Capítulo 3

Cómputos de audio y de control

3.1 El teorema del muestreo

Hemos hecho una discusión de las señales de audio como si ellas fueran capaces de describir cualquier función en el tiempo, como si sabiendo los valores que la función tiene en los números enteros, pudiéramos determinar de alguna manera los valores que toma entre éstos. Esto no es realmente cierto. Por ejemplo, suponga alguna función f (definida para los números reales) que tiene valor uno para todos los enteros:

$$f(n) = 1,\ n = ...,-1,0,1,...$$

Podríamos creer que $f(t) = 1$ para todos los reales t. Pero sin embargo, f es uno para los enteros y cero para todo lo demás -esa es una función perfectamente buena también, y nada de los valores de la función en los números enteros la distingue de la más simple $f(t) = 1$. Pero la intuición nos dice que la función constante está en el *espíritu* de las señales de audio digital, visto que esconde un secreto entre las muestras. Una función que "se deja tomar en muestras" debería ser aquella para la cual podemos utilizar un esquema de interpolación razonable para deducir sus valores en los no-enteros, partiendo de los valores en sus enteros.

Se acostumbra en este punto de las discusiones de la música por computador invocar el famoso *Teorema de Nyquist*. Este dice (de manera resumida) que si una función es una combinación finita o infinita de sinusoides, ninguna de cuyas frecuencias angulares excede π, entonces, por lo menos teóricamente, está totalmente determinada por los valores de la función en los enteros. Una forma posible de reconstruir la función podría ser como un límite del mayor y el más alto orden polinomial para la interpolación.

La frecuencia angular π llamada la *Frecuencia de Nyquist* corresponde a $R/2$ ciclos por segundo si R es la velocidad de las muestras. El período correspondiente es de dos muestras. La frecuencia de Nyquist es lo mejor que podemos hacer en el sentido que cualquier sinusoide real de frecuencia más alta, es igual en los enteros, a aquella cuya frecuencia es menor que la de Nyquist, y es esta la frecuencia más baja que tendremos para la reconstrucción con el proceso de interpolación ideal. Por ejemplo, una sinusoide con frecuencia angular entre π y 2π, por decir $\pi + \omega$, puede ser escrita como

$$\begin{aligned}&\cos((\pi + \omega)n + \phi) = \cos((\pi + \omega)n + \phi - 2\pi n)\\ &= \cos((\omega - \pi)n + \phi)\\ &= \cos((\pi - \omega)n - \phi)\end{aligned}$$

para todos los enteros n. (Si n no fuera un entero, el primer paso debería fallar.) Así, una sinusoide con frecuencia entre π y 2π, es igual, por lo menos en lo que se refiere a los números enteros, a otra con frecuencia entre 0 y π; usted simplemente no puede tener las dos aparte. Y ya que cualquier operación de conversión en el equipo de cómputo deberá hacer lo "correcto" y reconstruir la sinusoide de más baja frecuencia, cualquiera que usted trate de sintetizar con frecuencia más alta llegará a sus parlantes con la frecuencia errada -específicamente usted escuchará la frecuencia única entre 0 y π a la que cae la frecuencia en la forma descrita anteriormente. Este fenómeno es llamado *foldover* <sobre-doblado>, debido a que la media línea de frecuencias de 0 a ∞ se dobla de acá para allá, en longitudes de π sobre el intervalo de 0 a π. La palabra *aliasing* <alias> significa la misma cosa. La figura 3.1 muestra que las sinusoides de frecuencias angulares $\pi/2$ y $3\pi/2$, no se pueden distinguir una de

la otra al ser leídas como señales de audio digital.

Concluimos entonces que cuando, por ejemplo, estamos computando valores de una seriede Fourier (página “12”), ya sea como una tabla de onda o como una señal en tiempo real, mejor dejamos por fuera cualquier sinusoide de la suma, cuya frecuencia exceda π. Pero el cuadro en general no es así de simple, ya que la mayoría de la técnicas diferentes a la de la síntesis aditiva no son señales limitadas por bandas ordenadas (aquellas cuyos componentes paran a alguna frecuencia limitada). Por ejemplo una onda diente de sierra de frecuencia ω de la forma puesta por el objeto phasor~ de Pd, pero que considerada como una función continua $f(t)$, se expande a:

$$f(t) = 1/2 - 1/\pi(\text{sen}(\omega t) + \text{sen}(2\omega t)/2 + \text{sen}(3\omega t)/3 + \ldots)$$

la cual goza arbitrariamente de altas frecuencias; y más aún el parcial cien es únicamente 40 dB más débil que el primero. A cualquier valor muy bajo de ω, los parciales por encima de π se escucharán presentes -y, debido al sobre-doblado, se escucharán a las frecuencias incorrectas. (Esto no significa que no se deberían utilizar ondas diente de sierra como generadores de fase -la lectura a pasos de la tabla de onda corrige mágicamente el sobre-doblado de la diente de sierra- pero uno debería pensarlo dos veces antes de utilizar una onda diente de sierra como fuente de sonido digital.)

Muchas técnicas de síntesis, incluso si no están estrictamente limitadas en sus bandas, dan parciales que pueden ser apagados más rápidamente que $1/n$ como en el ejemplo de la diente de sierra, y de esta manera son más apropiadas trabajarlas digitalmente. En cualquier caso, siempre es buena idea mantener en la mente la posibilidad del sobre-doblado, y entrenar sus oídos para reconocerlo.

La primera línea de defensa contra el sobre-doblado es simplemente utilizar velocidades altas de muestreo; es una buena práctica utilizar sistemáticamente las velocidades de muestreo más altas que pueda manejar su computador con facilidad. La velocidad práctica más alta variará de acuerdo a si lo que sea que está trabajando en el computador es en tiempo real o no, las limitantes de tiempo y de memoria de la CPU, y/o los dispositivos de entrada y salida, y a veces incluso las limitaciones impuestas por los programas de cómputo.

Un tratamiento no muy técnico de la teoría del muestreo se da en [Bal03]. Más detalles se pueden encontrar en [Mat69, págs.1-30].

3.2 Control

Ya hemos discutido acerca de las señales de audio digital, las cuales son sólo secuencias $x[n]$ definidas para enteros n, los cuales corresponden a puntos espaciados regularmente en el tiempo. Este puede ser un esquema para describir las técnicas de síntesis, pero las aplicaciones reales de música electrónica usualmente suponen otros cómputos que deben ser realizados en puntos irregulares en el tiempo. En esta sección desarrollaremos un marco para describir lo que llamaremos cómputos de control. Siempre requeriremos que cualquier cómputo se corresponda con un específico *tiempo lógico*. El tiempo lógico controla cuál muestra de audio irá en primer lugar a la salida para reflejar el resultado de los cómputos.

