y Roland Corporation) se constituyeron en una comisión hasta que se pusieron de acuerdo en el Ïnterfaz Digital para Instrumentos Musicales"(MIDI, Musical Instruments Digital Interface), cuyos detalles se publicaron en la "Norma MIDI 1.0.^{el} 5 de agosto de 1983, aunque ya a principios de 1983 se habían sacado al mercado los primeros sintetizadores MIDI.[1]

No se trata de un formato de audio ni un lenguaje musical, sino que es un protocolo de comunicación, utilizado principalmente para la separación entre el dispositivo de control y el generador del sonido. Cuando se conectan diversos dispositivos, el estándar MIDI define los canales como un recurso para la comunicación multibanda, pudiendo transmitir por un único cable hasta 16 canales, permitiendo tener un aparato maestro que actúa como director de una orquesta y diferentes instrumentos a los que dirigir.

Actualmente, utilizando distintos programas, un ordenador puede actuar como controlador MIDI y también como generador de sonido.

Al tratarse de un protocolo de comunicación, los mensajes MIDI tienen una determinada estructura, están formados por 10 bits agrupados de 10 en 10, en *bytes* (Los dos primeros y el último son bytes reservados para indicar el tipo de mensaje). Existen diferentes tipos de mensajes, principalmente divididos en dos categorías:

Los mensajes de canal y los mensajes de sistema. Los primeros afectan únicamente al/los canales a los que van dirigidos mientras que los segundos afectan a todos los equipos que los reciben.

Dentro de los mensajes de canal se distingue entre los mensajes de voz, que se asocian con la generación de notas y su timbre; y los mensajes de modo, que se utilizan para indicar como se utilizan las voces y canales de un sintetizador.

Cuanto a los mensajes de sistema se diferencian los mensajes comunes, relacionados con la afinación y aspectos de la canción; los mensajes de tiempo real, que se utilizan en la sincronización de los diferentes equipos; y los mensajes exclusivos, que cada fabricante diseña y utiliza libremente.

Los más importantes para este proyecto son los mensajes de voz, ya que se utilizarán para generar y almacenar los diferentes sonidos resultados de la composición del sistema diseñado.

Cuando se desea generar una nota con un mensaje MIDI es necesario indicar la altura de la misma, teniendo disponibles 7 bytes para ello. Esto permite codificar un total de 128 notas, desde DO_{-1} hasta SOL_7. Es importante destacar que algunos programas (como Anvil Studio) establecen como octava 0 la octava -1 del estándar MIDI, desplazando un nivel cada una de las demás octavas.

El numero MIDI (entre 0 y 127) que codifica la altura puede hallarse a partir de la frecuencia de la nota, utilizando para ello la ecuación:

$$h = round(69 + 12log_2[f/440]$$

Cuando se genera una nota mediante un mensaje MIDI, también es posible especificar su *velocidad*, término referente a la intensidad de la nota en cuestión.

En el estándar MIDI, no es posible especificar la duración de una nota, por tanto, es necesario crear un segundo mensaje que .ªpague"la nota que esta sonando, esto se puede hacer de dos maneras: enviando un mensaje de encendido de nota con velocidad 0, o también, enviando un mensaje especifico de apagado de nota.

2.3 Composición algorítmica

Los compositores musicales siempre han estado especialmente interesados en los avances tecnológicos de su tiempo, desde el descubrimiento de la relación entre la frecuencia de una nota y la longitud de una cuerda o tubo, hasta los más avanzados modelos informáticos de cognición musical humana.[2]

El concepto de composición algorítmica hace referencia al método de creación musical por medio de procesos formales o algoritmos.[3]

La procesos de composición algorítmica se clasifican en dos tipos principalmente: estocásticos, basados en la aleatoriedad, y deterministas, cuyo resultado depende de unos valores de entrada. Además de estos dos tipos, debido a los avances tecnológicos, en la actualidad es bastante común oír hablar de los procesos evolutivos, o de aprendizaje, donde el sistema va evolucionando cuando se itera.

2.4 Teoría cadenas de Markov

Una cadena de Markov es una serie de eventos creados por un proceso estocástico cuya probabilidad viene definida teniendo en cuenta tantos eventos anteriores como valor tiene su grado (una cadena de primer orden tiene en cuenta únicamente el evento anterior, una cadena de segundo orden, los dos últimos eventos, etc.).

Una sucesión de observaciones $X_1, X_2, ...X_n$ se denomina proceso estocástico. Siendo X1 el valor que define el estado inicial del proceso, y Xn el valor que define el estado del proceso en el instante de tiempo n.

En una cadena de Markov de primer orden, es un proceso estocástico en el cual se conocen los estados actual, X_n, y anteriores, $X_1, ..., X_{n-1}$, pudiendo predecir el estado futuro,X_{n+1}, dependiendo únicamente del estado actual. Es decir, para $n = 1, 2, ...$ y para cualquier sucesión de estados $s_1, s_2, ..., s_{n+1}$:

$$P(X_{n+1} = s_{n+1}|X_1 = s_1, X_2 = s_2, ..., X_n = s_n) = P(X_{n+1} = s_{n+1}|X_n = s_n)$$

La propiedad de Markov indica que El estado en $t + 1$ sólo depende del estado en t y no de la evolución anterior del sistema.

Al trabajar con cadenas de Markov, a menudo es útil pensar la sucesión de ensayos como experimentos efectuados en cierto sistema físico, cada resultado dejando a este sistema en cierto estado. Se puede expresar la información probabilística resultado de estos experimentos en forma de matriz, conociéndose como *Matriz de transición*.

