Analisis de Señales y Sistemas Digitales 2016
Trabajo Práctico Nº2
Análisis y Síntesis de Audio digital

Pautas para la evaluación **(en orden de importancia):**

- **Resultados**
- Calidad del sonido obtenido
- Análisis y conclusiones
- Usabilidad
- Originalidad e inventiva
- Investigación (se valorará que busquen en fuentes adicionales a las provistas por la cátedra)
- Exposición oral
- Aportes no obligatorios
- Informe escrito

NOTA: Se les recuerda a los alumnos que la Política de Fraude y Plagio del Instituto rige sobre este trabajo e incluye el software y contenido digital entregado.

Fecha de entrega:

- Programa e informe: Miercoles 1 de Mayo
- Presentación oral: A determinar

Para este trabajo práctico deberán implementar distintos métodos de síntesis de instrumentos musicales y efectos de audio.

Existe material extenso, disponible en la web, en papers y libros. Se recomienda investigar en las referencias al final del documento pero se sugiere fuertemente ampliar la bibliografía.

La evaluación de los métodos de síntesis será por los resultados obtenidos, no por la dificultad o facilidad intrínseca del lenguaje en que hayan decidido implementarlos.

1 FFT

a. Implementar en C++ sólo con librerías estándar una FFT que responda a la DFT no normalizada:

$$Y(k) = \sum_{j=1}^{n} X(j) \cdot W_n^{(j-1)(k-1)} \qquad , \qquad W_n = e^{-\frac{2\pi j}{n}} \tag{1}$$

Se debe poder operar con una cantidad de puntos desde 1 hasta, al menos 4096, sujeto a las restricciones del algoritmo (potencias de 2). La implementación del código no tiene restriccionen salvo por su interfaz:

Algorithm 1 Prototipo de FFT.

void fft(vector<complex<float>>& in, vector<complex<float>>& out)
void ifft(vector<complex<float>>& in, vector<complex<float>>& out)
Entrada:

- in: Vector con datos de entrada.
- out: Vector con datos de salida.
- Cantidad de puntos. Debe ser una potencia de 2. No hace falta contemplar otro caso.

El buffer de salida puede ser el mismo que el buffer de entrada. La función debe operar correctamente aún en este caso.

a. Validar el funcionamiento. Comparar con una implementación de referencia.

b. Se debe entregar el test bench en un archivo aparte para que la cátedra pueda linkear el programa de prueba.

Bonus: Se otorgará un punto extra al grupo que implemente la FFT más rápida (si la nota sin este punto es 10, el punto será distribuido en el resto de los trabajos prácticos). El desempeño se medirá mediante el siguiente procedimiento:

- Con n = [256, 512, ... , 2048,4096]
- t_n= tiempo que se tarda en realizar 1 millón de FFT de n puntos
- p_n= t_n/n
- El tiempo total será la suma de los p_n.

Adicionalmente se probará con datos aleatorios para todos los $n = 2^k$ y si el resultado es incorrecto se descalificará todo el ejercicio.

2 Programa Principal

Deberán desarrollar un programa que sintetice una pista a partir de un archivo MIDI (.mid). No es necesario que el programa cuente con interfaz gráfica, se admite el uso de las funciones por linea de comando o archivo de configuración.

Los archivos MIDI contienen una cantidad de "*tracks*". Cada "*track*" se puede asignar a un instrumento, o "*programa*", en la jerga MIDI. Los "*tracks*" contienen los instantes en que se comienza un nota (por ejemplo, cuando se presiona la tecla en un piano), la intensidad de cada nota ("*velocidad*", en la terminología MIDI) y el momento en que termina la nota (cuando se levanta el dedo de la tecla o cuerda).

Su programa debe ser capaz de:

- Cargar un archivo en formato MIDI
- Asignar cualquier instrumento a cualquier track.
- Configurar los parámetros de cada instrumento en cada track.
- Mezclar los distintos tracks para generar una pista.
- Guardar la pista generada en algún formato de audio (wav, mp3, etc).
- Mostrar el espectrograma del audio generado (ver el punto correspondiente del TP). Esta funcionalidad puede estar en un programa independiente.
- Una vez completado el sistema, ensayarlo con un fragmento considerable del segundo movimiento (Adagio) del Concierto de Aranjuez, de Joaquín Rodrigo. En la web se pueden encontrar archivos MIDI con la obra. Incluir el wav generado como parte de la entrega.