En un sistema de tiempo no real (tal como Csound en su forma clásica), esto significa que el tiempo lógico procede desde cero hasta la longitud de la salida del archivo de sonido. Cada “tarjeta de puntuación” tiene un tiempo lógico asociado (el tiempo en la puntuación), y actúa una vez el cómputo del audio ha alcanzado ese tiempo. Así los cálculos de audio y de control (la salida de las muestras y el manejo de las tarjetas de notas) se manejan cada uno por turnos, incrementándose con todo el tiempo lógico.

En un sistema de tiempo real, el tiempo lógico, el cual corresponde todavía al tiempo para la salida de la muestra siguiente de audio afectada, está siempre ligeramente adelantado al *tiempo real*, el cual está medido por la muestra que sale en ese momento del computador. Los cómputos de control y de audio son llevados a cabo de manera alterna, seleccionados por el tiempo lógico.

La razón para utilizar el tiempo lógico y no el tiempo real en los cálculos de la música por computador es mantener los cómputos independientes del tiempo de ejecución del computador, el cual puede variar por una cantidad de razones, incluso para dos cálculos aparentemente idénticos. Cuando estamos haciendo el cálculo de un nuevo valor para la señal de audio o procesando alguna entrada de control, el tiempo real puede pasar, pero nosotros requerimos que el tiempo lógico sea el mismo a través del cálculo completo, como si tomara lugar instantáneamente. Como resultado de esto, los cómputos en la música electrónica, si se han hecho de la manera correcta, son determinísticos: dos corridas de los mismos cálculos del mismo audio en tiempo real o no real, teniendo cada una las mismas entradas, deberán tener resultados idénticos.

La figura 3.2 (parte a) muestra esquemáticamente cómo están alineados el tiempo lógico y los cómputos de las muestras. Las muestras de audio están calculadas en períodos regulares (marcados por líneas onduladas), pero antes del cálculo de cada muestra hacemos todos los cálculos de control que podrían afectarla (marcados con segmentos de línea rectos). Hacemos primero los cómputos de control asociados con los tiempos lógicos empezando de cero hasta uno, pero sin incluir ese valor; entonces computamos la primera muestra de audio (de índice cero) en el tiempo lógico uno. Hacemos luego todos los cálculos de control hasta el tiempo lógico 2, pero sin incluir ese valor, luego la muestra de índice uno, y así sucesivamente. (Adoptamos aquí ciertas convenciones para el etiquetado que podrían haber sido escogidas de manera diferente. Por ejemplo, no hay razón fundamental para que el control debe ilustrarse entrando “antes” que los cómputos de audio pero es más fácil pensarlo de esa manera.)

La parte (b) de la figura muestra la situación cuando queremos computar la salida del audio en bloques de más de una muestra a la vez. Utilizando la variable B para denotar el número de elementos en un bloque (en la figura B = 4), el primer cálculo de audio tendrá com salida las muestras de audio 0,1, ...$B - 1$ todas a la vez en un bloque computado en el tiempo lógico B. Debemos realizar los cómputos de control relevantes para todos los períodos de tiempo B, con anticipación. Hay un retraso de las muestras B entre el tiempo lógico y la aparición del audio en la salida.

La mayor parte de los programas de música calculan el audio por bloques. Esto se hace para incrementar la eficiencia de las operaciones de audio individuales (tal como lo hacen las unidades generadoras de Csound y Max/MSP y los objetos tilde de Pd). La unidad generadora o el objeto tilde incurren en gasto cada vez que se hacen funcionar, igual a aproximadamente veinte veces el costo del cómputo de una muestra en promedio. Si el tamaño del bloque es uno, esto significa un gasto de 2000%; si es sesenta y cuatro (como lo es en Pd por defecto), el gasto es de sólo el 30%.

3.3 Secuencias de control

Los cálculos de control pueden provenir de una variedad de fuentes, tanto internas como externas, para el cómputo general. Los ejemplos de los cálculos de control engendrados internamente incluyen secuenciación (cuyos cálculos de control pueden tener lugar en tiempos predeterminados) o la detección de la salida del audio (casos de búsqueda de ceros cruzando en una señal). Los engendrados externamente pueden venir de aparatos tales como controladores MIDI, el ratón y el teclado, paquetes de red y otros. En cualquier caso, los cálculos de control pueden ocurrir a intervalos irregulares, a diferencia de las muestras de audio las cuales corresponden a una marca fija de reloj de muestras.

Necesitaremos una manera para describir cómo fluye la información entre los cómputos de control y de audio, para lo cual nos basaremos en la noción de una *secuencia de control*. Esta es simplemente un conjunto de números -posiblemente vacía- que aparece como un resultado de los cálculos de control, ya sea regular o irregularmente espaciada en el tiempo lógico. La secuencia de control más simple posible no tiene otra información que una *secuencia de tiempo*:

$$...,t[0],t[1],t[2],...$$

Aunque es mejor dar los valores de tiempo en muestras, sus valores no están cuantizados; ellos pueden ser números reales arbitrarios. Requerimos que estén organizados en orden no decreciente:

$$... \leq t[0] \leq t\ [1] \leq t[2] \leq ...$$

Cada ítem de la secuencia es llamado un *evento*.

Las secuencias de control se pueden mostrar gráficamente como aparecen en la figura 3.3. Una línea numerada muestra el tiempo y una secuencia de flechas señala los puntos de tiempo asociados con cada evento. La secuencia de control que se muestra no tiene datos (es una secuencia de tiempo). Si queremos mostrar datos en la secuencia de control los escribiremos en la base de cada flecha.

Una secuencia de control numérica es la que contiene un número por cada punto en el tiempo, de tal manera que aparece como una secuencia de pares ordenados:

$$...,(t[0],x[0]),\ (t[1],x[1]),\ ...$$

donde los $t[n]$ son los puntos en el tiempo y los $x[n]$ son los valores de la señal en esos tiempos.

Una secuencia de control de números es casi análoga a un “controlador MIDI”, cuyos valores cambian irregularmente, por ejemplo cuando un control físico se mueve por parte de un ejecutante. Otras fuentes posibles de secuencias de control pueden tener las velocidades de cambio más altas posibles y/o mayor precisión. De otro lado, una secuencia de tiempo podría ser una secuencia de golpes de pedal, los cuales (no obstante la implementación MIDI) no deberían considerarse conteniendo *valores*, si no *tiempos*.