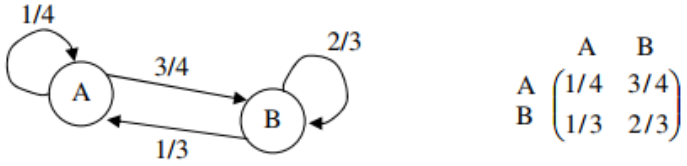

Figura 2.4: Esquema de los estados posibles de una cadena de Markov y su matriz de transición

La suma de las probabilidades de pasar de un estado *i* a cualquier otro debe sumar 1, esto se traduce en que cada una de las filas de la matriz de transición suma 1.

En la figura 2.4 puede visualizarse un esquema de una cadena de Markov con dos únicos estados posibles, y la matriz de transición correspondiente a dicha cadena.

2.5 Csound

Un sistema de programación musical es un paquete completo de software para crear música con computadores. Csound ofrece un entorno para crear música desde la base, desde elementos muy básicos que componen sus propias ondas sonoras y espectro, hasta los niveles más altos de la composición musical que involucran objetos sonoros, notas, texturas, armonía, gestos, frases, secciones, etc.

Csound es uno de los sistemas de programación musical más longevos, su desarrollo empezó en los años 80, junto a otros sistemas como Cmusic o Cmix. El lenguaje utilizado para su desarrollo fue C, que se convirtió en un estándar para el desarrollo de sistemas. Csound evolucionó de su diseño original convirtiéndose en un sistema multifuncional de composición musical mucho más amplio, sobretodo con la llegada de las versiones Csound5 (2006) y Csound6 (2013).

En la versión más reciente, Csound6, se ordenó el código y se añadió un nuevo analizador, hecho que permitió una considerable simplificación a la hora de modificar y extender el código, añadiendo gran variedad de facilidades para los programadores. Esto hace que Csound, en esta última versión, adquiera otra definición. A parte de un sistema para la creación de música, desde el punto de vista de un desarrollador, Csound puede considerarse como un motor de audio, un componente del sistema que proporciona servicios para añadir sonido a una aplicación.

Como entorno de programación, es de especial interés para el proyecto ya que permite un uso bastante simplificado de matrices (o *arrays*), elemento esencial para tratar con cadenas de Markov, así como un soporte bastante bueno para trabajar con archivos MIDI donde almacenar las partituras para entrenar al sistema y los resultados.

Los programas de Csound siguen una estructura común formada principalmente por dos partes, la orquesta y la partitura. Estas partes pueden encontrarse en un único archivo (con extensión .csd) o separadas en dos archivos independientes (con extensiones .orc para la orquesta, y .sco para la partitura).

La orquesta incluye la cabecera, donde se encuentran las variables globales reservadas, y la declaración de los diversos instrumentos, donde se realizan las operaciones con variables.

La partitura contiene las órdenes de control para los instrumentos.

Además de las partes anteriores existe una parte llamada *"CsOptions"* que contiene las opciones de compilación.

Una herramienta de especial utilidad para almacenar información y consultarla entre diversas ejecuciones son las tablas, que se almacenan como un fichero de texto y permiten almacenar la información de un vector o array unidimensional.

CAPÍTULO 3

Sistema de composición algorítmica por cadenas de Markov

El sistema diseñado tiene como objetivo generar una secuencia musical partiendo de una serie de melodías que el usuario quiere tomar como ejemplos, para generar una secuencia resultante de los mismos.

3.1 Metodología

Se han implementado dos partes diferenciadas, utilizando un archivo .csd para cada una de ellas.

La primera parte recibe el nombre de *'Entrenamiento'* o *'Training'*, recibe la melodía del usuario y seguidamente cuenta el numero de veces que aparece una altura y una duración después de otras, generando dos matrices que contienen, respectivamente, el numero de veces que una duración y una altura siguen a otras. Seguidamente, estas matrices se almacenan en archivos que se utilizan en la segunda parte.

La segunda parte, que recibe el nombre de *'Compositor'* o *'Composer'*, recibe los archivos con las matrices guardadas, y crea la matriz de transición para ejecutar el proceso de Markov. Empezando con una nota aleatoria se utiliza la Matriz de transición para buscar crear la siguiente nota en función de su probabilidad. Finalmente, se genera una secuencia, que puede sonar en tiempo real o guardarse en un archivo de sonido, además de almacenarla en un archivo MIDI.

La Figura 3.1 muestra el diagrama de flujo para ilustrar el proceso que sigue el sistema para generar la secuencia musical.

El sistema ha sido programando utilizando la versión 6.08 de Csound, con el editor CsoundQT, además para la visualización de los archivos MIDI se ha utilizado el programa Anvil Studio.