Observaciones:

- Existen librerías para leer archivos MIDI. Se sugiere utilizarlas.
- Lo que se describió anteriormente es la funcionalidad esencial que deberá tener su programa. Puede agregar las características y funciones que considere relevantes.

3 Síntesis aditiva de sonidos

La síntesis aditiva consiste en combinar un número de señales senoidales de distintas frecuencias (llamadas "*parciales*" o *partials*), con distinta amplitud. De esta manera se intenta reproducir la estructura de armónicos de un instrumento musical. La ventaja de la síntesis aditiva es su baja complejidad de cómputo y moderados requerimientos en memoria. Si la amplitud de cada "*parcial*" es constante, la calidad del sonido resultante no será buena. En la realidad, cada nota tiene una envolvente de amplitud. La manera más básica de especificar la envolvente se denomina *ADSR: Attack, Decay, Sustain, Release*.

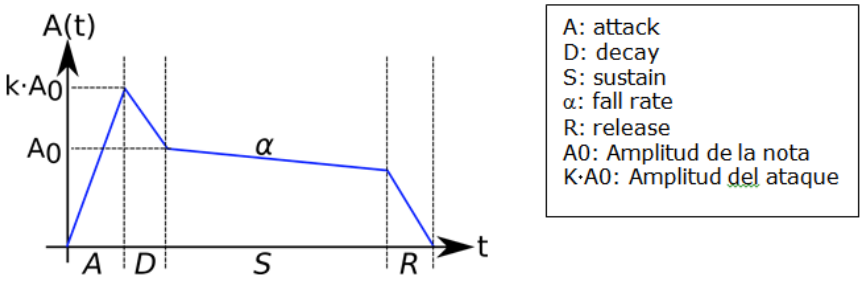


Figure 1: ADSR

Se pide lo siguiente:

- Obtener las amplitudes de los *"parciales"* para al menos 4 instrumentos distintos. Sumar y modular la resultante con una envolvente ADSR. Encontrar los parámetros óptimos.
- La parametrización ADSR es una posibilidad para especificar la envolvente. **Proponga una mejor parametrización y/o forma de envolvente.**
- En general, no es realista suponer que todos los *"parciales"* comparten la misma envolvente. Sintetizar al menos 4 instrumentos, con envolventes independientes para cada "*parcial*". La cantidad de parámetros a determinar puede llegar a ser muy grande. Se recomienda que desarrollen un programa aparte para extraer esta información a partir de muestras. No es necesario restringirse a envolventes ADSR. Utilicen la parametrización que consideren más apropiada para cada caso.
- Investigar e implementar, en caso que mejoren la calidad de síntesis, las siguientes mejoras:
  - Ley de caída de los *"parciales"* según la altura (*pitch*) de la nota.
  - Desviación aleatoria en frecuencia de cada parcial. Esto puede transformar el espectro en inarmónico. Estudiar empíricamente el rango apropiado de desviación para "mejorar" el realismo del sonido generado.
- Proveer, en su programa de síntesis, presets para al menos 4 instrumentos.
- Sacar conclusiones sobre lo aprendido.
- Definir rango en el cual la síntesis funciona correctamente.

4 Síntesis mediante modulación en frecuencia

Una señal de F.M. está definida mediante la siguiente expresión:

$$x(t) = A(t) \cdot cos\left(\Psi(t)\right)$$

donde A es la amplitud, variable en el tiempo; y Ψ es la fase, también variable en el tiempo. La derivada de la fase es la frecuencia instantánea:

$$f_i = \frac{1}{2\pi}\frac{d\Psi\left(t\right)}{dt}$$

La fase se varía mediante la siguiente ley:

$$\Psi(t) = 2\pi f_c t + I(t) cos\left(2\pi f_m t + \phi_m\right) + \phi_c$$

Reemplazando en $x(t)$:

$$x(t) = A(t) cos\left(2\pi f_c t + I(t) cos\left(2\pi f_m t + \phi_m\right) + \phi_c\right)$$

La síntesis F.M. es especialmente adecuada para simular instrumentos de viento. Se utiliza:

$$\phi_m = \phi_c = -\frac{\pi}{2}$$

Lograr un buen sintetizador consiste en elegir adecuadamente $A(t)$ e $I(t)$.