Las secuencias de control numéricas son como señales de audio, pues ambas son sólo valores numéricos variando en el tiempo. Pero toda vez que las señales audio van a una velocidad constante (y así los valores de tiempo no requieren estar especificados para cada muestra), las secuencias de control son impredecibles -a veces son parejas, a veces disparejas, a veces no aparecen.

Vamos a mostrar lo que sucede cuando tratamos de convertir una secuencia de control numérico en una señal de audio. Como antes, escogeremos un bloque de tamaño $B = 4$. Consideraremos como una secuencia de control una onda cuadrada de período 5.5:

$$(2,1),(4.75,0),(7.5,1),(10.25,0),(13,1),...$$

y demostramos tres maneras en que podría convertirse en señal de audio. La figura 3.4 (parte a) muestra la conversión más simple y más-rápida-posible. Cada muestra de audio a su salida refleja simplemente el valor más reciente de la señal de control. De esta manera las muestras de 0 a 3 (las cuales son computadas en el tiempo lógico 4 debido al tamaño del bloque) son de valor 1 según el punto (2,1). Las cuatro muestras siguientes también son uno ya que de los dos puntos, (4.75,0) y (7.5,1), el más reciente tiene también el valor 1.

La conversión más-rápida-posible es la más apropiada para controlar secuencias

que no cambian con frecuencia, en comparación al tamaño del bloque. Sus principales ventajas son la simplicidad para el cálculo y la respuesta más rápida posible a los cambios. Según se ve en la figura, cuando los cambios de la secuencia de control son muy rápidos (cercanos al orden del tamaño del bloque), la señal de audio pueden no tener mucha semejanza con la esporádica. (Si, como en este caso, la secuencia de control viene en intervalos de tiempo regulares, podemos utilizar el teorema del sampleo para analizar el resultado. Aquí, la frecuencia de Nyquist asociada con la velocidad del bloque R/B es menor que la frecuencia de entrada de la onda cuadrada, y así la salida produce un alias a una nueva frecuencia que es más baja que la frecuencia de Nyquist.)

La parte (b) muestra el resultado de la conversión de la muestra-más-cercana. Cada nuevo valor de la secuencia de control en el tiempo t afecta la salida de las muestras comenzando desde el índice $\lfloor t \rfloor$ (el entero más grande que no exceda a t). Este es el equivalente a usar la conversión más-rápida-posible a un tamaño de bloque de 1; en otras palabras, la conversión de la muestra-más-cercana esconde el efecto de los tamaños de bloque más grandes. Esta es mejor que la conversión más-rápida-posible en los casos en los que la secuencia de control podría cambar rápidamente.

La parte (c) muestra la conversión esporádica-a-audio, de nuevo a la muestra más cercana, pero ahora utilizando también la interpolación de dos puntos para un incremento adicional en la precisión del tiempo. Conceptualmente podemos describir esto como sigue. Suponga que el valor de la secuencia de control fue al final igual a x, y que el punto siguiente es $(n + f, y)$, donde n es un entero y f es la parte fraccional de valor del tiempo (así $0 \le f < 1$). El primer punto afectado en la salida del audio será la muestra en el índice n. Pero en lugar de ajustar la salida a y como antes, ajustamos esta a

$$fx + (1 - f)y$$

en otras palabras, a una aproximación ponderada del valor previo y del valor nuevo, cuyos pesos favorecen más al nuevo valor si el tiempo del valor esporádico es anticipado, más cercano a n. En el ejemplo que se muestra, la transición de 0 a 1 en el tiempo 2 da

$$0 \cdot x + 1 \cdot y = 1,$$

mientras la transición de 1 a 0 en el tiempo 4.75 da:

$$0.75 \cdot x + 0.25 \cdot y = 0.75$$

Esta técnica proporciona una representación todavía más cercana de la señal de control (por lo menos la porción que queda por debajo de la frecuencia de Nyquist), a expensas de mayor cómputo y un retraso ligeramente mayor.

Las secuencias de control numérico pueden convertirse también a señales de audio utilizando funciones rampa para suavizar las discontinuidades. Se utilizan con frecuencia cuando una de estas secuencias debe controlar una amplitud, como se describió en la sección 1.5. En general hay tres valores para especificar el ajuste de una función rampa en movimiento: el tiempo de inicio y el valor al que se quiere llegar (especificado por la secuencia de control) y el tiempo para llegar al valor, que usualmente se expresa como un retraso después del tiempo de inicio.

En tales situaciones es casi siempre suficientemente seguro ajustar los tiempos del comienzo y del final para que coincida con la primera muestra de audio computada en un tiempo lógico posterior, una opción que corresponde al escenario de más-rápida-posible anterior. La figura 3.5 (parte a) muestra el efecto de la rampa desde 0, comenzando en el tiempo 3, a un valor de 1 en el tiempo 9,

comenzando el regreso de inmediato a 0 en el tiempo 15, con un tamaño de bloque $B = 4$. Los tiempos 3, 9 y 15 están truncados a 0, 8 y 12 respectivamente.

En situaciones reales el tamaño del bloque podría ser del orden de un milisegundo, y el ajuste de los puntos finales de las rampas con las fronteras del bloque funciona bien para controlar las amplitudes; alcanzar un objetivo en una fracción de milisegundo antes o después raramente hace audible una diferencia. Sin embargo otros usos de las rampas son más sensibles a la cuantización del tiempo en los extremos. Por ejemplo si queremos hacer algo repetitivo de unos pocos milisegundos, la variación en el segmento hará audible una aperiodicidad.

Para situaciones tales como esta, podemos mejorar el algoritmo de generación de la rampa para iniciar y detener en muestras arbitrarias, como se muestra en la figura 3.5 (parte b), por ejemplo. Aquí los extremos de los segmentos de línea coinciden exactamente con las muestras requeridas 3, 9 y 15. Podemos ir incluso más allá y ajustar para muestras fraccionadas, haciendo que los segmentos de línea toquen los valores 0 y 1 en puntos específicos sobre la línea de números.

Por ejemplo, suponga que queremos repetir un sonido grabado de una tabla de onda 100 veces por segundo, cada 441 muestras a la velocidad de muestreo usual. Los errores de redondeo debidos a los bloques de 64 muestras podrían desafinar la reproducción en casi un tono completo, e incluso redondear a fronteras de una muestra puede introducir variaciones de cerca del 2%, o tres cents. Esta situación debería llevar a la exactitud de sub-muestras en la conversión esporádica-a-audio.