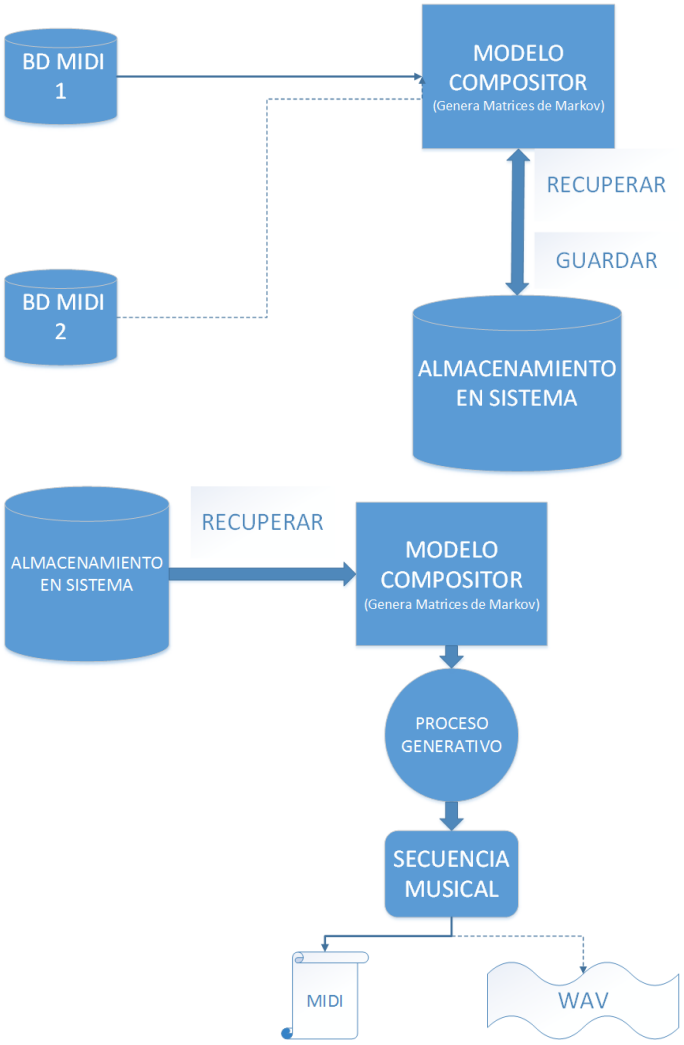

Figura 3.1: Diagrama de flujo del sistema diseñado

3.2 Entrenamiento del compositor (*Training*)

Este subsistema está formado por 4 instrumentos diferenciados: El instrumento *"inicio"*, el cual recibe los datos almacenados en archivo. Los instrumentos 1 y 2 que reciben los datos de la nota actual y incrementan las matrices. Y, finalmente, el instrumento 3 que almacena las matrices en archivo. A continuación, se procederá a explicar más detalladamente el funcionamiento de este subsistema.

El primer paso es la carga del archivo MIDI con el que se quiere realizar el entrenamiento, en la sección de opciones de compilación (*CsOptions*) se carga con el comando -F seguido de la ruta donde se encuentra el archivo. Un inconveniente es que Csound únicamente admite un archivo MIDI por cada ejecución, por tanto, si se quiere realizar el entrenamiento con diversos archivos, se tendrá que cambiar el comando anterior para ejecutar el entrenamiento cambiando el archivo cada vez.

La primera acción en la orquesta es la declaración de todas las variables globales, así como los arrays y las tablas para cada fila de la matriz que almacena el numero de apariciones de cada nota y duración posible.

El primer instrumento, llamado "*inicio*"se utiliza para consultar si existen tablas con los datos de alguna ejecución anterior y, en caso afirmativo, leer su contenido con la instrucción **ftload**. A continuación, se utiliza el comando **copyf2array** para enviar el contenido de estas tablas a un vector por cada fila, y finalmente se utiliza un bucle para introducir los valores de cada uno de estos vectores en la matriz que se utiliza para contar el numero de repeticiones de cada nota y cada duración.

A continuación se utilizan los instrumentos 1 y 2 para identificar cuál es la altura y la duración de la nota que está ejecutándose en cada momento, y rellenar las matrices teniendo en cuenta la altura y duración actuales y anteriores.

Primeramente, se utiliza el comando **notnum** para recibir la altura MIDI de la nota actual y, después,incrementar en una unidad el contador que se almacena en un vector que se utilizará para representar el histograma del entrenamiento.

El paso siguiente es recibir la altura en formato .ºctava.semitonoz la velocidad, utilizando el comando **midinoteopch**.

El proceso para calcular la duración de cada nota tiene algunas peculiaridades, ya que el formato para representar las duraciones en MIDI se basa en el inicio y el final de la nota, mientras que en Csound se hace mediante el inicio y la duración.

Para calcular la duración de las notas se utiliza el comando **timeinsts**, el cual devuelve el valor de tiempo desde que se inició el instrumento (al empezar a sonar la nota) hasta el momento actual. A continuación, se utiliza el comando **release**, el cual devuelve el valor "1çuando se detecta un mensaje MIDI de fin de nota. De esta manera, se llama al instrumento 2 cuando se detecta un final de nota, enviándole los parámetros de la nota actual.

El instrumento 2 se utiliza para rellenar las matrices contadoras de alturas y matrices. Después de recibir los parámetros de la nota actual, se clasifica la duración de la nota, utilizando unos condicionales simples anidados, para que tenga uno de los cinco valores posibles (semicorchea, corchea, negra, blanca o redonda). El primer paso es inicializar las variables de nota y duración anteriores, con los valores de la ejecución anterior. Para evitar contar notas y duraciones de más, en la primera iteración se inicializan con valores fuera del rango posible.

Recordando que la altura de la nota está en formato octava.semitono, se almacenará en una variable global el valor del semitono, de manera que quede en formato 0.semitono, para que al multiplicar por 100 este valor obtener el valor entero (entre 0 a 11) del semitono de la nota. A la hora de rellenar la matriz, se incrementa la matriz en una unidad en la posición de la fila del valor actual y la columna del valor anterior. Esta acción se realiza tanto para altura como para duración.

