ANÁLISIS DE UN FILTRO CONTROLADO POR TENSIÓN EN EL SINTETIZADOR ANALÓGICO RSF KOBOL EXPANDER

Treball Fi de Grau de

Ricard Martín Fernández

Directors: Perfecto Herrera

Xavier Lizarraga

Grau en Enginyeria en Sistemes Audiovisuals

Curs 2021-2022

*A mi família y mis amigos músicos*

**Agraïments**

Quiero agradecer sobre todo a mis tutores, Perfecto Herrera y Xavier Lizarraga, por su ayuda durante todo el proyecto. Y a mi familia por el apoyo incondicional que me han dado durante este trabajo final y durante toda la carrera. También agradezco a mis amigos y a mis compañeros de mis grupos musicales por animarme y ayudarme a desconectar de vez en cuando.

**Resumen**

El objetivo de este trabajo es analizar el comportamiento del filtro controlado por voltaje[[1]](#footnote-0) (significa que se pueden regular sus parámetros mediante un aparato externo), de una unidad del sintetizador analógico RSF Kobol Expander. Para ello, primero revisaremos la documentación relacionada con la unidad para conocer las especificaciones técnicas de cada control y su funcionamiento. A continuación, crearemos una base de datos, los cuales usaremos para el posterior análisis. Ésta se obtendrá enviando señales de audio desde un ordenador hasta el filtro y se grabará la salida habiendo manipulado los potenciómetros de este, tanto manualmente como con una fuente de voltaje externa. En la tercera y última parte del proyecto, analizaremos detalladamente estas salidas, usando diversos códigos escritos en Python. Con este análisis, podremos comparar los valores que dan en las especificaciones técnicas con los valores encontrados empíricamente y conocer el comportamiento real del filtro.

**Resum**

L’objectiu d'aquest treball és analitzar el comportament del filtre controlat per voltatge[[2]](#footnote-1) (significa que es poden regular els seus paràmetres mitjançant un aparell extern), d‟una unitat del sintetitzador analògic RSF Kobol Expander. Per això, primer revisarem la documentació relacionada amb la unitat per conèixer les especificacions tècniques de cada control i el seu funcionament. A continuació, crearem una base de dades, les quals farem servir per a la posterior anàlisi. Aquesta s'obtindrà enviant senyals d'àudio des d'un ordinador fins al filtre i s'enregistrarà la sortida havent manipulat els potenciòmetres d'aquest, tant manualment com amb una font de voltatge externa. A la tercera i última part del projecte, analitzarem detalladament aquestes sortides, utilitzant diversos codis escrits en Python. Amb aquest anàlisis, podrem comparar els valors que donen a les especificacions tècniques amb els valors trobats empíricament i conèixer el comportament real del filtre.

****

**Abstract**

The objective of this work is to analyze the behavior of the voltage controlled filter[[3]](#footnote-2) (meaning that its parameters can be regulated by an external device), of an RSF Kobol Expander analog synthesizer unit. To do this, we will first review the documentation related to the unit to know the technical specifications of each control and its operation. Next, we will create a database, which we will use for later analysis. This will be obtained by sending audio signals from a computer to the filter and the output will be recorded having manipulated its potentiometers, both manually and with an external voltage source. In the third and final part of the project, we will take a closer look at these outputs, using various code written in Python. With this analysis, we will be able to compare the values ​​given in the technical specifications with the values ​​found empirically and to know the real behavior of the filter.

**Índice**

1. Introducción
2. Síntesis de sonido controlada por tensión
3. RSF Kobol Expander
   1. Historia y funcionamiento general
   2. Parámetros y controles
   3. Emulaciones existentes

1. Metodología

4.1. Grabación

4.2. Procesamiento de señales

4.3. Función de transferencia

1. Análisis de muestras
   1. Análisis global
   2. Análisis de cada parámetro
2. Conclusiones
3. Bibliografía
4. Apéndice

**Llista de figures**

Figura 1: RSF Kobol Expander (completo)

Figura 2: RSF Kobol Expander (sin teclado)

Figura 3: VCF

Figura 4: PolyKB II

Figura 5: Configuración inicial

Figura 6: Loopback

Figura 7: Balanceado vs No Balanceado

Figura 8: Montaje de grabación

Figura 9: Función de transferencia de un filtro

Figura 10: tone100 (input)

Figura 11: Bypass\_tone100 (output)

Figura 12: tone440 (input)

Figura 13: Bypass\_tone440 (output)

Figura 14: tone1k (input)

Figura 15: Bypass\_tone1k (output)

Figura 16: tone10k (input)

Figura 17: Bypass\_tone10k (output)

Figura 18: THD

Figura 19: VCF\_4096F\_allRes\_sweepstat

Figura 20: FFT\_Noise\_Tones

Figura 21: Filter\_Response

Figura 22: TF\_1024F\_10R\_sweepstat

Figura 23: Rango\_Resonancia\_sweepstat

Figura 24: TF\_64F\_0R\_sweepvar

Figura 25: TF\_512F\_5R\_sweepvar

Figura 26: TF\_CV\_freq\_1V\_sweepstat

Figura 27: TF\_CV\_res\_7V\_sweepstat

**Llista de taules**

Tabla 1: Frecuencia\_sweepstat

Tabla 2: Resonancia\_sweepstat

Tabla 3: Frecuencia\_sweepvar

Tabla 4: Resonancia\_sweepvar

Tabla 5: Frecuencia\_CV\_sweepstat

Tabla 6: Resonancia\_CV\_sweepstat

Tabla 7: Frecuencia\_CV\_sweepvar

Tabla 8: Resonancia\_CV\_sweepvar

**1. Introducción**

En este primer apartado del trabajo expondré la motivación de realizar dicho proyecto y sus objetivos y ambiciones. En este trabajo me propongo analizar el funcionamiento del filtro pasa bajos controlado por tensión de una unidad del sintetizador analógico RSF Kobol Expander, disponible en la fundación Phonos.

Para empezar, vamos a definir que es un sintetizador analógico. Un sintetizadores un instrumento musical de tipo electrónico que, a través de circuitos, genera señales eléctricas que son convertidas en sonidos. Una característica que diferencia al sintetizador de otros instrumentos electrónicos es que sus sonidos pueden ser creados y modificados. Además, pueden imitar otros instrumentos o generar nuevos. [1] En el caso del sintetizador analógico, este usa circuitos analógicos para generar estos sonidos. Esta capacidad de generar muchos sonidos diferentes hace del sintetizador un instrumento musical de gran calibre, que se ha usado en muchos géneros musicales, entre los cuales están, sobre todo, la música electrónica o el synthpop, pero también géneros más duros como el industrial metal o el nu metal.

Un dispositivo analógico genera una señal continua, no discretizada y es imperfecto sónicamente, ya que padece de desviaciones, ruidos y otros aspectos técnicos que normalmente no se consideran en el audio digital. De hecho, es casi imposible que una unidad del mismo modelo suene exactamente igual que otra, pueden ser similares pero el estado de cada componente, su diseño y el control de calidad durante la producción definen esa similitud. Son esas imperfecciones las que hacen que el sonido analógico sea único y que hoy en día fabricantes de plugins busquen recuperar y modelar estos sonidos, inundando el mercado de emulaciones analógicas. Actualmente, los dispositivos analógicos siguen consumiéndose y van al alza en un entorno profesional y una prueba de ello es que en la mayoría de estudios de grabación apuestan por un sistema híbrido donde conviven el mundo analógico y el digital.

Personalmente, el mundo de los sintetizadores es un mundo que, como músico, me intriga mucho y en el que nunca he participado activamente. Por eso, me interesa hacer un proyecto sobre un sintetizador. Además me intriga la diferencia entre lo que se dice en las especificaciones técnicas, respecto al comportamiento y los valores reales, ya que, al ser analógico, es muy probable que, en la práctica, haya desviaciones. Uno de los elementos más característicos y diferenciadores de un sintetizador es su filtro, a menudo de tipo pasa-bajos. Por ello, este trabajo se centra en él. Un filtro pasa-bajos es, esencialmente, un proceso que se aplica a una señal para mantener los sonidos graves y eliminar los agudos. Veremos en la sección 2 que significa ell hecho de que se pueda controlar por tensión.

Otra motivación es que al ser un sintetizador muy antiguo (lo veremos en el capítulo 3) es posible que haya componentes que ya no se fabriquen y que estén en mal estado, por lo que el sintetizador original puede tener un comportamiento diferente al de otros sintetizadores más modernos. Además, como también veremos en el apartado 3, el sintetizador con el que quiero trabajar es muy raro, cosa que le aporta un valor especial.

Por esos motivos, el objetivo de este proyecto es analizar el procesador analógico del VCF (hacerlo de todo el sinte sería un trabajo mucho más largo y complejo). Un VCF (Voltage Control Filter) es un filtro cuya frecuencia de corte depende del valor del voltaje en su puerto de control [2]. En otras palabras, lo que éste hace es modificar la envolvente de la señal en torno a la frecuencia de corte del filtro usando una señal externa.

El análisis lo haremos utilizando respuestas impulsionales (IR), como explicaremos en las secciones 4 y 5. Para ello, es necesario tener una base de datos con sonidos o señales de test para extraer resultados y conclusiones y poder analizar. La calidad de esta base de datos (resolución y calidad del audio) definirá la calidad del resultado final del análisis. Cuantas más muestras tengamos, más preciso será este.

Como hemos mencionado, en este proyecto, lo interesante es definir y almacenar señales de prueba y registrar la salida del VCF. De esta manera tenemos la opción de analizar la respuesta en frecuencia del filtro ante una señal que conocemos.

Analizando las muestras registradas podremos conocer el valor de la frecuencia de corte del LPF, y también el valor del pendiente y de los parámetros de la resonancia. Lo primero de todo es capturar el efecto de cada parámetro por separado con el objetivo de conocer su rango, desviación y comportamiento o respuesta (si es lineal, logarítmico o exponencial).

Por otro lado también es importante conocer las especificaciones técnicas de la unidad proporcionadas por el fabricante y analizar aspectos técnicos como la THD (Total Harmonic Distortion), la SNR (Signal to Noise Ratio)  y la envolvente en frecuencia del ruido de fondo del filtro, puesto que también forman parte de la señal de salida.

Veremos cómo funcionan los parámetros del filtro en la sección 3, además de explicar brevemente la historia y el funcionamiento general del sintetizador que contiene el módulo que queremos trabajar, el RSF Kobol Expanded, y daremos más detalles sobre la grabación y el posterior análisis de estas en las secciones 4 y 5.

**2. Síntesis de sonido controlada por tensión**

Antes de entrar en materia, vamos a explicar brevemente cómo funciona un VCF y el concepto de control por tensión eléctrica.

El VCF se basa en los siguientes pasos:

* Existe una señal de entrada que excita una envolvente temporal. Esta señal puede ser externa, que es como lo haremos en este proyecto, o interna (producida por un generador de ondas, por ejemplo).
* Esta envolvente temporal se mapea a un valor en frecuencia.
* Este valor define la frecuencia de corte del filtro LPF del VCF.

Es decir, tenemos un LPF variable según la envolvente de la señal de entrada. Aunque es posible desconectar esa modulación, y hacer que los parámetros del filtro sólo se modifiquen manualmente.