3.4 Conviertiendo de señales de audio a secuencias de control numérico

A veces necesitamos hacer la conversión en otra dirección, de una señal de audio a la señal esporádica. Para ir en esa dirección, de alguna manera proveemos una serie de tiempos lógicos (una secuencia de tiempo), así como una señal de audio. A la salida queremos una secuencia de control que combine con los valores tomados de la señal de audio. Hacemos esto cuando queremos incorporar el valor de la señal como parte de los cómputos de control.

Por ejemplo, podríamos estar controlando la amplitud de una señal utilizando el objeto `line~` como en el ejemplo A03.line.pd (página “21”). Suponga que queremos apagar el sonido a una velocidad fija, en lugar de hacerlo en un tiempo fijo. De esta manera podríamos querer re-utilizar la red para otro sonido y querer mutearlo tan rápido como fuera posible sin artefactos audibles; podemos probablemente hacerlo en forma de rampa a cero en menos tiempo si la amplitud en ese instante es más baja. Para hacer esto debemos confeccionar un mensaje en el objeto `line~` para enviarlo a cero en una cantidad de tiempo que calcularemos con base en el valor de su salida en ese momento. Esto requiere antes que nada, que “sampleemos” la salida del objeto `line~` (una señal de audio) a una secuencia de control.

El mismo asunto del retraso de tiempo y exactitud aparece también para la conversión de datos esporádicos a audio. De nuevo, habrá un compromiso entre inmediatez y exactitud.

Suponga como antes que estamos calculando en bloques de audio de 4 muestras, y suponga que en el tiempo lógico 6 queremos buscar el valor de una señal de audio, y utilizarla para cambiar el valor de otra. Como se muestra en la figura 3.2 (parte b), el valor más recientemente calculado de la señal será para el índice 3, y el primer índice al cual nuestros cálculos pueden afectar la señal es 4. Podemos de esta manera llevar el asunto completo con un retraso de únicamente una muestra. Sin embargo, no podemos escoger exactamente cuál muestra -la actualización sólo puede ocurrir en las fronteras del bloque.

Como antes, podemos negociar inmediatez por mayor exactitud en el tiempo. Si

esto sucede exactamente en la muestra en la que llevamos el cómputo audio-a-control-a-audio, leemos la muestra del índice 2 y la actualizamos en el índice 6. Entonces, si queremos hacer la misma cosa de nuevo en el tiempo lógico 7, leemos desde el índice 3 y actualizamos en el índice 7 y así sucesivamente. En general, si el tamaño del bloque es *B*, y para cualquier índice *n*, podemos leer siempre la muestra en el índice *n* - *B* y afectarla en el índice *n*. Existe así un retraso de *B* muestras en el cómputo audio a control a audio, que es el precio en el que se incurre por ser capaces de nombrar a *n* exactamente.

Si queremos ir más allá, para ser capaces de especificar la fracción de una muestra, entonces (como antes) podemos utilizar interpolación -con un ligero incremento en el retraso. En general, como en el caso de la conversión esporádica-a-audio, en la mayoría de los casos la solución más simple es la mejor, pero ocasionalmente tenemos que hacer trabajo extra.

3.5 Secuencias de control en diagramas de bloque

La figura 3.6 muestra cómo las secuencias de control se expresan en diagramas de bloque, utilizando la conversión control-a-señal y señal-a-control como ejemplos. Las secuencias de control están representadas utilizando puntos (opuesto a las señales de audio que aparecen como flechas sólidas).

El bloque *señal* <*signal*> convierte de una secuencia de control numérico a una señal de audio. El tipo exacto de conversión no se especifica a este nivel de detalle; en los ejemplos Pd el tipo de operador de la conversión determinará esto.

El bloque *instantánea* <*snapshot*> convierte la señal de audio de nuevo a secuencia de control numérico. Adicional a la señal de audio, se requiere una señal de control separada para especificar la secuencia de tiempo a la cual se toman las muestras de la señal audio.

3.6 Detección de evento

Además de tomar instantáneas, un segundo modo de pasar información de señal de audio a cómputos de control es la *detección de evento*. Aquí derivamos la información de tiempo de la señal de audio. Un ejemplo es la *detección de umbral*, en la cual la entrada es una señal de audio y la salida es una secuencia de tiempo. Consideraremos el ejemplo de detección de umbral con algún detalle aquí.

Una razón típica para usar la detección de umbral es encontrar cuándo inicia algún tipo de actividad y cuándo se detiene, tal como un ejecutante tocando un instrumento. Supondremos que ya hemos hecho una medición continua de una actividad en forma de una señal de audio. (Esto puede ser hecho por ejemplo, utilizando un seguidor de envolvente). Lo que queremos es un par de secuencias de tiempo, una que marque el tiempo de inicio de la actividad y otra marcando la parada.

La figura 3.7 (parte a) muestra una realización simple de esta idea. Asumimos que la señal de entrada es como se muestra en la gráfica continua. Una línea horizontal marca el valor constante del umbral. La secuencia de tiempo marcada “onset” <“encendido”> contiene un evento cada vez que la señal cruza el umbral de abajo hacia arriba; la que está marcada como “turnoff” <“apagado”> marca los cruces en la dirección contraria.

En muchas ocasiones tendremos encendidos y apagados indeseables causados por pequeños rizos en la señal cercanos al umbral. Esto se evita con *deboucing*, <*apagado del rebote*> el cual puede ser hecho por lo menos de dos maneras. En la primera, como se muestra en la parte (b) de la figura, podemos ajustar dos umbrales: uno alto para marcar los encendidos y otro bajo para los apagados. En este esquema la regla es que únicamente reportamos el primer encendido después

de cada apagado y, viceversa, únicamente reportamos un apagado después de cada encendido. Así, la tercera vez que la señal cruza el umbral alto en la figura no reporta encendido, debido a que no se ha apagado desde la vez anterior. (Al iniciar, actuamos como si la salida más reciente hubiera sido de apagado, de tal manera que el primer encendido sí se reporta.)

Una segunda aproximación al filtrado de múltiples encendidos y apagados, se muestra en la parte (c) de la figura, es asociar un *período muerto* a cada encendido. Este es un intervalo de tiempo constante que sucede después de cada encendido reportado, durante el cual rechazamos el reporte de más encendidos y apagados. Luego de que el período finaliza, si la señal ha caído por debajo del umbral en el transcurso de ese tiempo, reportamos de manera tardía un apagado. Los períodos muertos se pueden asociar también con los apagados, y los dos períodos pueden tener valores diferentes.

Las dos estrategias de filtrado pueden utilizarse separada o simultáneamente. Es usualmente necesario adaptar los valores de umbral y/o los tiempos muertos manualmente para cada situación específica en la cual se usa el umbral.