Uno de los principales problemas encontrados en la creación del sistema es que normalmente la interacción MIDI-csound se realiza en tiempo real y el final de la ejecución lo efectúa el usuario, por tanto, en este caso es deseable informar al usuario que puede finalizar la ejecución cuando el archivo MIDI ha llegado a su final. Por tanto, la ultima acción necesaria en este instrumento es la comprobación de si se ha alcanzado el final del

archivo MIDI, para informar al usuario que puede parar la ejecución. Para ello se usa el comando **midifilestatus**, que devuelve el valor 0 cuando el archivo MIDI llega a su final, permitiendo ejecutar el último instrumento, instrumento 4, que servirá para almacenar las matrices en archivo e informar al usuario que puede finalizar la ejecución una vez guardados los archivos.

El primer paso en el instrumento 3 es guardar el vector histograma en una tabla, utilizando el comando **copya2ftab**, y seguidamente, el comando **ftsave** para guardar la tabla en un archivo de texto.

Seguidamente, se debe extraer cada fila de las matrices de altura y duración, para poder almacenar cada una en su vector unidimensional correspondiente. Para ello se utilizara un bucle para recorrer la matriz y un condicional para ir almacenando los valores de cada fila en su vector correspondiente. En total habrá 12 vectores de alturas y 5 de duraciones. Finalmente se utilizan con cada uno de ellos los comandos **copya2ftab** y **ftsave**, para almacenarlos en tablas y guardarlas en archivos de texto. Finalmente, después de guardar todos los archivos se muestra un mensaje en consola informando al usuario que puede parar la ejecución.

En la partitura de esta parte del sistema, se llama al instrumento inicio una vez, y se deja ejecutando el instrumento 1 con una duración superior al archivo MIDI ejecutado (una hora en este caso). Para ello se utilizan las siguientes líneas de código.

```
i ïnicio"0 1

i 1 1 3600
```

3.3 Generación de la secuencia (*Composer*)

Este subsistema está formado por 4 instrumentos, el instrumento *ïnicio"* es similar al del subsistema Training. El instrumento *"trigger-note"* se utiliza para ir disparando nuevas notas. El instrumento *"select-note"* sirve para ejecutar las cadenas de Markov para alturas y duraciones. Y, finalmente, el instrumento *"play-note"* se usa para reproducir las notas y crear el archivo MIDI resultante.

En este subsistema, la sección de opciones contiene los comandos necesarios para almacenar la salida en un archivo MIDI, y en un archivo WAV (en caso de ejecutar el renderizado del programa).

```
-Q999

-midioutfile=MIDICOMPUESTO.mid

-o wavcompuesto.wav
```

En la sección de la orquesta, primeramente se encuentra la declaración de las variables (para almacenar las duraciones y alturas anteriores) y matrices globales (que serán las *matrices de transición* en el proceso de Markov), así como las tablas y vectores necesarios para recuperar los datos almacenados en archivo. Existe una variable de especial relevancia, llamada **giBasFreq** cuyo valor determina la octava en la que se centrará la

composición resultante, por defecto se le ha dado el valor de 440 Hz (LA_4), para centrar la composicion en la octava central, sin embargo es posible modificar su valor si es necesario. También es de especial relevancia el valor con el que se inicializa la variable global que almacena el valor anterior de duración, **giPrevDur**, ya que afecta a la frecuencia de generación de notas en la secuencia, su valor optimo deberá ser ajustado manualmente, teniendo en cuenta la duración predominante en los archivos de entrenamiento a fin de obtener una secuencia resultante con sentido, ya que, por ejemplo, si en el entrenamiento predominan las notas corcheas, no tiene sentido que la secuencia generada se realice a blancas. Además se definen los posibles valores para las alturas, con unos factores basados en la afinación bien temperada ($2^{1/12} = 1,059463$ para cada semitono). Y también los posibles valores para las duraciones (1/8 para semicorcheas, 1/4 para corcheas, 1/2 para negras, 1 para blancas y 2 para redondas).

Antes de empezar con los instrumentos, aparece la declaración del operador para la cadena de Markov. Este operador recibe la variable con el valor de la nota actual y la matriz de transición, y después de la ejecución, devuelve el siguiente valor que tomará, bien la altura o bien la duración. La operación realizada por el operador consiste en generar un numero aleatorio entre 0 y 1, y va recorriendo la fila del elemento actual para cada posible valor de elemento siguiente, acumulando el valor de la matriz de transición. Cuando el valor acumulado iguala o supera al numero aleatorio, entonces el último utilizado elemento siguiente es devuelto al instrumento. Para evitar posibles errores, si el valor acumulado no llegara al aleatorio, el indice de valor siguiente excedería el rango de valores válidos (entre 0 y 11 para las duraciones), por tanto si se detecta un valor superior a 11 se devuelve el indice a 0. A continuación se muestra el código que realiza esta operación:

```
opcode Markov, i, i[][]i

iMarkovTable[][], iPrevEl xin

iRandom random 0, 1

iNextEl = 0

iAccum = iMarkovTable[iPrevEl][iNextEl]

until iAccum >= iRandom do

          iNextEl += 1

          if iNextEl >11 then

                    iNextEl = 0

          endif

          iAccum += iMarkovTable[iPrevEl][iNextEl]

od

 xout iNextEl

endop
```

A continuación, se halla el instrumento *inicio*, bastante similar al del subsistema de entrenamiento. Su finalidad es leer los archivos de texto que contienen las tablas y generar las matrices de transición, para ello será necesario utilizar el comando **ftload** para cargar los archivos en tablas, y el comando **copyf2array** para almacenar cada tabla como un vector unidimensional, por cada una de las filas de las matrices.

A diferencia de en el otro subsistema, en este las matrices generadas se normalizarán por cada fila (dividiendo cada valor por la suma total de cada fila), obteniendo así las matrices de transición necesarias para calcular la cadena de Markov.