El concepto de control por tensión (CV) está en la raíz misma del sintetizador modular. La idea general es que los niveles de tensión analógicos producidos por otro instrumento o por una fuente externa de voltaje se pueden utilizar para controlar las funciones y los parámetros de un módulo. Por ejemplo, un CV puede determinar el tono de un oscilador mientras que un segundo CV puede determinar la potencia de una señal después de que pase a través de un VCA (amplificador controlado por tensión). Por ejemplo, cuando un módulo de sintetizador dice que presenta "CV sobre la resonancia del filtro", eso significa que hay una entrada de CV para controlar la cantidad de resonancia que tiene el filtro usando una fuente de señal externa [3]. Un CV externo podría ser, por ejemplo, un pedal, un secuenciador o una fuente de voltaje.

Como veremos en el siguiente capítulo, casi todos los controles de la parte delantera del Kobol pueden ser regulados con un CV externo, lo cual es otra particularidad importante de este sintetizador.

**3. RSF Kobol Expander**

3.1 Historia y funcionamiento general

El RSF Kobol es un sintetizador monofónico (sólo puede producir una nota a la vez) francés lanzado en 1978. Fue creado por los hermanos Ruben y Serge Fernandez. Es un instrumento muy versátil que ofrece muchas características buscadas de otros sintetizadores analógicos de su época.[4]

Más tarde, en 1981, se lanzó el RSF Kobol Expander (Figuras 1 y 2), el cual era una versión en rack del RSF Kobol original. Entre otras cosas, contenía 2 VCO (*Voltage Control Oscillator*), 2 controles de ADS (*Attack, Decay y Sustain*), 1 LFO (*Low Frequency Oscillator*), 1 VCF, 1 VCA (*Voltage Controlled Amplifier*), 1 generador de ruido y 1 procesador de voltaje.[5]

Aunque es principalmente un sintetizador analógico tradicional con osciladores estables y muchos controles que le permiten controlar la envolvente ADS, la sección de filtro y el LFO, ofrecía algunas características únicas para su época. En particular, las formas de onda del oscilador se pueden barrer continuamente para obtener una transición muy interesante entre los diferentes tipos de onda que puede generar (sinusoide, sierra, pulso, etc). Esto, así como la activación y desactivación de notas y otros parámetros, se pueden controlar a través de CV.

La versión de rack venía en cuatro componentes diferentes. El Kobol Rack (Expander 1) contenía, esencialmente, las secciones VCO/VCF/VCA/LFO del Kobol en un módulo de rack (tal como hemos mencionado antes). El Expander 2 fue un complemento del Kobol que agregó algunos módulos de procesamiento nuevos, como modulación en anillo, retención de muestras y seguidores de envolvente, así como módulos VCA y LFO adicionales. El Programmer contenía los circuitos de memoria para almacenar *patches* y secuencias. Por último, el KM8 era un mezclador de montaje en rack de 8 canales. Estos cuatro componentes se pueden unir o se pueden usar por separado.

El Kobol se veía y sonaba muy bien, y rápidamente fue absorbido por las grandes empresas de sintetizadores de la época. El problema era que, al ser una pequeña compañía francesa, RSF nunca pudo producirlos en masa en el mismo tipo de escala que Moog, Arp, Roland o Korg. Se fabricaron menos de 200 teclados Kobol, alrededor de 800 RSF Kobol Expander, 200 Expander 2, 200 Programmer y solo unas decenas de KM8. Esto hace que sean sintetizadores raros y muy caros (alrededor de 5000 euros [6]). Han sido utilizados por artistas y grandes grupos de música como Vince Clarke, Vangelis, Jean-Michel Jarre, Depeche Mode, Peter Gabriel y Hans Zimmer.[7]

Como introducción al siguiente subapartado, vamos a definir brevemente los módulos que contiene el RSF:

**- *Keyboard:*** Es el módulo que contiene las teclas del teclado. Genera voltajes continuos al tocar las teclas.

**- VCA (*Voltage Controlled Amplifier*):** Es un amplificador cuya ganancia depende del valor del voltaje en su puerto de control.

**- VCF (*Voltage Controlled Filter*):** Es un filtro pasa bajos cuya frecuencia de corte depende del valor del voltaje en su puerto de control.

**- VCO (*Voltage Controlled Oscillator*):** Es un oscilador que genera una onda con una frecuencia que depende del valor del voltaje en su entrada.

**- LFO (*Low Frequency Oscillator*):** Es un oscilador que genera una onda de baja frecuencia en su salida (es un tipo particular de VCO).

**- Envolvente ADS:** Consiste en una serie de parámetros (*Attack, Decay, Sustain*) que modelan la envolvente de una onda. [2]

Ahora si, en el siguiente subapartado, veremos en detalle el funcionamiento teórico de los diversos módulos del Kobol Expander y como se pueden aislar para estudiar y modelar únicamente el comportamiento del VCF.

3.2. Parámetros y controles

El RSF Kobol Expander tiene tres modos de funcionamiento: manual, de memoria y secuenciador. El manual permite utilizar el sintetizador como un dispositivo convencional. El modo de memoria sirve para grabar y almacenar lo que toquemos en modo manual o secuenciador. Y por último, el modo secuenciador sirve para reproducir una secuencia de notas [8]. Cabe decir que la unidad con la que trabajaremos no contiene el teclado, solo el rack de los módulos, de modo que trabajaremos en modo manual.

Encontramos los siguientes controles y parámetros:

* *Voltage Controlled Oscillators* (VCO)

El RSF tiene dos VCO que van desde 10Hz a 10kHz [9], en un rango de 4 octavas cuantizadas por semitonos [8] (ampliable por CV, que también puede variar su tono y su nivel) con una salida para cada uno. Lo que hace inusual a este VCO es el control de selección de forma de onda; en lugar del selector de forma de onda conmutada habitual que se encuentra en la mayoría de los sintetizadores, los controles del RSF son totalmente variables entre las formas de onda. Éstas pueden ser: triángulo, diente de sierra ascendente, cuadrado y pulso. Al configurar el control a medio camino entre dos formas de onda que se muestran en el panel, se pueden obtener mezclas interesantes y hay muchos sonidos inusuales disponibles. Esto también puede ser controlado por CV, lo que significa que puede barrer automáticamente las formas de onda aplicando la salida de uno de los envolventes generadores o el oscilador de baja frecuencia a la entrada de control de forma de onda.[9]

Los dos VCO son idénticos y proporcionan una amplia gama de sonidos. El volumen de cada oscilador está controlado de manera separada, es decir, hay un potenciómetro de volumen independiente para cada VCO. También hay una función de sincronización, por lo que, como se puede ver, hay infinitas posibilidades para la creación de formas de onda únicas.

* *Voltage Controlled Filter* (VCF)

En la Figura 3 podemos ver con claridad el módulo de VCF, que para este proyecto es el más importante de los 4 que el RSF Kobol Expander contiene.

Este es un filtro pasa bajos (LPF) de 4o orden, con una pendiente de 24 dB/octava. Hay cuatro potenciómetros asociados con el filtro: frecuencia de corte (*VCF Frequency*), resonancia, cantidad de ADS (*Attack, Decay y Sustain*) y control de teclado (*KeyB Control*). También hay entradas para el CV externo de cada uno de estos controles y también una entrada de audio para procesar una señal externa [9].

De estos 4 parámetros que hemos nombrado, dos determinan el comportamiento del VCF:

* *VCF FREQUENCY*: define la frecuencia de corte. Se duplica con el botón VCF ubicado a la izquierda del teclado. [8]
* *RESONANCE*: define la campana de resonancia que se genera en la frecuencia de corte para dar un cierto carácter al sonido generado. Si la rueda está girada al extremo izquierdo, el filtro eliminará gradualmente las oscilaciones de los armónicos altos. Si está girada a la derecha, el filtro entrará en resonancia propia (oscila de manera automática), creando una onda sinusoidal audible, cuyo tono estará determinado por el control de frecuencia.[8]

Y dos modifican el sonido global del VCF y su relación con el teclado:

* *KEYB. CONTROL*: Realiza un seguimiento de la frecuencia que le transmite el teclado y mueve la frecuencia de corte en consonancia con esa frecuencia que le llega.

En otras palabras, permite que el VCF se sintonice con el teclado. Si está en la posición 0, el VCF será independiente de la nota tocada. Si está en la posición 1, se sintonizará con la nota tocada, eso hará que el sonido sea homogéneo en todo el teclado. Finalmente, colocado en la posición 2, obtendremos un sonido muy rico en armónicos en las notas altas del teclado [9].

* *ADS CONTROL*: controla el ADS del VCF. La frecuencia de corte del filtro evolucionará según la envolvente creada con los potenciómetros de *Attack, Decay* y *Sustain.*

*Attack* es el tiempo que tarda el sonido en escucharse desde que se aprieta la tecla; *Decay* es el tiempo que tarda el sonido en bajar hasta el volumen de *sustain* y *Sustain* es el tiempo que una nota se mantiene en un volumen constante.

Más adelante veremos que el otro parámetro que normalmente se ve en ADSR, *Release*(R), no se tiene en cuenta porque está vinculado al *decay*.

Para analizar de forma precisa, es necesario conocer el rango exacto del parámetro *VCF FREQUENCY*, y si su escala es logarítmica o lineal en frecuencia. En la serigrafía se indica un rango de 16Hz a 16KHz, pero, en la práctica, pueden haber desviaciones.

Con el parámetro de resonancia el objetivo es el mismo, conocer el ancho de banda y l’altura de la campana para valores discretos de este parámetro.

Por otro lado, el filtro se puede controlar con señales externas. Es interesante conocer la respuesta del VCF con ellas para aislarlo de las señales que genera el VCO, que pueden tener sus propias desviaciones. Para ello podemos rutear esas señales de prueba a estas entradas y ver cómo responde el filtro. Eso es lo que haremos en este proyecto (lo veremos en la sección 4), usar una señal de control externa.

* *Envelope Generator*

Hay dos generadores de envolvente en el RSF, uno para el VCA y otro para el VCF (VCA ADS y VCF ADS respectivamente).[9] El VCA controla la envolvente del sonido final, después de haber pasado por los otros módulos, antes de salir del sintetizador.

Los generadores de envolvente funcionan de manera que el tiempo de *release* final está vinculado al control de *decay* inicial y se activa o desactiva. El rango de control con los tiempos de *attack* y *decay* es muy amplio, desde 1 ms hasta más de 20 segundos. Una vez más, hay entradas que permiten variar estos tiempos y el nivel de sostenido con un CV externo.

Al aplicar el CV del teclado directamente a la entrada de decay, las notas bajas tendrán un *decay* más corto que las notas altas. En cambio, al enrutar el CV del teclado a través del inversor en el procesador de voltaje, las notas bajas tendrán una caída prolongada que se acortará progresivamente cuanto más alto toque el teclado, simulando así instrumentos acústicos más de cerca [9].

* *Low Frequency Oscillator (LFO)*

El LFO genera señales cuadradas o triangulares para modular los VCO y el VCF [8]. La tasa de modulación es muy amplia, desde 0,01 Hz hasta 100 Hz y esta tasa también es controlable por CV. El generador de ruido emite ruido blanco o rosa que se puede seleccionar mediante un interruptor interno. La salida es muy alta y también se puede utilizar como fuente de control [9].

Existen otros parámetros en el sintetizador, aunque son menos importantes para este trabajo, así que los definiremos brevemente, sin entrar en detalles técnicos: [8]

* *BYPASS*: Cambia para mantener un sonido indefinidamente. Equivalente a mantener presionada una tecla en el teclado.
* *TUNE*: Afinación del sintetizador en comparación con otros instrumentos de una orquesta. Si está en 0, el dispositivo es calibrado en la nota LA de 440Hz.
* *BEAT:* Ajuste de ritmo entre los 2 VCO. El 2 está sintonizado con el 1.
* *VCO-MOD, VCF MOD*: Modulación permanente de VCO y VCF a través del LFO.
* *VCO 1 MOD KEY OFF*: Elimina la modulación en VCO 1 cuando se presiona el botón.
* *SYNCHRO KEY*: sincronización con VCO 1 cuando la tecla está apretada.