El umbral a veces se utiliza com el primer paso en el diseño de estrategias de alto nivel para arreglar la respuesta de los computadores a las entradas audibles de ejecutantes musicales. Un ejemplo sencillo podría ser el de detener una secuencia de procesos pre-planificados, cada detención provocada por un encendido de sonido después de un período específico de relativo silencio, tal como lo podríamos ver si un músico ejecutara una secuencia de frases separadas por silencios.

Detectores más sofisticados (construidos por lo mejor de la detección de umbral) podrían detectar sonido o silencio continuo en un rango esperado de duraciones, o secuencias rápidas de alternancia entre tocar y no tocar, o períodos de tiempo en los cuales el porcentaje de tiempo de ejecución con respecto al de silencios está por encima o por debajo de un umbral, o muchas otras características. Estas podrían detener reacciones predeterminadas o figurar en una improvisación.

3.7 Señales de audio como control

De la tradición de la síntesis análoga viene una elegante y antigua aproximación para los problemas de control, que puede utilizarse como una alternativa para las secuencias de control que hemos estado tratado en este capítulo. En lugar o adicional al uso de las secuencias de control, podemos utilizar las mismas señales de audio para controlar la producción de otras señales de audio. Dos técnicas específicas de la síntesis análoga se prestan bien para este tratamiento: la secuenciación análoga y la muestra-y-retención.

El secuenciador análogo [Str95, págs.70-79][Cha80, págs.93,304-308] era utilizado con frecuencia para lograr una secuencia de sonidos repetida regular o semi-regularmente. La secuencia misma típicamente repite una secuencia de voltajes, junto con una señal de disparo la cual es pulsada cada que hay una transición entre los voltajes. Uno utilizaba los voltajes para tonos o para los parámetros tímbricos, y el disparador para controlar uno o más generadores de envolvente. Lograr secuencias enlazadas de valores predeterminados en audio digital es tan simple como el envío de un objeto `phasor~` a un lector de tabla no interpolada. Si usted quiere, por decir, cuatro valores en la secuencia, escale el valor de la salida de `phasor~` para tomar valores de 0 a 3.999... de tal manera que el primer cuarto de ciclo lea el punto 0 de la tabla y así sucesivamente.

Para disparar repetidamente, el primer paso es sintetizar otra diente de sierra que corra en sincronía con la salida de `phasor~`, pero cuatro veces más rápido. Esto se hace utilizando una variante de la técnica de la figura 2.8, en la cual utilizamos un operador de suma y un operador de envolvente para lograr el cambio de fase deseado. La figura 3.8 muestra el efecto de multiplicar una onda diente

de sierra por un entero, y luego envolverla para conseguir una diente de sierra a un múltiplo de la frecuencia original.

A partir de aquí es fácil conseguir una envolvente que se repita con lectura de tabla de onda, por ejemplo (utilizando un lector de tabla de onda esta vez, en lugar de una secuencia de voltajes). Toda la generación de la onda y las técnicas de alteración utilizadas para hacer sonidos a una cierta altura también pueden ser usados en este caso.

La otra técnica de control común derivada del control de los sintetizadores análogos es la unidad de muestra-y-retención [Str96, págs.80-83][Cha80, pág.92]. Se coge una señal de entrada, se toman ciertos valores instantáneos de ésta, y se "congelan" estos valores para su salida. Los valores particulares que se toman se seleccionan por una entrada de "disparo" secundaria. En puntos específicos del tiempo determinados por la entrada de disparo, se toma un valor de la entrada principal que es enviado a la salida de manera continua hasta el siguiente punto en el tiempo, cuando es reemplazado por un nuevo valor de la entrada principal.

En el audio digital es a veces útil tomar la muestra de un nuevo valor sobre los bordes de caída de la señal de disparo, es decir, cada vez que el valor corriente de la señal de disparo es más pequeño que el valor previo, según se ve en la figura 3.9. Esto es especialmente conveniente útil con un disparador diente de sierra, cuando queremos tomar muestras de señales en sincronía con un proceso conducido por oscilador. El objeto muestra-y-retención de Pd fue presentado previamente en el contexto del muestreo (ejemplo B08.sampler.nodoppler.pd, página "53").

3.8 Operaciones en secuencias de control

Hemos discutido cómo hacer la conversión entre secuencias de control y secuencias de audio. Adicional a esta posibilidad, existen cuatro tipo de operaciones que usted puede ejecutar sobre las secuencias de control para lograr otras secuencias de control. Estas operaciones con las secuencias de control no tienen operacones correspondientes en las señales de audio. Su existencia explica en gran parte porqué es útil introducir una estructura de control completa paralelamente con la de la señal de audio.

El primer tipo consiste en las operaciones de *retraso*, las cuales modifican los valores de tiempo asociados con una secuencia de control. En los sistemas de tiempo real los retrasos no pueden tener valores negativos. Una secuencia de control se puede retrasar en una cantidad constante o, alternativamente, usted pude retrasar cada evento de manera separada por diferentes cantidades.

Dos tipos de retraso se utilizan en la práctica: *simple* y *compuesto*. Ejemplos de cada uno de estos se muestran en la figura 3.10. Un retraso simple actúa sobre una secuencia de control programando cada evento, tal como viene, para un tiempo en el futuro. Sin embargo, si otro evento llega a la entrada antes de que el primer evento salga, el primer evento es olvidado en favor del segundo. En el retraso compuesto, cada evento a la entrada produce una salida, incluso si otros eventos llegan antes de que aparezca la salida.

Una segunda operación en las secuencias de control es la *mezcla*: toma dos secuencias de control y combina todos los eventos en uno nuevo. La figura 3.11 (parte a) muestra cómo esta y las otras operaciones son presentadas en diagramas de bloques.

La parte (b) de la figura muestra el efecto de mezcclar dos secuencias. Las secuencias pueden contener más de un evento a la vez. Si dos secuencias que se van a mezclar contienen eventos simultáneos, la secuencia mezclada los contiene a ambos, en un orden bien definido.

Un tercer tipo de operación sobre las secuencias de control es la *selección*. Selección en una secuencia de control significa buscar en los datos asociados y permitir pasar únicamente ciertos elementos. La parte (c) muestra un ejemplo en el cual los eventos (los cuales tienen cada uno un número asociado) pasan únicamente si el número es positivo.

Finalmente, está el concepto de resincronización de una secuencia de control con otra, como se muestra en la parte (d). Aquí una secuencia de control (la fuente) proporciona valores que son puestos en la secuencia de tiempo de otra secuencia (la sync). El valor dado a la salida es siempre el más reciente viniendo de la secuencia fuente. Note que cualquier evento de la fuente puede aparecer más de una vez (según se sugiere en la figura), o de otro lado, podría no aparecer.