Un aspecto a tener en cuenta es cuando en una fila no se ha rellenado ningún valor, ya que la altura o duración en cuestión no aparecían en las composiciones de entrenamiento, en este caso toda la fila se rellenará con el mismo valor de probabilidad (1/5 para las duraciones o 1/12 para las alturas), así evitando posibles errores. Además, se aplica método conocido como *smoothing* o *suavizado*, consistente en aplicar una probabilidad muy baja a todas las posiciones con valor 0, de esta forma se asegura que siempre hay una probabilidad, aunque sea muy pequeña, de pasar a otro estado. El siguiente instrumento, *trigger-note*, utiliza el comando **metro**, el cual genera un metrónomo con una frecuencia igual a la inversa de la duración inicial, esta duración inicial debe ser modificada para que coincida con la duración predominante en las composiciones de entrenamiento para obtener el mejor resultado posible, permitiendo llamar al siguiente instrumento con cada uno de sus tics.

Seguidamente, el instrumento *select-note* se encarga de ejecutar el operador de Markov para la altura y duración actuales. A continuación llama al último instrumento, encargado de reproducir y guardar la nota actual.

El último instrumento, *play-note* recibe del instrumento anterior, el valor de la altura y duración actuales, y calcula el valor e la frecuencia de la nota actual, multiplicando el valor de la nota base, **giBasFreq** , por el factor correspondiente al semitono actual. A continuación se utiliza la fórmula

$$h = round(69 + 12log_2[f/440]$$

para calcular el valor MIDI de dicha frecuencia, y se va incrementando un nuevo vector de histograma, que permitirá una comparación entre el entrenamiento y el resultado y que se almacena en una tabla y se guarda en un archivo de texto. La última acción de este instrumento consiste en generar los mensajes MIDI de encendido y apagado de nota con la altura y duración actuales. Para ello se utiliza el comando **noteondur2**, que recibe los parámetros del canal midi que se utilizara (al ser melodias simples se utilizará el canal 1), la altura MIDI actual, la velocidad (al no haber tratado la dinámica en el proyecto todas las notas se generarán con la misma intensidad) y la duración actual.

Por último se utiliza un generador de sonido simple, con un paneo y filtro aleatorios, utilizando los comandos **mpulse** y **mode** para generar el sonido que se envía a la salida (altavoces en ejecución normal o archivo WAV en ejecución renderizada).

Finalmente, cuanto a la partitura de este subsistema, contiene una llamada al instrumento *inicio* y seguidamente una llamada al instrumento *trigger-note* con la duración deseada (en segundos) para la secuencia resultante.

i ïnicio"0 1

i "trigger-note"1 30

CAPÍTULO 4

Resultados

Para poder visualizar el funcionamiento del sistema se ejecutará con diversos ejemplos diferenciados y se utilizará el programa Anvil Studio para visualizar las partituras. Además se ha implementado un script en MATLAB que representa los histogramas del entrenamiento y del resultado de cada ejemplo.

Cabe destacar que antes de pasar de una prueba a la siguiente se debe cambiar la ubicación o eliminar los archivos de las tablas a fin de evitar que el programa utilice sus datos en las pruebas posteriores.

Como ejemplos de partituras para los diferentes entrenamientos se utilizarán algunas melodías del corpus *9GDB (9 genres database) del Departamento de Lenguajes y Sistemas Informáticos - Universidad de Alicante*.

4.1 Ejemplo 1. Entrenamiento con escala cromática ascendente.

Para el primer ejemplo se ha creado un archivo MIDI con una escala cromática descendente, para que todas las alturas tengan probabilidad de pasar únicamente a la siguiente. La duración de las notas se ha escogido de manera aleatoria.

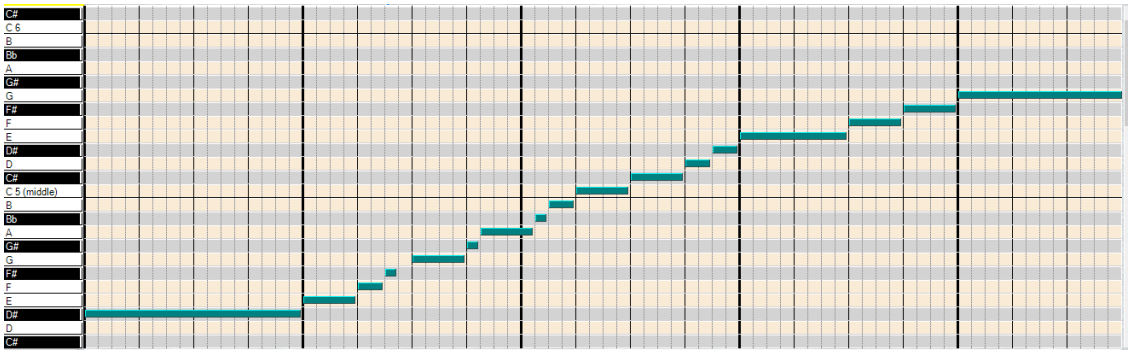

Figura 4.1: Sección de la partitura utilizada en el entrenamiento del Ejemplo 1.

En la partitura resultante se puede observar que la secuencia sigue el patrón ascendente del entrenamiento, aunque se centra en una única octava, cuando llega a la altura más alta vuelve al principio de la misma octava.