Finalmente, llegamos al procesador de voltaje. Este es un módulo realmente útil que es capaz de aumentar los voltajes hasta 4 veces más, y hay salidas normales e invertidas [9]. En la Figura 1 podemos ver que a la izquierda del teclado, debajo de todos los parámetros que acabamos de describir, hay una pequeña placa con más controles. Estos son los siguientes: [8]

* RUEDA DE TONO: Controla el tono de los osciladores en un rango de más o menos un quinto.
* RUEDA MOD: Modula los osciladores a través del LFO.
* *INTERRUPTOR DECAY*: en posición *OFF* elimina el tiempo de retorno del sonido (si dejamos de apretar la tecla, el sonido se apaga instantáneamente).
* INTERRUPTOR DE DESLIZAMIENTO (*GLIDE*): en la posición *ON*, se realiza un enlace entre las diferentes notas tocadas en el teclado a una velocidad variable determinada por el potenciómetro *GLIDE*, situado arriba a la izquierda.
* INTERRUPTOR DE OCTAVA : Modifica el teclado aumentando o reduciendo en 1 octava su rango.
* BOTÓN VCF: Modifica la frecuencia de sintonía del filtro. Su acción se combina con el potenciómetro de FRECUENCIA del VCF.

3.3. Emulaciones existentes

Para familiarizarnos con la unidad antes de ir al estudio a grabar, hemos usado una versión bastante parecida de un plugin en VST llamado PolyKB II (Figura 4), el cual es bastante parecido al Kobol original, con algunas modificaciones però con un sonido y una estètica muy similares.

Los osciladores del Polykobol II se han implementado basándose en el RSF Kobol original, incluida su capacidad para cambiar suavemente de una forma de onda a la siguiente. También tiene una función bastante ingeniosa (no derivada del sintetizador original) que le permite colocar distorsión en la entrada del filtro o en su salida.

PolyKB cuenta con dos generadores de contorno ADSR, el ADSR2 (que controla el VCF) y el ADSR1 (que controla el VCA). El botón *Loop2* recorre las etapas 'A' y 'R' del ADSR2.

Por otro lado, el *key tracking* (el equivalente del control del teclado en el modelo original) solo afecta las etapas de ataque de las envolventes, lo que significa que es inútil para imitar las respuestas de instrumentos punteados o martillados como guitarras, pianos y percusión.

Los 2 LFO permiten seleccionar cualquier número de formas de onda simultáneamente para crear formas de onda complejas.

En la sección de Modulations vemos cinco fuentes de modulación (LFO1, Ruido, ADSR2, VCO2 y velocidad) enrutadas (a través de la rueda de modulación) a los tonos y formas de onda de los osciladores, así como a la frecuencia de corte del filtro.

En la parte izquierda, hay un arpegiador simple (reproduce una secuencia de notas) con modos arriba, abajo y arriba/abajo. Si no se selecciona ninguno de estos, se arpegia aleatoriamente.

Debajo de este, el panel de *OUTPUTS* te permite seleccionar el número de voces disponibles (de 1 a 16), el rango de inflexión, la sensibilidad a la velocidad, la sensibilidad a la presión y la cantidad de desafinación. Encima se encuentran los menús que permiten seleccionar unison (de dos a seis voces) y si queremos sonido polifónico o monofónico (esto es una mejora importante, ya que el modelo original era monofónico).

Debajo de la sección de modulación tradicional se encuentra un panel que ofrece dos buses que permiten seleccionar dos fuentes de modulación de un total de 12 y dirigir cada una hasta cuatro destinos seleccionados de un total de 29. Esto es muy útil, pero no se puede determinar la cantidad de modulación aplicada a cada destino.[10]

En resumen, es un *plugin* bastante parecido al modelo original, con algunas mejoras y algunos fallos, pero con un funcionamiento muy similar

**4. Metodología**

En esta sección explicaremos con detalle qué procesos realizaremos con tal de obtener los resultados que analizaremos en la sección 5.

4.1. Grabaciones

El primer paso es crear la base de datos con tal de ver cómo responde el filtro a una señal externa. Pero antes de eso tenemos que preparar esas señales. Éstas se crearán conforme a una serie de preguntas, las cuales responderemos en el apartado 5 de este trabajo.

Estas preguntas nos ayudan a establecer un esquema o una guía de cara a realizar un buen análisis. Y son las siguientes:

1. Hay diferencias en la respuesta en frecuencia del filtro para diferentes niveles de señal de audio de entrada?

2. ¿Cuál es el rango de la frecuencia de corte?

3. ¿Cuál es el rango de la resonancia?

4. ¿Cuál es el comportamiento de los parámetros? Lineal, logarítmico, exponencial?

5. ¿Cuál es la SNR del filtro?

6. ¿Cuál es el nivel de THD del filtro a la salida?

7. ¿Cuál es su pendiente real?

8. ¿Cuál es el rango de la Q definida por la resonancia?

9. ¿Responden igual los parámetros controlados por CV? .

En la siguiente página veremos qué señales son las que hacen falta para responder cada pregunta.

Las grabaciones se obtendrán enviando señales de prueba, previamente creados y preparados, desde un ordenador hasta el filtro, y se grabará la salida de éste habiendo manipulado los parámetros de este con diversas combinaciones, tanto manualmente como con una fuente de voltaje externa.

Además de las salidas del filtro manipulando los parámetros, también necesitamos captar la salida en *bypas*s (sin ningún proceso, con el potenciómetro de la frecuencia de corte al máximo y el de la resonancia al mínimo, es decir, con el filtro abierto) para tener una referencia de la respuesta sin que el filtro actúe. Nos servirá para comparar con los niveles de las tomas procesadas y obtener geométricamente los valores de la frecuencia de corte y la pendiente de cada salida (lo veremos en el siguiente apartado).

También hará falta una grabación del paso de la señal por la tarjeta de sonido utilizada para conectar el ordenador con el sintetizador, sin llegar al filtro, para tener la señal que le llegará al VCF, que contiene tanto la señal de control como el ruido y coloración aportados por la tarjeta (esa señal es el *loopback*).

Un paso muy importante antes de realizar las grabaciones es la calibración del sintetizador. Es necesaria para asegurarnos de que siempre estamos en las mismas condiciones de nivel. Para este paso, es muy importante tener en cuenta el nivel de referencia que utilizaremos para la calibración, el cual depende de la tarjeta de sonido que usemos.

Para realizar las grabaciones con el ordenador portátil, usamos una tarjeta de sonido modelo *Edirol UA-101*, la cual, usando las entradas de *jack*, tiene un nivel de voltaje de salida máximo de 16 dBu y un voltaje nominal (de referencia) de 4dBu [11]. El nivel máximo se corresponde con los 0 dBFS de la señal digital, al cual, para tener la señal en el nivel de referencia, hay que restarle 12 dBFS (que es la diferencia entre 16 y 4 dBu). Pero estos son los dB referencia de rms, si queremos el voltaje de pico, tenemos que sumarle 3 dB, con lo cual, finalmente tenemos que restarle 9 dBFS a la señal original, con tal de referenciarla y calibrarla correctamente.

Realizando estos ajustes, las señales de entrada que tenemos para las diferentes preguntas son los siguientes:

Pregunta 1: *Sweep* variable con 3 amplitudes diferentes, empezando con el nivel máximo referenciado, una diferencia de 10 dB y 5 segundos entre una y la siguiente (con niveles de -9 dBFS, -19 dBFS y -29 dBFS respectivamente).

Preguntas 2,3 y 4: *Sweep* estática al nivel máximo (-9 dBFS). Es un tren de 11 señales con 5 segundos de espacio entre dos señales consecutivas, para poder grabar la salida con 11 puntos diferentes de los potenciómetros de frecuencia de corte y resonancia.

Preguntas 5 y 6: 4 tonos puros de 100 Hz, 440 Hz, 1 kHz y 10 kHz. (todos al nivel de referencia de -9dBFS). El tono de 440 Hz nos sirve para ver el comportamiento en la nota de calibración musical estándar.

Preguntas 7 y 8: No hace falta ninguna grabación ni, por tanto, ninguna señal de entrada, ya que son preguntas que responderemos después, en el análisis de las tomas para las preguntas 2, 3 y 4.

Pregunta 9: Usaremos las mismas señales que para las preguntas 1,2,3 y 4 pero en vez de cambiar los parámetros manualmente, lo haremos usando una fuente de voltaje externa.

Además del nivel, todos los señales tienen en común una série de parámetros:

*frame rate*: 48 kHz

formato: *wav* y 24 bits de *encoding*

canales: mono (1 canal)

duración: 15 segundos

Estas señales las hemos creado usando los *plugins* de *Aurora* en *Audacity* [12]. La razón por la que esas señales son *sweeps* es por que para poder observar el comportamiento del filtro según los parámetros de la frecuencia de corte y la resonancia hace falta una señal que recorra todo el espectro que nos interesa (de 20 a 22 kHz) y esa señal es la *sweep*.

Las salidas que grabemos tienen que tener un nombre fácilmente interpretable y único, ya que habrá muchas grabaciones. Las señales de entrada tendrán el nombre del tipo de señal que son, por ejemplo, *sweep\_static*. Por otro lado, las salidas tendrán el nombre de los parámetros que toquemos y del tipo de señal de entrada de origen, por ejemplo *VCF\_16F\_0Res\_sweepvar (*que seria la salida con la frecuencia de corte a 16 Hz y la resonancia a 0, habiendo enviado la *sweep* variable como señal de entrada) o *Bypass\_sweepstat* (que sería la salida en *bypass* enviando la *sweep* estática como señal de test). Esto es importante porque también en la parte del análisis, generaremos muchos archivos, por lo que es importante tener clara la nomenclatura que usaremos, y que sea fácil y comprensible para alguien externo al proyecto.

Algo muy importante y útil para el proyecto es que en este sintetizador, se puede acceder a cada proceso por separado, lo que implica que el VCF acepta señales externas sin tener que pasar por los módulos anteriores (VCO y LFO), de modo que la señal le llega limpia, sin ninguna interferencia causada por los otros módulos, únicamente con las posibles interferencias provocadas por el cable y la tarjeta de sonido.

Aún así, por si acaso, dado que queremos observar únicamente el filtro, todos los demás controladores (VCO, LFO, VCA…) deben estar al mínimo (a 0) de modo que no afecten a la salida del VCF.

No obstante, el primer día de grabación descubrimos que los potenciómetros del VCA (el siguiente módulo por el cual pasa el sonido después de pasar por el VCF) tienen que estar en una posición concreta para que el sonido no se vea afectado, concretamente el *Sustain* del ADS 1 debe estar al máximo. Sucede igual con el potenciometro del volumen final, antes de salir del sinte, tal como vemos en la Figura 5, la cual muestra también la configuración inicial de los parámetros de frecuencia de corte y resonancia del filtro, de manera que el filtro está desactivado y no debería afectar al sonido entrante (la salida en este estado sería el *bypass*).