De nuevo tenemos que considerar lo que sucede cuando las dos secuencias contienen cada una un evento al mismo tiempo. Debería la secuencia "sync" ser considerada como la que ocurre antes de la fuente (de tal manera que la salida proporciona el valor del evento previo de la fuente), o debería el evento de la fuente ser considerado como el primero de tal manera que su valor vaya a la salida al mismo tiempo? Cómo se podrá solucionar esto es una cuestión de diseño, por lo cual los diversos ambientes de los programas de computación toman diversas posiciones. (En Pd esto es controlado explícitamente por el usuario).

3.9 Operaciones de control en Pd

Hemos utilizado Pd principalmente para el procesamiento de señales de audio, aunque ya antes, como en la figura 1.10, hemos tenido que hacer la distinción entre la noción de señales de audio y de secuencias de control en Pd: las conexiones delgadas llevan secuencias de control y las gruesas llevan audio. Las secuencias de control en Pd aparecen como secuencias de *mensajes*. Los mensajes pueden contener datos (la mayoría usualmente, uno o más números) o no. Una *secuencia de control numérico* (sección 3.3) aparece como una conexión (delgada) que lleva números como mensajes.

Los mensajes sin datos hacen secuencias de tiempo. Así usted puede ver mensajes sin datos, que en Pd llevan el símbolo (arbitrario) de "bang".

Los cuatro tipos de operaciones de control descritos en la sección previa pueden ser expresados en Pd como se muestra en la figura 3.12. Los retrasos se logran utilizando dos objetos de retraso explícito:

`del`, `delay`: retraso simple. Usted puede especificar el tiempo de retraso con la creación de un argumento o por vía de la entrada derecha. Un "bang" en la entrada izquierda ajusta el retraso, el cual entonces da como salida un "bang" después de un retraso en milisegundos. El retraso es simple en el sentido de que envía un disparo a un retraso que ya ha sido ajustado y lo re-ajusta a un nuevo valor de tiempo, cancelando el que había sido programado con anterioridad.

`pipe`: retraso compuesto. Los mensajes que llegan a la entrada izquierda aparecen en la salida después del retraso especificado, el cual se ajusta por la creación del primer argumento. Si hay más creación de argumentos, ellos especifican una o más entradas para los datos numéricos o simbólicos que contendrán los mensajes. Cualquier número de mensajes puede ser almacenado por `pipe` de manera simultánea, y los mensajes pueden ser re-organizados para su salida, dependiendo de los tiempos de retraso dados para éstos.

La *mezcla* de secuencias de control en Pd se logra no con objetos explícitos, si no con el mismo mecanismo de conexiones de Pd. Esto se muestra en la parte (b) de la figura con cajas de números como un ejemplo. En general, siempre y cuando se haga más de una conexión para una entrada de control, las secuencias de control se mezclarán.

Pd ofece algunos objetos para la selección de secuencias de control, de los

cuales se muestran dos en la parte (c) de la figura:

moses: selecciona un rango numérico. Los mensajes numéricos que llegan a la entrada izquierda aparecerán en la salida izquierda si son más pequeños que un valor de umbral (ajustado en la creación de argumentos, o por la entrada de la derecha), y de lo conrario, salen por la entrada derecha.

select, sel: selecciona números específicos. Los mensajes numéricos que llegan a la entrada izquierda producen un "bang" en la salida únicamente si coinciden exactamente con un valor de prueba. El valor de prueba es ajustado ya sea por la creación de argumentos o por la entrada derecha.

Finalmente Pd se encarga de la resincronización de las secuencias de control implícitamente en sus mecanismos de conexión, según se ilustra en la parte (d) de la figura. La mayoría de los objetos con más de una entrada sincronizan todas las demás entradas partiendo de la entrada del extremo izquierdo. Así, el objeto float mostrado en la figura resincroniza su entrada del lado-derecho (la cual toma números) con la de su lado-izquierdo. Al enviar un "bang" a la entrada izquierda se tiene como salida el número float más reciente que ha sido recibido de antemano.

3.10 Ejemplos

Muestreo y sobre-doblado

El ejemplo C01.nyquist.pd (figura 3.13, parte a) muestra un oscilador ejecutando una tabla de onda barriendo frecuencias desde 500 hasta 1423. La tabla de onda consiste únicamente del parcial 46, el cual de esta manera varía de 23000 a 65458 Hertz <46*500=23000, 46*1423=65458>. Con una velocidad de lectura de muestras de 44100 estas dos frecuencias teóricamente suenan a 21100 y 21358 Hertz <44100-21000=23100, 44100-65458=-21358; la operación es una resta con respecto a 44100 = 2π>, pero en el barrido entre una y otra el doblado desciende, pasando por cero y luego asciende.

Las otras dos formas de onda están provistas para mostrar los interesantes efectos del batido entre parciales que, aunque "deberían" estar muy apartados, se encuentran vecinos ellos mismos a través del sobre-doblado. Por ejemplo, a 1423 Hertz el segundo armónico es 2846 mientras que el armónico 33 suena a 1423*33-44100 = 2859 Hertz -una fuerte disonancia.

Otros ejemplos menos extremos pueden producir sobre-doblado audible en formas menos bruscas. Usualmente sigue siendo desagradable, y vale la pena aprender a escucharlo. El ejemplo C02.sawtooth-foldover.pd (no ilustrado aquí) demuestra esto para una diente de sierra (el objeto phasor~). Para tablas de onda que contienen grabaciones de audio, el error de interpolación puede generar sobre-doblado extra. Los efectos de esto pueden variar ampliamente; el sonido a veces se describe como "crujiente" o como "salpicado", dependiendo de la grabación, la transposición y el algoritmo de interpolación.

Convirtiendo controles a señales

El ejemplo C03.zipper.noise.pd (figura 3.13, parte b) demuestra el efecto de convertir una secuencia de control actualizada lentamente a una señal de audio. Este ejemplo presenta un nuevo objeto:

line: un generador de rampa con control de salida. Al igual que line~, line toma pares de números (objetivo, tiempo) como parejas y ejecuta una rampa en dirección al objetivo, en la cantidad de tiempo dada; sin embargo, a diferencia de line~, la salida es una secuencia de control numérico, apareciendo, por defecto, en intervalos de 20 milisegundos.

En el ejemplo usted puede comparar el sonido al elevarse y al caer la amplitud

controlada por la salida de line con la controlada por la señal de audio generada por line~.