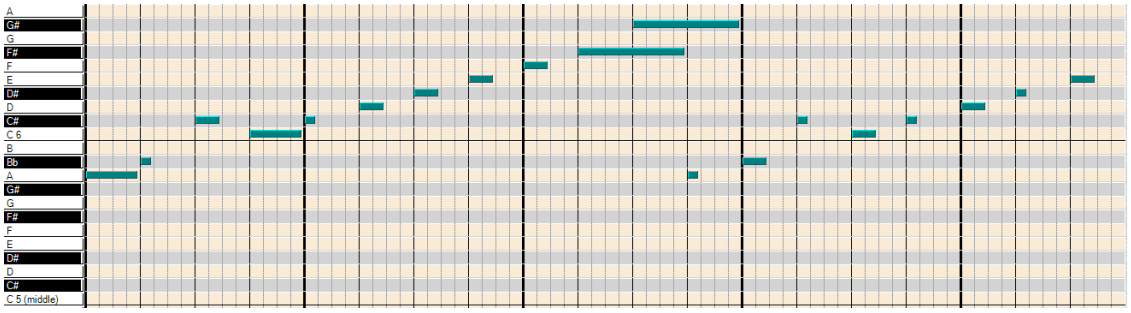

Figura 4.2: Sección de la partitura compuesta en el Ejemplo 1.

Esto puede observarse también en los histogramas. En el del entrenamiento se observa una distribución con menos repeticiones de cada nota a lo largo de dos octavas, con la predominancia de una nota respecto de las demás. En el resultado, sin embargo, se observan más repeticiones contenidas en una única octava, aunque también se puede ver la predominancia de una nota respecto de las demás.

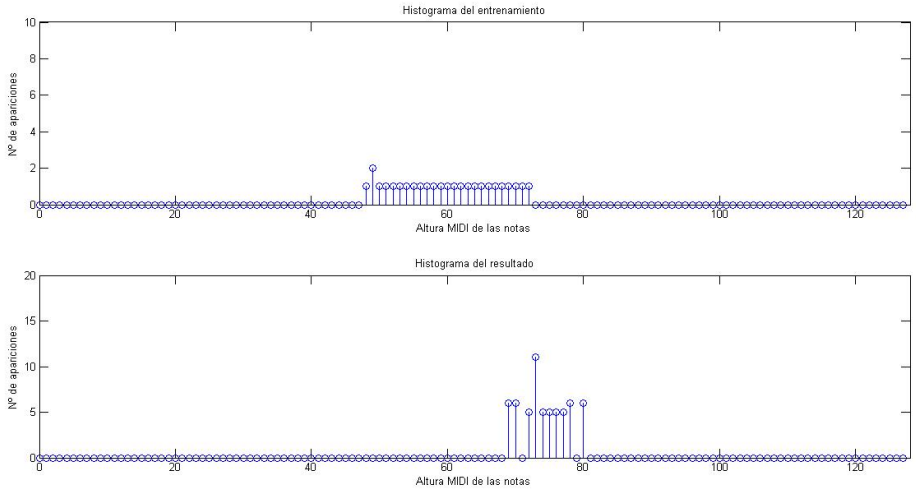

Figura 4.3: Histograma del Ejemplo 1.

Esta composición, aunque no tiene ningún valor artístico, permite destacar la utilidad del sistema implementado.

4.2 Ejemplo 2. Entrenamiento con una partitura con motivos musicales concretos.

Entrenamiento con una única partitura jazz bossanova, llamada *Eu Sei Que Vou Te Amar*, composición original de Antônio Carlos Jobim and Vinícius de Moraes. Esta melodía está caracterizada por contener un motivo melódico consistente en una serie de notas cortas (corcheas y semicorcheas) que preceden a una larga (blanca o redonda). Las notas cortas siguen una estructura triangular, esto es ascienden algunos semitonos y, seguidamente, descienden hasta alcanzar la altura inicial.

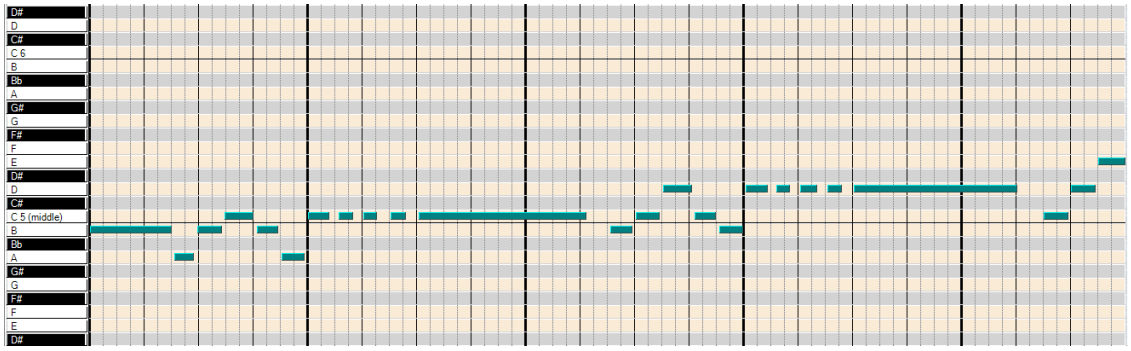

Figura 4.4: Sección de la partitura de entrenamiento del Ejemplo 2.

En este caso, en la melodía compuesta al principio se puede observar la presencia de algunas notas largas, sin embargo, cuando aparece una nota corta aparece una predominancia de notas cortas, lo cual es consecuencia de los motivos musicales comentados anteriormente, ya que es mucho más probable que una nota corta preceda a otra nota corta que a una larga. En lo que respecta a las alturas es posible observar que la composición sigue la estructura triangular comentada.