Se pide lo siguiente:

- Sintetizar un clarinete. Para ello deberán crear la funciones $A(t)$ e $I(t)$. En este instrumento la realción entre f_c y f_m es $\frac{n}{m}$ y f_0 resulta del máximo común divisor entre ellas. Investigar la literatura para determinar A, I,n, m. Procurar obtener el sonido más realista. Obtener la evolución del sonido mediante un espectrograma.
- Sintetizar una campana. Ajustar los valores de n y m según corresponda.
- Buscar otros instrumentos de viento. Ajustar n y m para emular esos instrumentos. Se puede variar $A\left(t\right)$ e $I\left(t\right)$.
- Sacar conclusiones sobre lo aprendido.

5 Síntesis de sonidos mediante modelos físicos

El modelo de Karplus-Strong es utilizado para sintetizar instrumentos de cuerda percutida. Se trata de un sistema linear excitado con una secuencia aleatoria de longitud finita. El sistema consiste en una línea de retardo de L muestras, realimentada mediante un filtro La siguiente figura ilustra el modelo conceptual de Karplus-Strong.

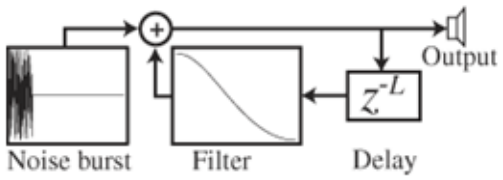


Figure 2: Modelo conceptuaal de Karplus-Strong.

La línea de retardos simula la longitud de la cuerda. A medida que los distintos armónicos recorren la línea, se atenúan y su amplitud va decayendo. El filtro suele ser un pasa-bajos. De esta manera se atenúan más rápidamente las frecuencias más altas. La longitud de la secuencia de entrada debe ser de L muestras. La siguiente figura muestra una posible implementación básica del modelo de Karplus-Strong:

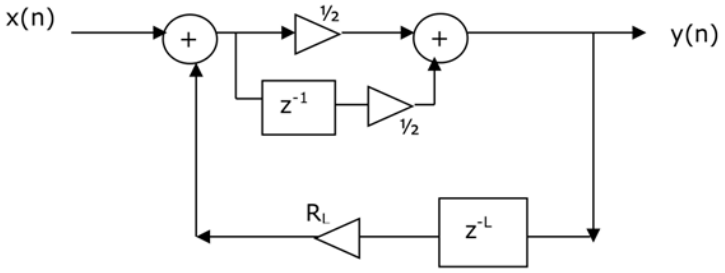


Figure 3: Implementación básica de Karplus-Strong.

En este caso, la salida $y(n)$ se toma luego del pasa-bajo. La ganancia R_L permite ajustar el tiempo de caída de la nota. La frecuencia de la nota es $\frac{fs}{L+\frac{1}{2}}$. Mediante una modificación al modelo, es posible sintetizar instrumentos de percusión. El cambio consiste en realimentar la línea de retardo con ganancia positiva o negativa, de manera aleatoria. Esto se observa en la siguiente figura.

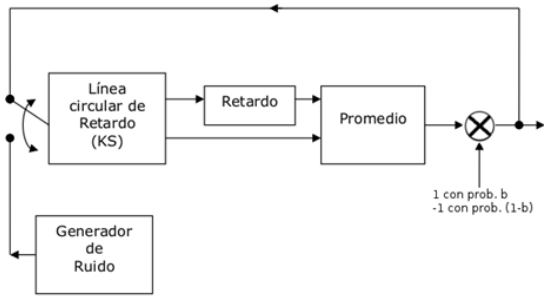


Figure 4: Diagrama en bloques de Karplus-Strong con la variante para la síntesis básica de instrumentos de percusión.

Si $b = 1$, el sistema se comporta como un filtro pasabajo y el sonido es como un instrumento de cuerda. Si $b = 0.5$, el sonido es diferente a una cuerda. Se pierde su "*pitch*" y se asemeja más a un sonido percutido. Si $b = 0$, la señal es negada cada $p + 0.5$ muestras. El sonido es de un arpa en los registros bajos.

Se pide, tanto para el modelo original como para el modificado:

- Encontrar la $H(z)$ y la respuesta en frecuencia del modelo analíticamente. Obsérvese que el modelo modificado no es temporalmente invariante y por lo tanto, estrictamente hablando no tiene $H(z)$.
- Implementar el modelo. Observar salidas relevantes. Investigar la influencia de la distribución de ruido (normal vs uniforme). Simular en Matlab.
- Encontrar y graficar la posición de los polos y los ceros. ¿Varían a lo largo del tiempo?