El método de grabación consistirá en enviar la señal al módulo desde un ordenador y grabar la salida del VCF, moviendo los potenciómetros de frecuencia y resonancia . Cada pregunta tendrá un método diferente, aunque en todas (excepto la 9) los cambios en los potenciómetros se harán manualmente:

Pregunta 1: (*sweep* variable)

La idea es trabajar con 4 puntos del potenciómetro de frecuencia (16, 64, 512 y 4096 Hz) separados uniformemente y con 5 de resonancia (0, 2, 5, 8, 10) y grabar la salida con las 20 combinaciones posibles: 16 Hz y 0 resonancia, 16Hz y 2 resonancia, …, 4096 Hz y 10 resonancia. La frecuencia máxima (16 kHz) no la tenemos en cuenta porque equivale al *bypass*.

Preguntas 2, 3 y 4: (*sweep* estática)

En este caso queremos ver cómo responde cada frecuencia al cambio de resonancia, para saber cómo se comporta el filtro con una señal de mismo nivel y cual es el rango real de cada potenciómetro. Para ello, la idea es coger cada frecuencia y grabar la salida con 11 valores de resonancia, que corresponden a los puntos marcados en el potenciómetro (por eso la señal de test es un tren de 11 señales, una para cada resonancia). Moveremos el potenciómetro de resonancia de menos a más, es decir, de izquierda a derecha.

También habrá una pista (la *VCF\_allF\_0R\_sweepstat*) en la que queremos ver cómo cambia el comportamiento del filtro a medida que la frecuencia de corte va decreciendo (grabamos moviendo el potenciómetro de más a menos, es decir, de derecha a izquierda), sin ninguna resonancia. Esta salida es la que usaremos posteriormente para conocer los valores de la frecuencia de corte y pendiente con esta señal de entrada.

Preguntas 5 y 6: (tonos puros)

Para conocer el SNR (*Signal to Noise Ratio*) y la THD (*Total Harmonic Distorsion*) solo nos hace falta conocer la salida del filtro sin que esté actúe, es decir, el *bypass*, para poder compararlo con la señal original. Usamos un tono puro (4 tonos puros a diferentes frecuencias) ya que será más sencillo ver si hay distorsión o ruido que con una *sweep.*

Pregunta 9: (fuente de voltaje externa)

En este caso, usaremos una fuente de voltaje externa (la *PROMAX FAC-363B*) para modificar los parámetros. Concretamente, haremos dos grupos de grabaciones, en el primero, usaremos la fuente para cambiar la frecuencia de corte y en el segundo, la usaremos para cambiar la resonancia. En ambos casos, haremos 4 grabaciones enviando 4 voltajes diferentes (1V, 4V, 7V y 10V). Estos voltajes se corresponden, respectivamente, con 32Hz, 256Hz, 2048Hz, 16kHz ya que el sintetizador puede recibir 1 Voltio/octava [8] y con 0V tenemos la frecuencia de corte mínima (16Hz).

Hay que tener en cuenta que es necesario enviar una señal de audio igualmente, ya que esta señal de voltaje únicamente sirve para modificar los valores de frecuencia y resonancia. Por sí sola, no envía ninguna señal de audio. Haremos los dos tipos de grabaciones con la *sweep* estática y con la *sweep* variable, por lo que al final tendremos 16 grabaciones.

En total, contando todas las preguntas y todas las grabaciones que acabamos de explicar, más los *loopbacks* y las señales de test, tendremos un total de 66 archivos.

Hemos hablado de dos parámetros del VCF, eso es porque los otros dos (KB y ADS) son parámetros que añaden detalles al filtro pero que no son parte de su funcionamiento básico, sinó que sirven para controlar el teclado. Por tanto, ambos parámetros tendrán que estar a 0, para que no afecten al comportamiento del VCF. En otras palabras, el filtro solo responde a la frecuencia de corte y a la resonancia en esta frecuencia.

Como hemos mencionado, además de las salidas del filtro con los potenciómetros manipulados, previamente habrá que grabar como se comporta el VCF con un señal que le entra sin modificar los parámetros (con la configuración de la Figura 5) y otra señal al final de todo (el *loopback*), que contendrá la señal original más lo que aporta la tarjeta de sonido, lo cual se usará en el análisis para obtener la función de transferencia de cada salida. Es esencial que el *loopback* se grabe cada día después de la calibración del equipo. También es importante grabar el *loopback* para cada señal de control, ya que puede variar en función de qué tipo de audio le entre a la tarjeta. En el caso de la *sweep* variable y fija, el *loopback* tiene que ser grabado con el nivel máximo, ya que es el que está referenciado.

Este se graba usando un cable *jack-jack* que salga de la targeta y entre en ella de nuevo, tal como se ve en la Figura 6, en la cual el *loopback* es el cable negro. En esta misma imagen, el cable blanco es la señal de entrada del filtro, que sale de la tarjeta y el gris es la salida del filtro, que entra en la tarjeta (hay que tener en cuenta que las entradas de *jack* de la fila superior son las salidas y las entradas inferiores son las entradas de la tarjeta). Más adelante mostraremos un diagrama completo del sistema de grabación usado.

Cabe destacar que hay una diferencia entre usar un cable balanceado o no balanceado a la hora de grabar el *loopback*. El primer día de grabación descubrimos que con un cable no balanceado, había una pérdida de, aproximadamente, 6 dB entre la señal y el *loopback*. En cambio, con un cable balanceado, no había ninguna diferencia de niveles. Teniendo esto en cuenta, hay que enfatizar que es necesario realizar todas las grabaciones con cable desbalanceado, ya que el sinte es desbalanceado.

Para contextualizar, vamos a explicar brevemente la diferencia entre cables balanceados y no balanceados. En los primeros se envía la misma señal duplicada, una en fase y la otra en contrafase, de manera que en el punto de destino la señal se ve reforzada y el ruido se elimina invirtiendo la fase de una de las dos señales y sumándolas. El ruido se elimina ya que, si enviamos la señal A + ruido, y la -A + ruido y en el punto final invertimos una fase, tenemos (A + ruido) + [-(-A + ruido)] = A + ruido + A - ruido = 2A

En el segundo caso, se envía únicamente una señal, con lo cual el ruido no se elimina.[13] En la Figura 7, podemos ver la diferencia entre ambos cables (en estas grabaciones usamos cables *jack*). En la izquierda está el cable balanceado y a la derecha, el no balanceado.

Antes de entrar en el siguiente apartado, en la Figura 8 tenemos una imagen del sistema de grabación que hemos usado para todas las grabaciones (a la izquierda) excepto las correspondientes a la pregunta 9 (a la derecha). Enviamos la señal de test desde el ordenador a la tarjeta de audio y desde ahí, la enviamos directamente al VCF. A continuación, la salida hace el recorrido inverso, vuelve a la tarjeta y de ahí al ordenador. En la derecha vemos el mismo montaje pero añadiendo la fuente de voltaje externa, correspondiente a la pregunta 9.

4.2. Procesamiento de señales

Una vez tengamos todas las grabaciones hechas, habrá que realizar cierto procesamiento de señales para poder analizar los parámetros de la frecuencia de corte y la resonancia. Este consiste en coger la inversa del filtro (que está generada por el *plugin* de *Aurora* y que es independiente del nivel, por lo que es igual tanto para la *sweep\_static* como para la *sweep\_variable*) y convolucionarla con la señal *output* (la que sale del VCF). El resultado de esta convolución es la respuesta impulsional (IR) de esa salida. Para cada salida, su IR consiste en un audio de nivel de 0 dB con un único pico muy corto y muy fuerte (un impulso) al final. Calculando su transformada de *Fourier*, la función de transferencia y aplicando ciertas operaciones que veremos en el apartado 5, podemos calcular los valores de frecuencia de corte, pendiente, ancho de banda, etc. para cada grabación.

Los archivos que se generarán tendrán una nomenclatura específica, la cual será la misma que la de las salidas, pero en vez de VCF, empezará con IR. Por ejemplo *IR\_16F\_2R\_sweepstat*, correspondiente a la IR de la salida obtenida con el potenciómetro de la frecuencia de corte en 16Hz y el de la resonancia a 2, y habiendo enviado la *sweep* estática como señal de entrada. Veremos en el siguiente capítulo, que el primer paso es separar las IRs de las salidas, para tener una sola IR en cada archivo.

El mencionado filtro inverso es, simplemente, la inversión temporal de la señal de entrada, ecualizado con una pendiente de 6dB/octava [14]. No tiene nada que ver con el filtro VCF.

4.3. Función de transferencia

La función de transferencia describe la relación entre la señal de salida y la de entrada en términos de voltaje, tal como se muestra en la Figura 7. Se puede descomponer en módulo y fase [15].

En el análisis frecuencial de sistemas representa la amplitud de la salida en función de la frecuencia de la señal de entrada. Por ejemplo, la función de transferencia de un filtro electrónico es la amplitud del voltaje en la salida en función de la frecuencia de una onda sinusoidal de amplitud constante aplicada a la entrada.[16]

En nuestro caso, la función de transferencia de una salida es la transformada de *Fourier* de la IR de esta salida dividida entre la transformada de *Fourier* de la IR del l*oopback*. Al resultado le podemos calcular la magnitud y la fase, y esa magnitud es la que usamos para calcular los parámetros, tal como veremos en el apartado 5.

Aquí también obtendremos una serie de archivos, con una nomenclatura concreta. En vez de empezar con VCF o con IR, estas figuras (en este caso, los archivos no son audios, sino figuras) empezaran con las letras TF (*Transfer Function*). Por ejemplo *TF\_512F\_5R\_sweepvar\_ampmid,* que será la función de transferencia de la IR obtenida a partir de la salida generada con el potenciómetro de la frecuencia de corte a 512 Hz y el de la resonancia en el punto 5 y con la *sweep* variable como señal de control. Aquí añadimos otra parte en el nombre (*ampmid*) que indica que se trata de la *sweep* de amplitud media (recordemos que la *sweep* variable contenía 3 sweeps, cada una más baja que la anterior).

**5. Análisis de las muestras**

En este apartado vamos a analizar las grabaciones obtenidas en el apartado anterior, con tal de conocer el comportamiento real del filtro. A continuación, mostraremos los pasos que seguiremos y los conceptos que analizaremos, concretando a cuál de las preguntas que hemos planteado en el apartado anterior corresponde cada paso. Esto nos servirá como guía para esta sección:

Analisis global

* + Estimación de la THD con el sistema de muestreo calibrado

(Pregunta 6: ¿Cuál es el nivel de THD del filtro a la salida?)

* + Estimación de la SNR

(Pregunta 5: ¿Cuál es la SNR del filtro? )

* + Estimación del ruido de fondo
  + Estimación del rango dinámico

Análisis de cada parámetro

* Análisis del control frecuencia
  + Tipología del filtro
  + Rango en frecuencia de corte (Pregunta 2: ¿Cuál es el rango de la frecuencia de corte? )
  + Desviación en frecuencia de corte
  + Orden del filtro (Pregunta 7: ¿Cuál es su pendiente real?)
  + Respuesta del control de frecuencia: lineal, logarítmico, exponencial.

(Pregunta 4: ¿Cuál es el comportamiento de los parámetros? Lineal, logarítmico, exponencial? )

* Análisis de control de resonancia
  + Factor Q y ancho de banda para cada frecuencia central

(Pregunta 8: ¿Cuál es el rango de la Q definida por la resonancia? )

* + Amplitud para cada nivel de resonancia
  + Respuesta de resonancia

(Pregunta 3: ¿Cuál es el rango de la resonancia?)