La salida de line es convertida a una señal de audio en la entrada del objeto *~. La conversión está implicada aquí al conectar una secuencia de control numérico a una entrada de señal. En Pd, las conversiones implícitas desde las secuencias de control numérico a las secuencias de audio están hechas con el modo más-rápida-posible mostrado en la figura 3.4 (parte a). La salida del objeto line se convierte en una señal en escalera con 50 escalones por segundo. El resultado se denomina comúnmente “ruido zipper”.

Mientras que las limitaciones del objeto line para generar señales de audio eran claramente audibles incluso sobre períodos de tiempo de 300 milisegundos, la variante de señal del objeto no produce problemas hasta períodos de tiempo que pueden ser mucho más cortos. El ejemplo C04.control.to.signal.pd (figura 3.13, parte c) demuestra el efecto de utilizar line~ para generar una onda triangular de 250 Hertz. Aquí los efectos mostrados en la figura 3.5 vienen al caso. Ya que line~ siempre alinea segmentos de línea con las fronteras de los bloques, la duración exacta de los segmentos de línea varía, y en este ejemplo la variación (del orden de milisegundos) es una fracción significante de su longitud.

Un objeto más preciso (y más costoso, en términos del tiempo de computación) se provee para estas situaciones:

vline~: generador de segmento de línea exacto. Este tercer miembro de la familia “line” tiene a la salida una señal de audio (parecida a line~), pero alinea los extremos de la señal a los puntos de tiempo requeridos, con la exactitud de una fracción de muestra. (La exactitud está limitada únicamente por el formato numérico del punto flotante utilizado por Pd.) Aún más, muchos segmentos de línea pueden estar especificados dentro de un solo bloque de audio; vline~ puede generar ondas en períodos por debajo de las dos muestras (más allá de lo cual sólo conseguirá sobre-doblado).

El objeto vline~ puede también utilizarse para convertir secuencias de control numérico a secuencias de audio en los modos muestra-más-cercana e interpolación-de-dos-puntos, según se muestra en la figura 3.4 (parte b y c). Para conseguir la conversión muestra-más-cercana, simplemente dele a vline~ un tiempo de rampa de cero. Para interpolación lineal, dele un tiempo de rampa de una muestra (0.0227 milisegundos si la velocidad de las muestras es de 44100 Hertz).

Reproductor de tabla de onda sin lazo

Un área de aplicación que requiere de cuidadosa atención para el control de señales de frontera de secuencia/audio es la lectura de muestras. Hasta ahora nuestros lectores de muestras se han ceñido al asunto de enlazarse perpetuamente. Esto permite una rica variedad de sonidos a los que se puede acceder haciendo cambios continuos en parámetros tales como tamaño del lazo y forma de la envolvente. Sin embargo muchos usos del muestreo requieren que surjan características internas de la tabla de onda en momentos de tiempo predecibles y que se puedan sincronizar. Por ejemplo, los sonidos de percusión grabada son ejecutados usualmente desde el comienzo en una relación de tiempo determinada con el resto de la música.

En esta situación, las secuencias de control están mejor adaptadas que las señales de audio como disparadores. El ejemplo C05.sampler.oneshot.pd (figura 3.14) muestra una forma posible de lograr esto. Los cuatro objetos tilde de la parte inferior izquierda forman la red para el procesamiento de la señal en la reproducción. Un objeto vline~ genera una señal de fase (precisamente un índice de lectura de tabla) para el objeto tabread4~; este reemplaza al phasor~ del ejemplo B03.tabread4.pd (página “49”) y sus derivados.

La amplitud de la salida de tabread4~ está controlada por un segundo objeto

vline~ para prevenir discontinuidades en la salida en caso de que un nuevo evento comience mientras el evento previo todavía suena. El objeto “cutoff” de vline~ tiene como salida una rampa a cero (esté sonando o no), de tal manera que, una vez la salida es cero, el índice de la tabla de onda puede ser cambiado de manera discontinua.

Para iniciar una “nota” nueva, primero el objeto “cutoff” de vline~ es llevado a cero por una rampa; después, luego de un retraso de 5 milisegundos (al cabo del cual el punto vline~ ha alcanzado el cero) la fase se reajusta. Esto se hace con dos mensajes: primero, la fase se ajusta a 1 (sin valor de tiempo, de tal manera que salta a 1 sin rampa). El valor de “1” especifica el primer punto de lectura de la tabla de onda, ya que estamos utilizando interpolación de 4 puntos. Segundo, en la misma caja de mensajes, la fase va en rampa a 441,000,000 en un período de tiempo de 10,000,000 milisegundos. (En Pd los números muy grandes se muestran usando notación exponencial, estos dos aparecen como 4.41e+08 y 1e+07.) El cociente es 44.1 (en unidades por milisegundos) dando una transposición de uno. El objeto vline~ de la parte superior (que genera la fase) recibe estos mensajes vía el objeto “r phase” que está sobre él.

El ejemplo asume que la tabla de onda es llevada a cero suavemente en rampa en el otro extremo, y la porción inferior derecha del parche muestra cómo grabar una tabla de onda así (en este caso de cuatro segundos de duración). Aquí un objeto line~ regular (y computacionalmente más barato) es suficiente. Aunque la tabla de onda debería ser de por lo menos 4 segundos de duración para este trabajo, usted puede grabar tablas de onda más cortas simplemente recortando el objeto line~ antes. La única dificultad es que, si está leyendo y escribiendo simultáneamente de la misma tabla, debería evitar situaciones en las que las operaciones de lectura y escritura ataquen la misma porción de la tabla de onda a la vez.

Los objetos vline~ que rodean a tabread fueron escogidos por sobre line~ debido a que los redondeos de este último de los puntos de quiebre de las fronteras del bloque más cercano (típicamente 1.45 milisegundos) pueden hacer audibles las aperiodicidades en el sonido si la tabla de onda es repetida a más de 10 o 20 veces por segundo, y prevendría un hermoso y periódico sonido a altas frecuencias de repetición.

Regresaremos a la lectura de muestras basadas en vline~ en el siguiente capítulo para adicionar transposición, envolventes y polifonía.

Señales a control

El ejemplo C06.signal.to.control.pd (que no se ilustra) demuestra la conversión de las señales de audio a secuencias de control nuevamente, vía un objeto tilde presentado aquí.

snapshot~: convierte la señal de audio a mensaje de control. Siempre da la muestra de audio computada más recientemente (con la conversión más-rápida-posible) de tal manera que el tiempo de lectura de muestras varía en hasta un bloque de audio.