Para modelo original:

- ¿Cual es el rango de valores que puede tomar R_L para que el sistema sea estable? Plantear la transferencia y extraer el rango de valores a partir de plantear su estabilidad.
- Investigar e implementar un filtro que emule la caja de resonancia de la guitarra.
- El modelo planteado anteriormente tiene un problema. Dado que la frecuencia de la nota está dada por la longitud de la línea de retardo más $1/2$, y esta longitud es un número entero, no se puede obtener cualquier frecuencia. El error resultante es más grande cuanto más aguda es la frecuencia. Investiguen e implementen una solución para mitigar este problema.

Para el modelo modificado:

- Hallar la fase $\Phi(\omega)$ del filtro en función de b.
- Hallar una expresión de b en función de β que permita obtener el valor exacto de la frecuencia fundamental deseada:
$$\beta = \frac{f_s}{f_k} - \left(L + \frac{1}{2}\right)$$
- Hallar la desventaja introducida por este filtro e indicar las limitaciones que genera. (Evaluar en qué octavas conviene usarlo).

Extensiones de Karplus-Strong

Aplicar lo expuesto en el paper "Extending the Karplus-Strong Algorithm to Synthesize Electric Guitar Timbres with Distortion and Feedback", tanto para la distorsión como para el feedback.

Nota: Con las modificaciones y ajustes adecuados, la síntesis de instrumentos de cuerda puede hacerse de muy buena calidad. Investiguen e implementen lo que consideren adecuado.

6 Síntesis basada en muestras

La síntesis basada en muestras busca reproducir un sonido más realista, a partir de muestras grabadas de instrumentos.

Como no es factible tener grabaciones de todas las notas con todas las duraciones, se deben generar las notas a partir de una cantidad reducida de muestras, que se "*estiran*" o "*comprimen*" en tiempo y frecuencia. La siguiente figura ilustra el "*estiramiento*" temporal de una grabación.

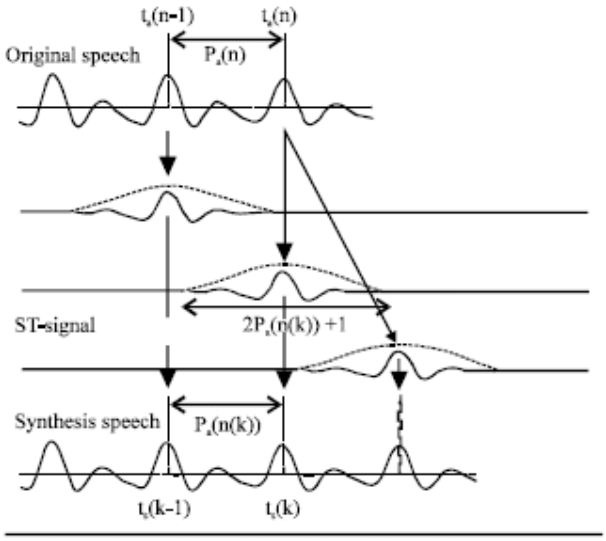


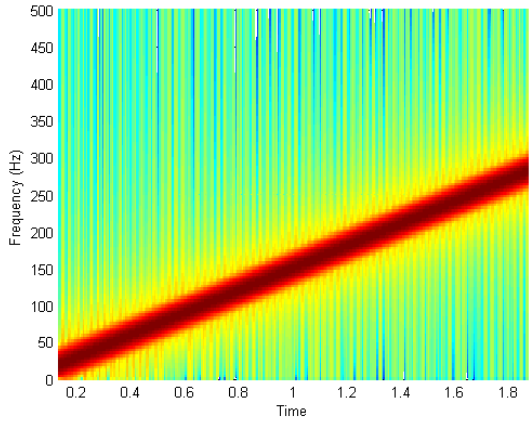
Figure 5: Esquema de funcionamiento de PSOLA.

Se pide:

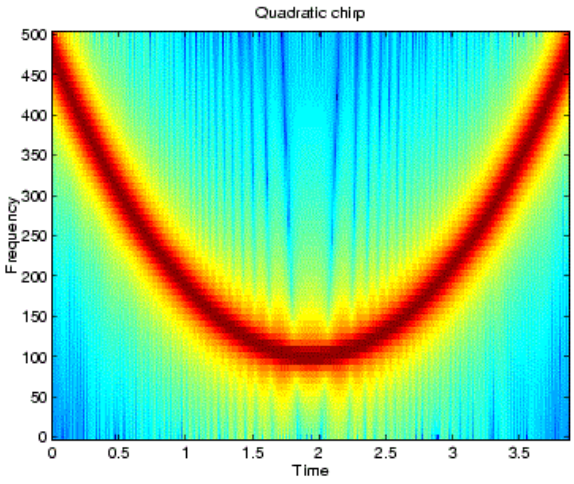
- Obtener varias muestras de uno o varios instrumentos e implementar un sintetizador basado en muestras.
- Investigar la literatura sobre el tema y los métodos PSOLA o WSOLA, entre otros.
- Sacar conclusiones sobre lo aprendido.