* Análisis de los controles con *sweeps* variables

(Preguna 1. Hay diferencias en la respuesta en frecuencia del filtro para diferentes niveles de señal de audio de entrada?)

* Análisis de los controles por *Control Voltage* (CV)

( Pregunta 9: ¿Responden igual los parámetros controlados por CV?)

* + Frecuencia de corte y resonancia para diferentes niveles de amplitud de la señal de control.

5.1. Análisis global

Para el análisis global, usaremos las grabaciones correspondientes a las salidas del filtro con la señal de entrada de los tonos, ya que con señales puras es más fácil de ver la distorsión y el ruido. En las siguientes figuras, observamos las diferencias entre la señal de entrada y la de salida. Cabe recordar que estas salidas se grabaron con la frecuencia de corte al máximo y la resonancia al mínimo, es decir, con *bypass*.

Viendo las Figuras 10 a 17, la primera observación que podemos hacer es que con el tono puro como señal de test hay una diferencia de unos 3 dB entre la entrada y la salida. Y cuanto más alta es su frecuencia, mayor es esta diferencia. Esto es debido a que usamos cables desbalanceados para las grabaciones y a que, probablemente, el propio filtro tiene tendencia a reducir el nivel de las frecuencias agudas. También podemos ver que hay algo de ruido y poca distorsión armónica. Ahora veremos más en detalle estas dos últimas cuestiones

Estimación la THD con el sistema de muestreo calibrado

(6. ¿Cuál es el nivel de THD del filtro a la salida?)

La Distorsión Armónica Total (THD) es una medida de cuánto el sistema (en este caso, el VCF) está distorsionando la forma de onda de la señal de entrada. Siempre está presente, pero demasiada distorsión puede causar problemas [17].

La THD consiste, esencialmente, en la deformación de la señal original a causa de la aparición de armónicos (frecuencias múltiples de la fundamental). La causa de estos efectos es el paso de la señal a través de un sistema no lineal como puede ser el filtro controlado por tensión. El THD aparece en dos magnitudes, corriente (THDI) y voltaje (THDV). En nuestro caso, obviamente, se trata del segundo caso.

Las Figuras 10, 11, ..., 17 representan el espectro frecuencial de la entrada y la salida. En ellas podemos ver claramente un ejemplo de THD (el armónico que causa la distorsión es el de 20 kHz). En la Figura 18 [17], podemos observar el efecto que tiene la THD, donde la onda verde hace referencia al tono de la señal de entrada, la roja es el armónico que aparece al pasar por el filtro y la negra es la señal de salida. Se puede observar claramente que la señal de salida está deformada respecto a la entrada.

Matemáticamente, la THD se calcula como el porcentaje del cociente de la suma del nivel de cada armónico respecto al nivel de la frecuencia fundamental [18]. Dado que tenemos diferentes señales de entrada, podemos calcular la THD para cada uno de ellos y después hacer la media aritmética para conocer el porcentaje de THD del filtro. No obstante, si nos fijamos en las figuras 10-17 vemos que únicamente aparece distorsión armónica cuando enviamos el señal de 10 kHz (en el que aparece el 1r armónico (20 kHz)), con lo que la THD del filtro será la THD de este tono:

THD\_tone10k = (suma nivel armónicos / nivel frecuencia fundamental)\*100 = - 64 dBFS / -15 dBFS

Para poder operar con dB cómodamente, es preferible convertirlos a magnitudes lineales, tales como voltaje o potencia. En este caso, lo más sencillo es convertir los dBFS en dBu y de dBu a Voltios, y lo hacemos de la siguiente manera [19]:

dBu = dB FS + 24 i V = 10 \*\* (dbu / 20) \* 0.775

(- 64 dBFS / -15 dB FS)\*100 = (-40 dBu / 9 dBu)\*100 = (0.00775 V / 2.184 V ) \*100 = **0.35 %**

El resultado final muestra lo que vemos en las figuras 10 a 17, la THD del filtro es muy baja, y únicamente aparece con frecuencias altas, a partir de 1 kHz. Dado que hemos tomado muestras de frecuencias específicas, no podemos determinar con exactitud a partir de que frecuencia empezamos a ver una distorsión. Para ello debemos analizar también las *sweeps*. Observando algunas de las salidas obtenidas con *sweep* estática o variable hemos visto que, concretamente, la distorsión armónica aparece a partir de los 2 o 3 kHz, tal como vemos en la figura 19. En este caso, no hemos observado las IRs de las salidas, simplemente el espectro frecuencial de estas.

Estimación la SNR

(5. ¿Cuál es la SNR del filtro? )

El SNR (*Signal to Noise Ratio*) es la diferencia entre el nivel rms de la señal y el nivel rms del ruido que aporta el sistema y se mide en decibelios. Lo podemos ver claramente en las figuras 10 a 17, donde en la salida se ve mucho ruido que en la entrada no estaba. Cuanto más alto es este valor de SNR, mejor, ya que quiere decir que el ruido es muy bajo. Viceversa, si tenemos un valor bajo de SNR significa que hay mucho ruido en el aparato por el que pasa la señal. En este trabajo, cada día de grabación, grabamos también el ruido del filtro, por lo que podemos escribir un sencillo código en *Python* que importe los archivos del *bypass* correspondiente y el ruido y haga los cálculos necesarios.

Matemáticamente, el SNR se calcula del siguiente modo:

SNR = Nivel señal / Nivel ruido

Dado que se mide en dB, esto se convierte en:

SNR = 10\*log(Nivel señal / Nivel ruido) lo cual se convierte en 20\*log(nivel señal/nivel ruido) si trabajamos con voltajes.

Para realizar estos cálculos podemos usar la librería de *Python Essentia* [20], la cual es un conjunto de algoritmos en código abierto que se usan en análisis musical, sonoro y técnico de señales de audio. Concretamente, para este cálculo, usaremos los algoritmos de *Monoloader* (para importar los audios) y RMS (para calcular los niveles rms de la salida y del ruido).

Una cosa importante que hay que tener en cuenta es que el SNR real del aparato será el resultado que obtengamos en cada audio más 12. Recordemos que la señal de entrada está referenciada a -9 dBFS de pico, es decir -12 dBFS RMS. Por otro lado, la señal de entrada podría, sin estar calibrada, llegar a 0 dBFS, con lo cual tenemos este espacio de 12 dB entre la señal que enviamos y el máximo nivel que podríamos enviar. Además, hay que recordar que los cables desbalanceados introducen una diferencia de unos 3 dB entre la entrada y la salida, con lo cual tenemos que sumarle 15 dB al resultado.

1. *Bypass\_tone100*

*RMS\_tone* (dB): -16.33

*RMS\_noise* (dB): -77.27

*SNR* (dB): 75.94

1. *Bypass\_tone440*

*RMS\_tone* (dB): -16.24

*RMS\_noise* (dB): -77.27

*SNR* (dB): 76.03

1. *Bypass\_tone1k*

*RMS\_tone* (dB): -16.20

*RMS\_noise* (dB): -77.27

*SNR* (dB): 76.07

1. *Bypass\_tone10k*

*RMS\_tone* (dB): -18.47

*RMS\_noise* (dB): -77.27

*SNR* (dB): 73.80

El SNR medio del filtro es:

(75.94 + 76.03 + 76.07 + 73.80) /4 = **75.46 dB**

Este es un buen valor para el SNR, ya que quiere decir que hay una diferencia de casi 80 dB entre la señal y el ruido. Esto implica que la señal es perfectamente reconocible y el ruido apenas es audible, cosa que cuadra con lo que podemos observar en las Figuras 10-17, donde vemos claramente la forma del tono puro a pesar del ruido. También podemos observar algo que hemos comentado antes, que es que a mayor frecuencia, el nivel del señal disminuye, con lo cual (dado que el ruido se mantiene constante) el SNR también disminuye. Si bien es cierto que se nota más esa diferencia a 10kHz.

* Estimación del ruido de fondo

Para estimar el ruido de fondo, el procedimiento más intuitivo consiste en restar las dos señales (el *bypass* y el tono de entrada correspondiente) y el resultado será el ruido, que es lo que hay de diferente en las dos señales. Para medirlo, implementaremos un *script* que haga la transformada de *fourier* de ambos audios para poder restar la magnitud de los espectros frecuenciales, o, que es lo mismo, hacer la división logarítmica de ellas. Para ello utilizaremos de nuevo la librería *Essentia* [20] .

En la figura 20 podemos ver los resultados de este pequeño código. Observamos que en las frecuencias en las que debería estar el tono puro, hay menos ruido (excepto en la última figura, en la que el ruido aumenta). En las figuras 10-17 parece que el ruido es constante en todas las frecuencias,pero no es así. Vemos, sobre todo, que aparece mucho ruido en las altas frecuencias y que el punto en que se reduce (cuando hay el tono) es desplazado, es decir, que el tono no está exactamente en la frecuencia donde debería estar). Esto es parte del comportamiento del filtro, que añade esas interferencias. Seguramente, aunque cada módulo sea independiente de los otros, algo de ruido electrónico se cuela entre ellos, por lo que el VCF puede contener ruido de los VCO o viceversa.

* Estimación del rango dinámico

El rango dinámico es la diferencia de nivel de presión sonora existente entre los sonidos más bajos y los más altos de una señal [21]. Al igual que en el caso de la SNR, también se mide en dBs y también es mejor cuanto más alto es su valor. Para este cálculo también hemos usado el *software essentia*, rehusando el mismo código que el que usamos para calcular el SNR. También cogeremos el valor rms del ruido ya que, estando en un nivel de decibelios tan bajo (menos de -70 dB), no hay diferencia entre coger un valor de -75 que uno de -114. Auditivamente no se nota la diferencia, dado que es una escala logarítmica. Por otro lado, también es necesario añadir 15 dBs, por el mismo motivo que había que hacerlo con el cálculo del SNR.

1. *Bypass\_tone100*

*RMS\_noise* (dB): -77.27

*peak* (dB): -12.95

*Dynamic Range* (dB): 79.31

1. *Bypass\_tone440*

*RMS\_noise* (dB): -77.27

*peak* (dB): -12.88

*Dynamic Range* (dB): 79.39

1. *Bypass\_tone1k*

*RMS\_noise* (dB): -77.27

*peak* (dB): -12.86

*Dynamic Range* (dB): 79.41

1. *Bypass\_tone10k*

*RMS\_noise* (dB): -77.27

*peak* (dB): -15.16

*Dynamic Range* (dB): 77.11

DR\_total = (79.31 + 79.39 + 79,41 + 77.11)/4 = **78.805 dB**

Vemos que el resultado es muy próximo al del SNR, concretamente, hay 3 dB de diferencia, que es la diferencia entre el nivel de pico y el de rms de una señal. También vemos que hay el mismo patrón en cuanto al nivel del *bypass*. Éste es más bajo a 10kHz.

5.2 Análisis de cada parámetro

Como hemos mencionado, el VCF tiene dos parámetros principales, la frecuencia de corte del filtro y la resonancia. No obstante, hay diversos parámetros secundarios, pero no por eso menos importantes. Estos son la pendiente del filtro (cuando se llega a la frecuencia de corte, el sonido no es repentinamente eliminado, sino que es un proceso gradual), el ancho de banda de la resonancia (que frecuencias ocupa la campana, va desde la f1 hasta la f2), la ganancia (la diferencia entre el máximo nivel de esta resonancia y el nivel de la referencia), la frecuencia central y la de resonancia (en que frecuencia se encuentra el nivel máximo de la resonancia, no siempre coincide con la frecuencia central) y, finalmente, el factor Q (indica la forma de la señal en la resonancia, cuanto más alto es este valor, más estrecha es la forma).

