# Radio Taller Fourier

Laboratorio 3

Muestreo y Multi-Rate Signal Processing II

## 1. Introducción

En el laboratorio pasado escuchamos algunos audios a una tasa que no era la correcta, generando fuertes distorsiones en el espectro (y por tanto en lo que se escuchaba). En este laboratorio trabajaremos en cómo adaptar la tasa de una fuente minimizando (o incluso evitando, dependiendo de la situación) la distorsión en el espectro.

## 2. Tareas

Comenzaremos adaptando un flujo de una tasa alta a una más baja (en particular, la primera será múltiplo de la segunda). Para ello utilizaremos el bloque `Keep 1 in N`, quien simplemente saca a la salida una de cada `N` muestras de la entrada. Este proceso se llama **Decimar** o **Downsampling**.

1. Genere una sinusoide con una tasa de muestras cuatro veces más alta que la tasa de la tarjeta de audio. Antes de conectarlo al bloque de audio, páselo por un bloque `Keep 1 in N` para disminuir su tasa y hacerla corresponder a la de la tarjeta.

2. ¿Qué sucede a medida que la frecuencia de la sinusoide aumenta? Verifique el espectro de la señal antes y después del bloque `Keep 1 in N`. (Preste atención a configurar correctamente el parámetro `Bandwidth` y `Sample Rate`.)

3. Repita la experiencia con dos sinusoides: una a frecuencias baja y otra a frecuencia configurable que se irá aumentando.

4. Relacione estos fenómenos con lo realizado en la primera parte de la práctica anterior. ¿A qué se deben? Justifique matemáticamente lo que observa.

5. En base a su experiencia responda: ¿qué habría que hacer con el espectro de la señal **antes** de decimar para evitar este último problema? Dicho de otra manera, ¿es posible mantener las dos sinusoides de la parte 3 a la salida del decimador?

6. Implemente lo anterior en GNU Radio y verifique que funciona correctamente (y qué tan cercano al ideal es su comportamiento). Calcule los parámetros de el/los bloques que sean necesarios.

Ahora haremos una prueba similar, pero con música.

7. Descargue un archivo ∼`44.wav` del EVA del curso (el cual debe ser leído a 44 kHz). Genere un diagrama tal que se pueda escuchar el archivo en una tarjeta configurada a 22 kHz utilizando un bloque `Keep 1 in N`.

8. Describa qué sucede con el audio y su espectro (con y sin la corrección implementada en las partes anteriores). En particular, ¿qué sucede en las frecuencias más altas?

Investigaremos a continuación el proceso inverso. Es decir, tomaremos un flujo cuya tasa de muestras es baja y lo adaptaremos a una más alta (nuevamente, la primera será múltiplo de la segunda). Esto generalmente se denomina **Interpolación** o **Upsampling**. Utilizaremos una técnica que quizá a priori no parezca tan intuitiva, pero que veremos da buenos resultados.

El primer paso será agregar una cierta cantidad de ceros entre cada muestra de la secuencia original. Para ello utilizaremos el bloque denominado `Interpolating FIR Filter`, con el parámetro `Interpolation` fijado en `inter` y `Taps` en `[1] + [0]*(inter-1)`[1], donde `inter` es una variable indicando el factor de interpolación que vamos a aplicar.

9. Configure el bloque de audio para recibir muestras a 48 kHz. Genere una sinuoide a una tasa de 12 kHz y configure el bloque `Interpolation FIR Filter` para adaptar las tasa con la del audio. Verifique que el flowgraph esté funcionando como usted desea viendo la señal después de interpolar mediante un `QT GUI Time Sink`.

10. Escuche el audio resultante y compare el espectro antes y después de interpolar (una vez más, preste especial atención a configurar correctamente el `Bandwidth` y `Sample Rate` de los bloques involucrados).

11. Probar con distintos órdenes de interpolación y frecuencia de la sinusoide variable para llegar a una regla cuantitativa que relacione el espectro antes y después de interpolar. Justifique matemáticamente esta regla. Sugerencia: Quizá sea maś fácil el análisis utilizando un tono complejo.

Nuevamente, ahora haremos las pruebas con música.

12. Descargue un archivo ∼`12.wav` del EVA del curso. Genere un diagrama tal que se pueda escuchar el archivo en una tarjeta configurada a 48 kHz utilizando un bloque `Interpolating FIR Filter`.

13. Describa cualitativamente qué sucede con el audio y su espectro. Nuevamente, ¿qué sucede en las frecuencias más altas?

[1] Ésta es una sentencia Python. El companion en el fondo lo único que hace es generar un script Python, por lo que este tipo de sentencias son válidas, y muy útiles cuando queremos hacer cosas paramétricas, o evaluar funciones. En este caso, esto significa únicamente un arreglo con un 1 al comienzo e `inter-1` 0 al final.

14. En base a las experiencias anteriores, nuevamente responda: ¿qué habría que hacer con el espectro de la señal **después** de interpolar para evitar este problema? Calcule los parámetros de el/los bloques que sean necesarios.

15. Implemente lo anterior en GNU Radio y verifique que funciona correctamente (y qué tan cerca del ideal).

En las partes anteriores adaptamos entre tasas que son múltiplos enteros una de la otra.

16. ¿Y si son múltiplos racionales (por ejemplo, 2/3)? Diseñe un sistema capaz que de realizar esta adaptación.