7 Espectrograma

Un espectrograma es un gráfico tridimensional que representa la amplitud del contenido espectral de la señal según va variando ésta a lo largo del tiempo. Las siguientes figuras son ejemplos de espectrogramas de señales determinísticas.



(a) Señal cuya frecuencia varía linealmente con el tiempo.



(b) Señal cuya frecuencia varía cuadraticamente con el tiempo. A estas señales se las conoce como "*chirp*"

Figure 6: Espectrogramas

Se pide:

- Implementar una función que calcule y grafique el espectrograma de una señal. Se debe permitir cambiar los parámetros, como el tamaño de bloque o el tipo de ventana.
- ¿Cual es el overlap requerido en un espectrograma según la ventana empleada? Justificar analiticamente.
- Obtenga el espectrograma de una escala de sol mayor (G3), generada con la síntesis aditiva. Emplee aproximadamente $120ms$ por nota. ¿Cuales son los parámetros apropiados? ¿Por qué?
- Sacar conclusiones sobre lo aprendido.

8 Efectos de Audio

Se deberá implementar una función que realice alguno de los efectos vistos en tiempo real. Para hacerlo deberá utilizarse C o C++ y librerías "cross-platform" como PortAudio. La implementación y su interfaz corren por cuenta de los alumnos. No es necesaria una interfaz gráfica, pero se debe presentar de manera tal que el cambio de parámetros de los efectos y la presentación de los mismos se pueda realizar agilmente.

Se debe incluir un análisis sobre cada efecto.

Reverberación

Se deben implementar los siguientes efectos:

- Eco simple.
- Reberverador plano.
- Reberverador pasa-bajos.
- Reberberador completo. Ver referencias al final del documento.
- Reverberador por convolución. Consiste en convolucionar la señal con la respuesta al impulso de un ambiente reverberante. Investigar sobre los métodos de grabación de respuestas al impulso.

Para cada caso:

a. Proveer al usuario ajustes "*amigables*" que se traduzcan en coeficientes para los filtros.

b. Proveer presets.

c. Permitir aplicar los filtros a archivos wav ubicados en disco. Se acepta que se haga un programa aparte para esto.

Flanger

Un flanger es un sistema lineal variable en el tiempo. Suma una señal a la misma señal retrasada. El retardo varía lentamente con el tiempo, de manera cíclica. Existen flangers feedforward y feedback (ver bibliografía citada).

Se pide:

- Encontrar la respuesta en frecuencia (en función del tiempo) del flanger.
- Implementar un flanger.

Robotización

El efecto de robotización está orientado a voz. Su versión más simple se basa en eliminar la información de fase de la señal. El proceso es el siguiente:

a. Se divide la señal en bloques, con overlap.

b. Se aplica una ventana a cada bloque y se calcula la FFT.

c. Se fuerza a cero la fase de cada elemento de la FFT.

d. Se realiza la IFFT de cada bloque.

e. Se reconstruye la señal, sumando cada bloque con el overlap correspondiente.

El motivo para usar solapamiento es que de otra manera se tendría transiciones abruptas entre los bloques. En general se usa overlap de 50% con ventana de Hanning. El motivo es que esta ventana es complementaria: si se suman desplazadas en 1/2 de su largo, suman 1. Demostrar esto.

- Implementar el efecto de robotización.
- Permitir aplicar el efecto a archivos wav ubicados en disco. Se acepta que se haga un programa aparte para esto

Otros efectos

- Giro 3D con auriculares (Stereo).
- Vibrato.

9 Referencias

- Ventanas y espectrogramas, entre otros:
  - https://ccrma.stanford.edu/~jos/sasp/
- Karplus-Strong
  - https://ccrma.stanford.edu/~jos/pasp/Karplus_Strong_Algorithm.html
- Efectos de audio
  - "*DAFx Digital Audio Effects*" de Udo Zölzer
- Lectura recomendada
  - Musical Sound Modeling with Sinusoids plus Noise