.

Para responder el resto de preguntas y analizar más a fondo los parámetros de frecuencia de corte y resonancia, es necesario obtener y observar la magnitud de la función de transferencia de cada salida (tal como explicamos en el apartado 4.3).

Para tener los resultados de manera más clara y ordenada usaremos una série de tablas (una para cada pregunta) con los valores de los potenciómetros usados en las filas y los parámetros que queremos analizar como columnas (frecuencia de corte, pendiente, Q, f1 y f2, frecuencia central, pico y frecuencia de resonancia). Pero antes de empezar a hablar de resultados, vamos a explicar con detalle cómo llegamos a ellos.

Para empezar, recordemos que cada respuesta impulsional que tenemos en este momento contiene varias. Es el caso de las IR que provienen de las *sweeps* estática y variable (preguntas 1, 2, 3 y 4). Recordemos que su nombre es, por ejemplo, *VCF\_16F\_allR\_sweepstat* (*IR\_16F\_allR\_sweepstat)* o bien *VCF\_64F\_2R\_sweepvar (IR\_64F\_2R\_sweepvar).* En el primer caso, el archivo contiene 11 IRs, una para cada resonancia, y en el segundo, cada archivo contiene 3 IRs, una para cada amplitud.

El primer paso es encontrar un método para separar esas IRs de manera uniforme y que cada archivo contenga una sola respuesta impulsional. El método que usaremos se implementará en el script *Separacion\_IRs.py* y es el siguiente:

1. Estimar los picos de la señal con el algoritmo *Peak Detection* de la librería *Essentia*. Es necesario dibujar el tren de IRs para conocer los valores de los parámetros que nos interesa pasarle al algoritmo, que son los siguientes:
   1. *maxPeaks*: determina el número máximo de picos que queremos detectar. Corresponden a cada IR, por tanto, este valor será 11 o 3.
   2. *minPeakDistance*: define la distancia mínima entre cada pico. Sabemos que la distancia entre las IRs es uniforme, por tanto, es suficiente con calcular una y ponerla. Hay que tener en cuenta que el algoritmo trabaja con distancias de 0 a 1, con lo cual hay que aplicar una sencilla regla de tres para convertir la distancia real (medida en muestras) en esta escala de 0 a 1.
   3. *threshold*: define el umbral por debajo del cual no se detectarán los picos. Tiene que estar por debajo de la IR más pequeña
2. Una vez conocidos los picos, conocemos el instante de tiempo en el que se encuentra cada IR. Esta puede contener cola o un pre-eco con lo que hemos de asegurarnos que no cortamos demasiado.

La diferencia entre cada pico nos da un intervalo temporal T que podemos utilizar para seccionar, cogiendo la señal entre el instante del pico p - (T/2) y p + (T/2).

1. Los archivos resultantes de este proceso serán los audios con nombres como *IR\_16F\_2R\_sweepvar\_ampmax* o *IR\_128F\_3R\_sweepstat* en los cuales únicamente hay una IR, correspondiente a la salida con los potenciómetros y la señal de control indicados en el nombre.

Una vez tenemos todas las IRs separadas, hay que comprobar que no hay diferencia de magnitud entre la salida y la entrada, es decir entre las IR correspondientes a las salidas del filtro (*output*) y las IRs de los *loopbacks* (*inputs*).

Hemos visto que en algunos casos, sí que hay una diferencia, y dado que es un patrón muy evidente, podemos decir que se trata de un comportamiento propio del módulo.

Estos son los resultados de esta comparación entre *output* e *input* (no los pondremos todos, sino solo los que conllevan un cambio respecto los anteriores) :

Pregunta 1: Vemos que a mayor frecuencia de corte, menos diferencia hay y que ni el nivel (si la *sweep* de entrada es de amplitud máxima, media o mínima) ni la resonancia afectan demasiado.

La diferencia general es la misma.

* *IR\_16F\_0Res\_sweepvar\_ampmax*: > 50 dB
* *IR\_64F\_0Res\_sweepvar\_ampmax*: > 40 dB
* *IR\_512F\_2Res\_sweepvar\_ampmax*: 20 dB
* *IR\_4096F\_5Res\_sweepvar\_ampmax*: 0 dB

Pregunta\_9: Igual que antes, a mayor frecuencia, menor diferencia. Pero a mayor resonancia, también hay menos diferencia. El patrón es el mismo tanto si enviamos *sweep* estática como si enviamos *sweep* variable (como antes, el nivel no afecta a la diferencia)

* *IR\_CV\_freq\_1V\_sweepvar\_ampmax*: 50 dB
* *IR\_CV\_freq\_4V\_sweepvar\_ampmax*: > 30 dB
* *IR\_CV\_freq\_7V\_sweepvar\_ampmax*: < 20 dB
* *IR\_CV\_freq\_10V\_sweepvar\_ampmax*: 5 dB
* *IR\_CV\_res\_1V\_sweepvar\_ampmax*: > 20 dB
* *IR\_CV\_res\_4V\_sweepvar\_ampmax*: 20 dB
* *IR\_CV\_res\_7V\_sweepvar\_ampmax:* < 20 dB
* *IR\_CV\_freq\_1V\_sweepstat*: > 50 dB
* *IR\_CV\_freq\_4V\_sweepstat*: > 30 dB
* *IR\_CV\_freq\_7V\_sweepstat*: 20 dB
* *IR\_CV\_freq\_10V\_sweepstat*: 5 dB
* *IR\_CV\_res\_1V\_sweepstat*: > 20 dB
* *IR\_CV\_res\_4V\_sweepstat*: 20 dB
* *IR\_CV\_res\_7V\_sweepstat* < 20dB

Preguntas\_2, 3 y 4: El patrón es el mismo que en los casos anteriores, a mayor frecuencia de corte, hay menos diferencia.

* *IR\_16F\_0R\_sweepstat*: > 50 dB
* *IR\_32F\_0R\_sweepstat*: 50 dB
* *IR\_64F\_0R\_sweepstat*: < 40 dB
* *IR\_128F\_0R\_sweepstat*: < 40-30 dB
* *IR\_256F\_0R\_sweepstat*: < 30-20 dB
* *IR\_512F\_0R\_sweepstat*: < 20-15 dB
* *IR\_1024F\_0R\_sweepstat*: < 20-0 dB
* *IR\_2048F\_0R\_sweepstat*: < 15-5 dB
* *IR\_4096F\_0R\_sweepstat*: < 10-0 dB
* *IR\_8192F\_0R\_sweepstat*: < 5 dB
* *IR\_16kF\_0R\_sweepstat*: < 10 dB

Podemos ver claramente ese patrón del que hablábamos, según el cual a medida que la frecuencia de corte aumenta, la diferencia entre el nivel de magnitud entre la entrada y la salida disminuye. Es posible que haya algún tipo de mecanismo interno del filtro con el cual, cuanto más se filtra la señal de entrada, más nivel da a la salida para mantener el nivel de sonoridad. Es decir, que, básicamente, la magnitud se incrementa en la banda pasante del filtro. Conociendo este comportamiento, podemos compensar este incremento para poder realizar los cálculos con más precisión y que la función de transferencia esté situada alrededor de los 0 dBs (ha de ser así porque la función de transferencia muestra la relación entre la entrada y la salida, y queremos que esta diferencia sea mínima). Por este motivo, un paso importante que hay que hacer una vez calculemos la función de transferencia de cada salida, es encontrar la región plana de ésta, antes de la frecuencia de corte o de la resonancia, para saber cuánto exactamente tenemos que restarle a la función para compensar esta ganancia extra.

El siguiente paso es importar las IRs necesarias para calcular la función de transferencia y, con ella, los parámetros del filtro, en un nuevo *script* (llamado *Transfer\_Function*). Éste contiene un código que calcula una única función de transferencia (TF). La función que hace este cálculo recibe 2 parámetros, la *IR\_input* (la respuesta impulsional del *loopback*) y la *IR\_output* (la respuesta impulsional de la salida).

Esas 2 IRs tienen que estar perfectamente alineadas, es decir, con el impulso en las mismas muestras (en dominio temporal), para que no haya desfases entre ellas, y han de tener el mismo tamaño, con tal de poder calcular la función de transferencia.

Para alinear las IR, simplemente encontramos la posición exacta del impulso de cada una, usando *np.argmax,* calculamos la diferencia entre los dos picos y recortamos la señal de la IR con el pico más alejado (más agudo). A continuación nos aseguramos que las 2 tienen el mismo tamaño recortando otra vez las señales para que tengan el tamaño de la IR más pequeña. Además, hay que tener en cuenta que el algoritmo del cálculo de la FFT de *Essentia* [20] solo funciona si el tamaño es par, de modo que también hay que asegurarse que este tamaño es par. Por último, antes de calcular la TF, puede ser que haya valores que son 0, con lo cual la división entre la entrada y la salida sería infinito o *nan*. Para que eso no ocurra, usamos *np.finfo(float).eps* para asignar los ceros a valores muy pequeños, y asegurarnos que no hay ceros en ninguna de las 3 IRs.

Una vez hecho todo esto, ya podemos calcular las transformadas de *fourier* de las 2 IRs. Usaremos los algoritmos FFT y *CartesianToPolar* para poder obtener la magnitud y la fase de la función de transferencia. La TF se calcula dividiendo la transformada de *fourier* de la salida (*IR\_output\_fft*) entre la transformada de fourier del *loopback* (*IR\_input\_fft*)).

Finalmente, convertimos la magnitud a dBs, para tener una mejor visión de la TF, usando *20\*np.log10(TF\_mag*) donde *TF\_mag* es la magnitud lineal de la TF y recortamos el señal para tener la región en la que no hay ruido (éste aparece en los extemos de la TF).

Recordemos que hay que compensar la diferencia de magnitud entre entrada y salida y eso lo haremos reduciendo la magnitud de las TF hasta que ambas estén a 0 dB. Para ello, necesitamos conocer la región donde la magnitud es plana y su nivel, la cual es la media del nivel de la señal en la región comprendida entre el final del ruido inicial y la parte en la que empieza a bajar de nivel. Una parte de esta región es común en todas las TF (de 40 a 90 Hz). Para la referencia podemos usar un rango mayor (de 500 a 2000 Hz) ya que tarda más en empezar a decaer. Esta operación la realizaremos en el *MAIN.py*, antes de llamar a los *scripts* en los que calculamos los valores de los parámetros de la resonancia y la frecuencia de corte.

Una vez tenemos una función para calcular la TF, podemos crear dos *scripts* más (uno para frecuencia y otro para resonancia), en los cuales calculamos los parámetros y guardamos tanto las figuras como los resultados. Para hacerlo de manera más eficiente, hay que automatizar y generalizar este código, es decir, que él mismo coja todos los archivos de una carpeta con IRs y haga todos los cálculos. Para ello, usaremos la librería *Pathlib* [21] en el *script* *MAIN.py*.

Esencialmente, lo que hacemos con esta librería es definir un directorio, usando *Path(“./AUDIOS\_TFG/IRs\_separadas/Preguntas\_2,3i4”)*, donde están las IR de las salidas, del *bypass* y del *loopback*.