Es deseable con frecuencia sensar la amplitud de la señal de audio en lugar del pico en una sola muestra; el ejemplo C07.envelope.follower.pd (que tampoco se muestra) presenta otro objeto que hace esto:

env~: seguidor de envolvente RMS. A su salida hay mensajes de control que dan la amplitud RMS en tiempos cortos (en decibeles) de la señal de audio que entra. Una creación de argumentos le permite seleccionar el número de muestras utilizadas para el cálculo de RMS; números más pequeños dan una salida más rápida (y posiblemente menos estable).

Secuenciador estilo análogo

El ejemplo C08.analog.sequencer.pd (figura 3.15) realiza la secuenciación análoga y la generación de envolvente descrita en la sección 3.7. La tabla "sequence", con nueve elementos, contiene una secuencia de frecuencias. El objeto phasor~ en la parte superior ejecuta los ciclos a través de la tabla de secuencia a 0.6 Hertz. La lectura de tabla sin interpolación (tabread~ en lugar de tabread4~) se utiliza para leer las frecuencias en pasos discretos. (Tales situaciones, en las que preferimos lectura de tabla sin interpolación son raras.)

El objeto wrap~ convierte la diente de sierra de amplitud 9 a una amplitud de una unidad, según se describió antes en la figura 3.8, tras lo cual se usa para obtener una función de envolvente de una segunda tabla de onda. Aquí la tabla de onda consiste en seis períodos de una sinusoide. Los granos se suavizan multiplicando por una función coseno elevada (cos~ y + 1).

El ejemplo C09.sample.hold.pd (que no se ilustra aquí) muestra una unidad de muestra-y-retención, otro recurso útil para hacer tareas de control en el dominio de la señal de audio.

Sintetizador estilo MIDI

El ejemplo C10.monophonic.synth.pd (figura 3.16) también implementa un sintetizador monofónico orientado hacia las notas, pero en este caso orientado también hacia el control MIDI. Aquí las tareas de generación de envolvente y secuencia de notas son manejadas utilizadando secuencias de control en lugar de señales de audio. Objetos de control nuevos se requieren en este ejemplo: notein: entrada de nota MIDI. Tres salidas dan la altura, la velocidad y el canal de entrada MIDI de los eventos nota-encendida y nota-apagada (con los eventos nota-apagada apareciendo como eventos nota-encendida con velocidad cero). Las salidas aparecen en el orden acostumbrado de derecha a izquierda.

stripnote: filtra los mensajes nota-apagada. Pasa pares (nota, velocidad) siempre y cuando la velocidad no sea cero, eliminando los otros. A diferencia de notein, stripnote no utiliza entrada o salida de hardware MIDI directamente.

trigger, t: copia un mensaje a la salida en el orden de derecha a izquierda, con conversión del tipo de mensaje a la salida. La creación de argumentos ("b" y "f" en este ejemplo) especifica dos salidas, una que proporciona mensajes "bang" y la otra mensajes "float" (es decir, números). Se crea una salida por cada argumento creado. Las salidas aparecen en el orden normal de Pd de derecha a izquierda.

Los objetos de control del parche alimentan las frecuencias del objeto phasor~ siempre y cuando se reciba un mensaje MIDI de nota-encendida. El control de la amplitud (vía objetos line~) es más difícil. Cuando un mensaje nota-encendida se recibe, el objeto sel 0 tiene como salida la velocidad a la derecha (debido a que la entrada no coincide con 0), esta se divide por la velocidad máxima MIDI de 127 y se empaca en un mensaje para line~ con un tiempo de 100 ms.

Sin embargo, cuando se recibe un mensaje nota-apagada, es apropiado parar el sonido únicamente si la altura de la nota-apagada coincide con la altura que está sonando a través del instrumento. Por ejemplo, suponga que los mensajes recibidos son "60 127", "72 127", "60 0" y "72 0". Cuando la nota-encendida en la altura 72 llega, la altura del sonido debe cambiar a 72 y entonces el mensaje "60 0" se debe ignorar, con la nota sonando hasta el mensaje "72 0".

Para lograr esto, primero almacenamos la velocidad en el objeto float de la parte superior. En segundo lugar, cuando la altura MIDI de la nota llega, se almacena también (el objeto float de la parte inferior) y luego la velocidad se examina de nuevo contra cero (la salida "bang" de t b f vuelve a llamar la velocidad que es enviada a sel 0). Si es cero, el segundo paso es volver a

llamar la altura y examinarla (con el objeto select) contra la altura de la nota-encendida que se recibió más reciente. Únicamente si son iguales (de tal manera que el “bang” aparece en la salida de la izquierda de select) hace que el mensaje “0 1000” vaya al objeto line~. <la nota tarda 1 segundo en dejar de sonar>.

Ejercicios

1. Cuántos parciales de un sonido La 440 pueden representarse digitalmente a una velocidad de muestras de 44100 Hertz?

2. Qué frecuencia debería escuchar si sintetizó una sinusoide a 88000 Hertz a una velocidad de muestras de 44100?

3. Suponga que está sintetizando un sonido a 44100 Hertz, y está computando bloques de audio de 64 muestras. Un evento de control se programa para que ocurra pasado exactamente un tiempo de un segundo, utilizando el esquema más-rápida-posible. En qué número de muestra ocurre el evento?

4. Leyendo a 44100 muestras por segundo, queremos tocar un sonido Do medio repitiendo una onda fija cada N muestras. Qué valor de N deberíamos escoger, y cuántos cents (página “7”) quedaremos por fuera del “verdadero” Do medio?

5. Dos ondas diente de sierra, de una unidad de amplitud, tienen frecuencias de 200 y 300 Hertz respectivamente. Cuál es la periodicidad de la suma de las dos? Cuál es si usted envuelve la suma para obtener un rango de 0 a 1? Al hacer esto hay algún cambio dependiendo de la fase relativa entre ambas?

6. Dos ondas diente de sierra con igual frecuencia y amplitud pero con un desfase de medio ciclo, se suman. Cuál es la forma de la onda y cuál es su amplitud y su frecuencia?

7. Cuál es el nivel relativo, en decibeles, del tercer armónico de una diente de sierra (tres veces la fundamental) comparado con el de la fundamental?

8. Suponga que sintetiza una diente de sierra de 44000 Hertz a una velocidad de muestras de 44100 Hertz. Cuál es la onda resultante?

9. Usando las técnicas de la sección 3.7, dibuje un diagrama de bloques para generar dos sinusoides con fase fija a 500 y 700 Hertz.

10. Dibujar un diagrama de bloques mostrando cómo utilizar el umbral para detectar cuándo una señal de audio excede a otra en un cierto valor. (Podría querer hacer esto para detectar y filtrar la realimentación de los parlantes a los micrófonos.)