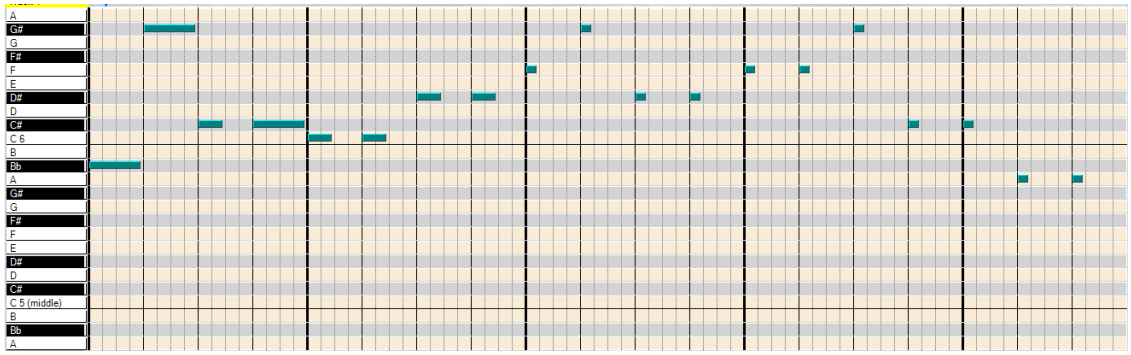

Figura 4.5: Sección de la partitura compuesta en el Ejemplo 2.

Si se observa el histograma, puede apreciarse que la forma es similar en ambos casos, y que la melodía resultante está centrada en una octava superior.

Este caso no puede considerarse de éxito total ya que, aunque se siguen las probabilidades generadas por el entrenamiento, el sistema no es capaz de detectar el motivo musical del ejemplo y en la melodía generada no existe presencia de notas largas.

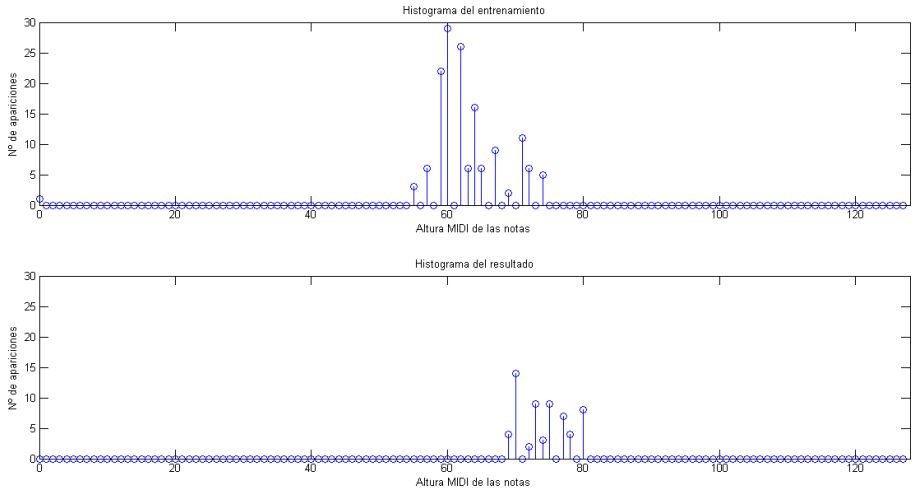

Figura 4.6: Histograma del Ejemplo 2.

4.3 Ejemplo 3: Entrenamiento con varias partituras.

Entrenamiento con 4 partituras de música popular mundialmente conocidas y tituladas:

- *'ll Be Watching You*,de The Police
- *You've Got A Friend*, de James Taylor
- *Arrow Through Me*, de Paul McArtney
- *Yesterday* de The Beatles.

En estas melodías existe una gran variedad de alturas y duraciones, contando con una gran cantidad de notas para el entrenamiento, muy superior a las cantidades manejadas en los ejemplos anteriores.

En este caso, como era de esperar, la composición resultante tiene diversidad de alturas y duraciones a causa de haber utilizado 4 composiciones relativamente largas y diferentes entre sí.

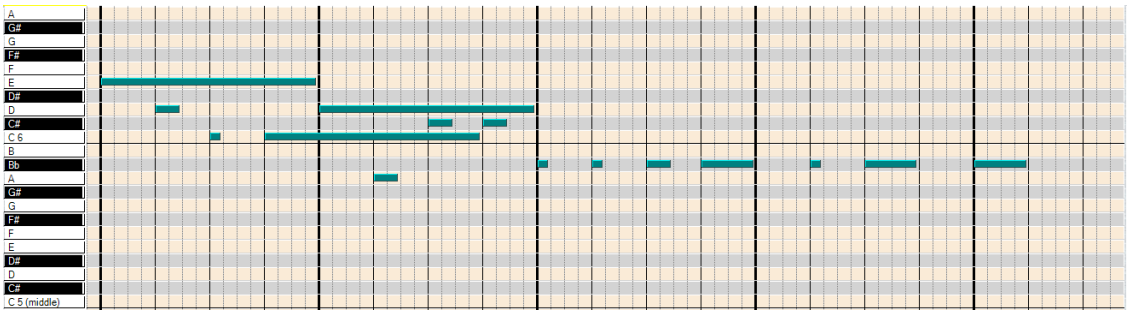

Figura 4.7: Sección de la partitura compuesta en el Ejemplo 3.

Si se observan los histogramas, en este caso parece que la forma del resultante es la mitad del original, esto puede deberse a que el resultado es una composición más corta y

además se centra únicamente en una octava. En ambos histogramas es posible observar la predominancia de una nota sobre las demás, así como un grupo de notas con presencia intermedia y finalmente otro grupo con presencia casi nula.