A continuación podemos obtener una lista de todos los archivos *.wav* que hay en este directorio, recorrer esta lista mediante un *for* y clasificar las salidas en 4 listas (*Loopback, Bypass, \_0R\_, otros*). En la lista *\_0R\_* van todos los archivos con resonancia 0, que se usarán para calcular los parámetros de la frecuencia en el *script* correspondiente. Es importante calcular estos parámetros (frecuencia de corte y pendiente) con salidas sin resonancia, porque ésta tiende a deformar el filtro y, por tanto, la frecuencia de corte y la pendiente se ven afectados y cambian) . Y en la lista de *otros*, estarán los archivos con resonancia, que se usarán en el *script* correspondiente para calcular los parámetros de la resonancia. Después de esta clasificación, simplemente llamamos a las funciones de *transfer\_function, frequency y resonance*, cada una implementada en el *script* correspondiente.

Para la salida en *bypass*, únicamente hay que llamar una vez a la función *transfer\_function,* ya que solo hay una en cada directorio de IRs. Después, tanto para llamar a *Frecuencia.py* como a *Resonancia.py*, hacemos un for que recorra la lista de salidas correspondiente (*\_0R\_* o *otros*), calcule la TF y los parámetros correspondientes en cada salida. A continuación, vamos a explicar que pasa en cada código.

* Análisis del control frecuencia

En esta subsección hablaremos del *script* *Frecuencia.py* donde calcularemos el pendiente y la frecuencia de corte de cada salida. Una vez hechos los cálculos necesarios, hay que guardar los gráficos resultantes en un *.png* con el nombre adecuado (sin la *IR,* tal como vimos en el apartado 4.3) y guardar los valores en un documento *.json.* (*JavaScript Object Notation*). Los archivos resultantes llevarán el nombre de *TF\_xF\_xR\_sweepx. png y Results\_xF\_xR\_sweepx.json* , respectivamente, donde *x* es el valor de los potenciómetros correspondientes o si se trata de la *sweep* variable o la estática.

Para estimar la frecuencia de corte, la opción más precisa es hacerlo geométricamente usando el *bypass* (la referencia). Puede ser tan simple como atenuar la magnitud de la TF del *bypass* 3 dB y buscar el punto de corte entre la TF de la salida y la del *bypass*. Para encontrar la intersección entre las dos gráficas, podemos usar la siguiente línea de código:

*fcorte = np.argwhere(np.diff(np.sign(TF\_mag\_out - TF\_mag\_ref)))*

Primero, calcula la diferencia de magnitudes y los signos correspondientes usando *np.sign*. Aplicando *np.diff* i *np.argwhere* *,* conocemos las posiciones donde cambia el signo (cosa que ocurre cuando ambas gráficas se cortan).

* + Tipología del filtro

Observando los resultados, vemos claramente que se trata de un filtro pasa bajos, ya que deja pasar las frecuencias que están por debajo de la frecuencia de corte y cancela las que están por encima, con un cierto pendiente.

* + Rango en frecuencia de corte (2. ¿Cuál es el rango en frecuencia? )

El rango de la frecuencia de corte lo podemos conocer observando los resultados de las preguntas 2,3 y 4, es decir, los resultados producidos por el filtro con la *sweep* estática como señal de entrada (ya que ésta es la señal de referencia), con la frecuencia de corte al mínimo y al máximo (sin contar el valor de 16 kHz, que corresponde al *bypass*).

* *TF\_16F\_0R\_sweepstat.png*: 30.64 Hz
* *TF\_8192F\_0R\_sweepstat:* 25.95 KHz
  + Desviación en frecuencia de corte

El rango teórico del potenciómetro de la frecuencia de corte es de 16 Hz a 16 kHz. Sin embargo, después de observar los resultados de los códigos producidos por las bases de datos, vemos que, en realidad, este rango va desde los 31 Hz hasta los 26 KHz de modo que hay una notable desviación.

En frecuencias graves esta es, únicamente, de 15 Hz, mientras que a altas frecuencias, la desviación alcanza los 10 KHz. Por otro lado, podemos decir claramente que cuando el potenciómetro se encuentra en la posición de 8kHz (el penúltimo punto), el filtro no actúa o, más bien dicho, no notamos que actúe, dado que el rango auditivo para los seres humanos se encuentra por debajo de estos 26 kHz. En la Tabla 1, podemos ver los valores concretos de cada punto del potenciómetro de frecuencia, y vemos que, efectivamente, a mayor frecuencia, se produce una mayor es esa desviación. Esto puede ser debido a que cuanto más se quiere filtrar el sonido que entra al filtro, menos control se tiene sobre la frecuencia en la cual empezamos a eliminar el sonido.

* + Orden del filtro(Pregunta 7: ¿Cuál es su pendiente real? )

El orden del filtro depende del pendiente real de este. Para calcular el pendiente de una salida, usamos las siguientes líneas de código

*if fcorte < 80000:*

*pendiente = (TF\_mag\_out[int(fcorte)]-TF\_mag\_out[int(fcorte\*4)])/2*

*elif (fcorte>80000)and(fcorte<210000):*

*pendiente = (TF\_mag\_out[int(fcorte)]-TF\_mag\_out[int(fcorte\*2)])*

*elif fcorte>210000:*

*pendiente = (TF\_mag\_out[int(fcorte)]-TF\_mag\_out[int(fcorte)+6000])*

Simplemente restamos el nivel de la magnitud en la frecuencia de corte menos el nivel de la magnitud en dos o en una octavas más adelante, y dividimos el resultado para obtener el pendiente por octava. Tenemos que establecer diversas condiciones ya que nuestro rango de análisis va desde 40 a 32 kHz, para poder observar el rango auditivo (de 20 a 20 kHz) y un poco del electrónico. Por eso, si por ejemplo estamos a más de 8kHz y multiplicamos por 4, nos vamos fuera del rango y da error. Por otro lado, hemos observado que en los extremos (hasta 40 Hz y a más de 40kHz) hay mucho ruido que impide realizar correctamente los cálculos. Los valores que marcan los intervalos corresponden con las muestras de la función de transferencia. Más adelante, cuando hacemos el *plot*, creamos un vector nuevo para tener esa TF en función de la frecuencia. Si la frecuencia de corte es mayor a 21kHz, no podemos obtener el pendiente por octava, ya que la siguiente octava está fuera del rango que nos interesa.

Para conocer el valor real de este parámetro, observaremos los valores encontrados con la *sweep* estática y haremos una media de ellos. Recordemos que, igual que la frecuencia de corte, el pendiente únicamente lo hemos encontrado con TF que no contienen resonancia, ya que ésta deforma el filtro y cambia los valores relacionados con la frecuencia.

En la tabla 1, podemos ver los valores de frecuencia de corte y pendiente de cada TF. En la primera columna tenemos el nombre del archivo correspondiente, simplificado. Por ejemplo, el primero seria *Results\_16F\_0R\_sweepstat.json*

En el último caso, no podemos conocer el pendiente por octava, ya que la frecuencia de corte es, en realidad, de 25 kHz y a una octava por encima de este valor, estamos fuera del rango que nos interesa, ya que no lo oïmos y, además, está en una región con mucho ruido.

Aun así, con estos valores podemos conocer el pendiente real, calculando su media, la cual nos da un valor de 10.16 dB/octava, el cual está 14 dB por debajo del valor teórico. Este valor corresponde a un filtro a medio camino entre orden 1 y orden 2.

Algo que debemos mencionar acerca del comportamiento es que a medida que la frecuencia de corte aumenta, el pendiente también lo hace, es decir, que se elimina el sonido de manera más abrupta. Esto tiene relación con lo que hemos hablado de una pérdida de control sobre el cómo se elimina el sonido a medida que aumenta la frecuencia de corte.

* + Respuesta del control de frecuencia: lineal, logarítmico, exponencial.

(4. ¿Cuál es el comportamiento de los parámetros? Lineal, logarítmico, exponencial? )

Para responder a esta pregunta hemos creado un nuevo script, llamado *Filter\_Response.py* en el cual nos proponemos ajustar una curva logarítmica, una exponencial y una lineal a los puntos de la función de transferencia de algunas IR de la *sweep* estática. No es necesario hacerlo con todas, con que hagamos unas pocas ya veremos un patrón que nos ayude a decidir qué comportamiento tiene el filtro. Éste viene determinado por el tipo de curva que se ajusta mejor a nuestra TF. Eso lo podemos ver tanto gráficamente como calculando el MSE (*Mean Squared Error*), aunque independientemente del método, el resultado será el mismo.

Para contextualizar un poco vamos a definir qué significa exactamente ajustar una curva. Éste es el proceso de construcción de una curva o función matemática, que tiene el mejor ajuste respecto a una serie de puntos de datos, es decir, que tiene la forma más parecida posible a estos datos [23], en este caso, a los puntos de una función de transferencia.

Para realizar los cálculos necesarios nos hemos basado en un artículo de *SWHarden* [24] (para ajustar la curva exponencial) y otro de *stackoverflow* [25] (para ajustar la curva logarítmica). Lo primero que hacemos es definir 3 funciones necesarias para el ajuste (una para cada tipo de curva). Cabe mencionar que hemos ajustado una exponencial negativa, ya que su forma es más parecida a la función de transferencia que la exponencial positiva.

En definitiva, estas funciones nos sirven para implementar las ecuaciones de las curvas que queremos ajustar:

* Curva Logarítmica:

donde *x* e *y* son las coordenadas de los datos (*freq y TF\_mag\_*out) y *a* y *b* son los coeficientes de la ecuación logarítmica.

* Curva Lineal

donde *x* e *y* son (*freq* y *TF\_mag\_out* respectivamente) y *a* y *b*  son los coeficientes de la ecuación lineal, que corresponde a la ecuación básica de la regresión lineal.

* Curva Exponencial:

donde *t* es el factor de *decay*, *x* e *y* son *freq* y *TF\_mag\_out* respectivamente y *a* y *b* son los coeficientes de la ecuación exponencial.

A continuación, importamos los audios necesarios para calcular una funcion de transferencia, llamamos a *transfer\_function* y compensamos la ganancia extra. Después calculamos el vector de frecuencias para hacer los *plots* y obtenemos los coeficientes.

En los tres casos, los coeficientes *a* y *b* se obtienen usando la función *optimize.curve\_fit*  de la librería *scipy* [24]. Ésta utiliza mínimos cuadrados no lineales para ajustar una función a los datos, en este caso, las funciones *logFunc, linFunc y* e*xpFunc.*

A continuación hacemos 3 *subplots* donde, en cada uno, *ploteamos* la función de transferencia y la curva ajustada a ella y calculamos el MSE usando la función *sklearn.metrics.mean\_squared\_error* [26]. El MSE se define con la siguiente ecuación:

donde *y* es cada dato real, de la función de transferencia, y es la curva correspondiente.

Finalmente, como con los otros códigos, guardamos los resultados tanto en *png* como en *json.*

En la Figura 21, podemos ver algunos de los resultados obtenidos, donde claramente podemos observar que la curva que más se ajusta es la exponencial, con lo cual, podemos decir que la respuesta del control de frecuencia es exponencial. Esto también lo vemos con el *MSE*, ya que ésta curva es la que lo tiene más pequeño.

* Análisis de control de resonancia

Para los cálculos de los parámetros de la resonancia, implementaremos un nuevo *script*, con el nombre de *Resonancia.py*.

El primer paso es encontrar la frecuencia central y el pico, usando *np.argmax* y *np.max* respectivamente. Estos nos servirán para encontrar los otros parámetros.

Como en el *script* de frecuencia, una vez acabados todos los cálculos, hay que guardar los gráficos y los valores en *.png* y en *.json* respectivamente. De la misma manera que con el parámetro de frecuencia, todos los valores resultados de este análisis, se pueden observar en forma de tabla (Tabla 2).