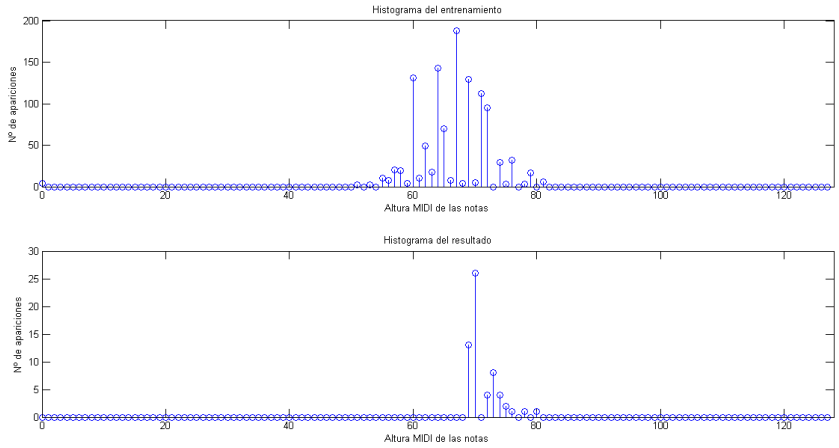

Figura 4.8: Histograma del Ejemplo 3.

CAPÍTULO 5
Conclusiones

5.1 Conclusiones generales

El objetivo principal del proyecto era estudiar las posibilidades de Csound para componer secuencias musicales utilizando cadenas de Markov.

Este objetivo ha sido superado, diseñando e implementando una herramienta que permite al usuario obtener una idea musical a partir de la cual desarrollar su trabajo compositivo, en un entorno como Csound, que permite una interacción bastante simple para aquellos compositores que tienen algunos conocimientos informáticos básicos.

El sistema implementado aplica el proceso generativo de Markov a dos parámetros de una nota musical, su altura y su duración y, además, ofrece al usuario la posibilidad de escuchar la composición en tiempo real o guardarla en un archivo WAV, además de almacenar un archivo MIDI con la composición, el cual puede ser abierto con cualquier secuenciador para su reproducción o visualización.

Una de las ideas preliminares del proyecto era implementar un entorno gráfico con el cual el usuario pudiera interactuar, modificando diferentes parámetros del sistema, sin embargo esta idea ha sido desechada ya que en el estado actual del sistema, el numero de parámetros que el usuario podría modificar son prácticamente nulos, únicamente la altura y duración iniciales de la secuencia generada, las cuales se ha optado por generar de manera aleatoria.

En conclusión, es razonable considerar el proyecto como exitoso ya que se ha conseguido alcanzar el objetivo principal, además de dejar encaminadas algunas posibles mejoras para hacer más atractivo el sistema diseñado.

5.2 Desarrollo del proyecto

Para el desarrollo del proyecto se ha seguido el sistema clásico de desarrollo de software, basado en las siguientes fases: documentación, análisis, diseño, programación y depuración.

Empezando por la consulta de materiales teóricos relacionados con las cadenas de Markov, así como la consulta de los manuales de Csound, especialmente el **Floss Manual** y **The Canonical Csound Reference Manual**, así como los ejemplos incluidos en CsoundQT.

Posteriormente se diseñó la estructura del sistema, conformado por las partes de entrenamiento y composición de la secuencia.

Y, finalmente, se procedió a la implementación y posterior depurado del código utilizando el entorno de programación CsoundQT.

5.3 Propuestas de desarrollo futuro

A lo largo del desarrollo del proyecto han ido surgiendo algunas ideas cuya implementación finalmente no ha podido realizarse, sin embargo podrían ser un buen punto de partida de cara a mejorar el sistema diseñado.

En primer lugar, ampliar el análisis de duraciones ya que en el estado actual del proyecto únicamente se tienen en cuenta 5 duraciones básicas por simplicidad, pero sería interesante una ampliación en este sentido para poder tener duraciones intermedias y más variedad, como pueden ser tresillos y notas más cortas o más largas, así como la posibilidad de trabajar con silencios.

También sería interesante implementar el análisis de dinámica para poder ejecutar el proceso de Markov también con este parámetro, dando aún más versatilidad al sistema.

Otra posible mejora consistiría en implementar funcionalidad con diferentes melodías simultáneas, permitiendo el análisis de archivos MIDI polifónicos y generando una orquesta con diferentes instrumentos.

Finalmente, se podrían introducir más variables que el usuario pudiese controlar, por ejemplo los parámetros iniciales de la composición o la duración de la melodía generada; y además implementar una interfaz gráfica para facilitar el uso al usuario,

Bibliografía

[1] Iñesta,J.M. *Apuntes de apoyo. Síntesis Digital del Sonido.* Dept. Lenguajes y Sistemas Informáticos, Universidad de Alicante (2016).

[2] Miranda,E. *Composing Music with Computers.* CRC Press (2001).

[3] Roads,C. *The Computer Music Tutorial* (1996)

[4] Kohan, D. et al. Material de estudio de la asignatura Métodos Estadísticos en Ciencias de la Vida. Unidad Nº6 .Universidad Nacional de Entre Ríos. Consultado en http://www.bioingenieria.edu.ar/academica/catedras/metestad/Cadenas%20de%20Markov-1.pdf.

[5] Lazzarini, V. *The development of computer music programming systems.* Journal of New Music Research 42(1), 97–110 (2013)

[6] Lazzarini, V. et al. *Csound. A Sound and Music Computing System* , 3-15 (2016)

[7] Manual de Csound de Floss Manuals. Consultado en http://write.flossmanuals.net/csound/preface/

[8] The Canonical Csound Reference Manual. Consultado en http://www.csounds.com/manual/html/