* + Factor Q y ancho de banda para cada frecuencia central

(8. ¿Cuál es el rango de la Q definida por la resonancia? )

Para encontrar el factor Q, usaremos la siguiente fórmula:

donde *fres* es la frecuencia de resonancia (que no siempre coincide con la frecuencia central y se calcula de manera directa a partir de f1 y f2:

Por otro lado, f1 y f2 determinan el ancho de banda y se calculan de manera parecida a cómo calculamos la frecuencia de corte: se definen como las frecuencias que están 3dB por debajo del pico de la resonancia. Para calcularlas, podemos sumarle el pico al *bypass*, restarle 3 dB, y calcular f1 y f2 como la intersección de la salida con éste, usando la misma línea de código usada para calcular la frecuencia de corte, y cogiendo la primera y la última posición del vector resultante.

En la Tabla 2 podemos encontrar los resultados para estos 6 parámetros (f1, f2, frecuencia central, frecuencia de resonancia, pico y factor Q) y hay varias observaciones que podemos hacer y un patrón que es común para todas las frecuencias de corte.

En resonancia muy baja, solo hay un punto de corte, o varios puntos muy cercanos, ya que el *bypas*s está demasiado por debajo de la salida (el pico es muy bajo, por lo que el *bypass* está, aproximadamente, 3 dB por debajo de la salida. Por eso f1 y f2 coinciden o están muy cerca una de la otra, y eso hace que el factor Q sea muy elevado, porque estamos dividiendo la *fres* (que coincide con las f1, f2) entre el ancho de banda (f2-f1) el cual es un valor muy bajo.

Por otro lado, hemos visto que, concretamente, el ancho de banda aumenta, es decir hay dos puntos de corte claramente definidos, cuando el pico de la resonancia está por encima de los 4 dB, momento en que el *bypass* está por encima del nivel estable de la salida.

El punto del potenciómetro en el cual el pico está por encima de los 4 dB y, por tanto, tenemos una resonancia bien definida varía a medida que la frecuencia de corte aumenta. Cuando estamos a 16F, este punto no llega hasta que tenemos resonancia 7, y en 32F, éste llega en la resonancia 4, después se estabiliza y siempre tenemos una resonancia definida a partir de 3R.

También hay conflicto cuando llegamos a resonancias altas (8R (a partir de 64F) ,9R y 10R). Como explicamos en la sección 3, cuando nos encontramos en resonancia máxima, se produce una oscilación automática, es decir, aparece una onda sinusoidal que no obedece a ningún parámetro ni a ningún control. En la práctica eso significa que la diferencia entre la entrada y la salida es muy grande y en la función de transferencia vemos mucho ruido y una respuesta frecuencial indefinida, si bien con cierta concordancia con la forma que debería tener según la frecuencia de corte en la que estemos, como podemos ver en la Figura 22.

En este punto, el pico es muy alto y estrecho, por tanto, solo hay un punto de corte entre entre *bypass*  y *output*, por lo que tenemos un comportamiento muy parecido al inicial, con f1 y f2 iguales o muy cercanas, *fres* igual muy cercana a f1 y f2, y el factor Q muy alto (que cuadra con una resonancia muy estrecha).

Algo importante acerca de este último punto es que a partir de *8192F*, es decir cuando el potenciómetro de la frecuencia de corte está en el penúltimo punto y en el último, esta resonancia automática desaparece y no vemos ese aumento de pico ni de factor Q. En cambio, dado que a 16kF, la salida se parece mucho al *bypass (*éste corresponde a la salida *16kF\_0R*), solo hay un punto de corte. También vemos que en estos dos puntos del potenciómetro de la frecuencia de corte, los valores de f1,f2,*fres* y fcentral se van fuera del rango de la región sin ruido, por lo cual no los podemos ver en los gráficos de función de transferencia que hemos obtenido.

Una última observación acerca de estos valores es que, tal como dijimos en secciones anteriores, la resonancia deforma la respuesta frecuencial de la TF. Concretamente, vemos que a medida que la resonancia aumenta, la frecuencia central (que equivale a la frecuencia de corte en resonancia 0), así como f1 y f2, se van desplazando hacia la derecha, es decir, los valores son más agudos.

* + Amplitud para cada nivel de resonancia

La amplitud de cada resonancia viene determinada por la ganancia, y esta es, simplemente, la diferencia de nivel entre el pico y el nivel de la TF del *bypass*, la cual es la misma en todo su dominio, excepto en los extremos, donde se reduce y aumenta a causa del ruido e interferencias de estas dos regiones. Dado que el nivel estable del *bypass* es cero, podemos decir que la ganancia es simplemente el pico de cada TF. Igual que hemos hecho antes, vamos a analizar estos valores, junto con los de la frecuencia central, observando la Tabla 2.

En primer lugar, debemos mencionar que, en teoría, la frecuencia de resonancia y la frecuencia central deberían coincidir o estar cerca una de la otra en todos los casos. Sin embargo, dado que la *fres* depende de f1 y f2, y éstas, como hemos explicado antes, no siempre definen un ancho de banda coherente (por ejemplo cuando solo hay un punto de corte entre referencia y salida), este requisito no se cumple en todos los casos.

La frecuencia central, donde se encuentra el pico, debería estar entre la f1 y la f2, eso sucede a partir de un pico mayor a 4 dB, dado que en ese momento la resonancia está bien definida y f1 y f2 son valores coherentes. Antes de llegar a este punto, la fcentral se encuentra fuera del rango, en valores de frecuencia muy pequeños y con un nivel de pico por debajo de 1 dB. A medida que aumenta el punto del potenciómetro de resonancia, el pico va aumentando y la fcentral se va desplazando hacía la derecha.

Un dato muy importante, de cara a mapear los puntos del potenciómetro con valores reales de estos parámetros, es que, a partir de 64 Hz, los puntos (2R,..., 7R) corresponden a valores del doble en el pico. Por ejemplo, 2R corresponde a, aproximadamente, 4 dB de pico, y 4R corresponde a unos 8 dB de pico, y así sucesivamente. Con 1R tenemos menos de 1 dB de pico y en los 3 últimos puntos (8R, 9R y 10R) tenemos un pico mucho más alto que el doble ya que estamos en zonas de oscilación automática.

Por último, podemos observar que cuanto más alta es la frecuencia de corte, más alto es el pico de la resonancia. En otras palabras, cuanto menos actúa el potenciómetro de frecuencia de corte, más actúa el de la resonancia, con tal de compensar y mantener equilibrada la actuación del filtro.

* + Respuesta de resonancia

(3. ¿Cuál es el rango de resonancia? )

Para responder a esta pregunta tenemos que fijarnos en los valores de pico, ya que es el parámetro más estable y el que mejor define cuando la resonancia es perfectamente visible y clara. El ancho de banda y el factor Q también son parámetros muy importantes, pero no són tan estables, como hemos explicado anteriormente, sobre todo en los extremos (a poca resonancia y a mucha resonancia). Observando los valores de pico de los puntos del potenciómetro de resonancia desde 1R hasta 7R, es decir, antes de la oscilación automática, podemos decir que el rango de este potenciómetro va desde 0.30 dB (correspondientes a la salida *TF\_64F\_1R\_sweepstat)*  hasta 15.16 dB (correspondientes a la salida *TF\_512F\_7R\_sweepstat*). Ambas salidas se pueden observar en la Figura 23.

* Análisis de los controles con *sweeps* variables

(Preguna 1. Hay diferencias en la respuesta en frecuencia del filtro para diferentes niveles de señal de audio de entrada?)

Para este apartado, el proceso es el mismo que con la *sweep* estática, únicamente cambia el directorio de donde el código extrae las IRs. Podemos observar los resultados en las Tablas 3 y 4. Podemos sacar diversas conclusiones a partir de estos resultados:

En cuanto al control frecuencial, comparando estos valores con los de la Tabla 1, vemos que tienen un comportamiento muy similar al obtenido con una *sweep* estática.

Sin embargo, vemos dos diferencias importantes entre los diferentes niveles. La primera es que a medida que disminuye el nivel, la frecuencia de corte se desplaza a la izquierda, es decir, es más grave. Esto implica que el valor real se acerca más al valor teórico. También notamos que la pendiente es menor, más suave, y esto implica que el filtro actúa menos, deja pasar más frecuencias, con tal de compensar esta pérdida de magnitud en la entrada.

La segunda diferencia entre niveles es que a medida que este disminuye, hay más ruido en la función de transferencia. En el análisis global vimos que el ruido tiene un nivel constante, es decir se mantiene, independientemente de la señal que entre, ya que es parte del filtro, no de la *sweep.* Sin embargo, dado que el nivel de la señal de entrada disminuye, el SNR también lo hace, y eso provoca que veamos más el ruido, ya que la diferencia entre señal y ruido es menor que con la máxima amplitud. Estas dos diferencias las podemos observar también en la Figura 24, donde se muestran los tres niveles para una frecuencia de corte teórica (correspondientes al 3r punto del potenciómetro de frecuencia de corte).

Respecto al control de resonancia, también aparecen estas dos diferencias que hay en la frecuencia de corte, los puntos de corte con el *bypass* són más graves a medida que el nivel de la *sweep* se reduce, y aparece ruido con amplitudes bajas. No obstante, el comportamiento es muy parecido al de las TF obtenidas con la *sweep* estática.

A medida que la frecuencia de corte aumenta, la resonancia, tarda menos en aparecer y, tanto f1, f2 como la frecuencia central y la de resonancia están donde tienen que estar cuando el pico está por encima de los 4 dBs. También observamos los efectos que tiene la oscilación automática provocada en las máximas resonancias y los valores de f1 y f2 cuando el pico está por debajo de 3 dBs, momento en el que, o bien solo hay un punto de corte o bien las dos funciones coinciden y hay muchos puntos de corte. Y, igual que antes, cada punto del potenciómetro de resonancia se puede mapear al doble de valor en el pico, aproximadamente. Vemos un ejemplo de estas funciones de transferencia en la figura 25.

* Análisis de los controles por Control Voltage (CV)

9. ¿Responden igual los parámetros controlados por CV?

Para este apartado, el proceso es el mismo que con la *sweep* estática y la *sweep* variable, únicamente cambia el directorio de donde el código extrae las IRs. Podemos ver los resultados obtenidos en las tablas 5, 6, 7 y 8.

En la primera vemos los resultados controlando la frecuencia de corte con CV externo y la *sweep* estática como señal de audio de entrada. El pendiente es muy similar con respecto a los valores de la tabla 1 (donde veíamos los valores con los potenciómetros manipulados a mano). En cambio, la frecuencia de corte es mucho más exacta. Esto es lo que esperábamos, ya que con el control modificado manualmente, es posible que haya desviaciones milimétricas a la hora de seleccionar el punto exacto en el potenciómetro, pero un control con CV es mucho más preciso, ya que es la señal de voltaje quien selecciona el punto. En la primera columna podemos observar como hace esta selección, es decir, el mapeo entre el número de Voltios que enviamos con CV y la frecuencia de corte que estamos eligiendo. En el manual del sintetizador [8] vemos que éste puede recibir 1V/octava, y que si enviamos 0 V, obtenemos el primer valor del potenciómetro (16 Hz), con lo cual, con 1V seleccionamos 32 Hz, con 4V seleccionamos 256Hz y así sucesivamente. Vemos un ejemplo de este comportamiento en la figura 26.

También podemos observar que la respuesta en frecuencia no es exactamente la misma, ya que, si lo fuera, la última fila (correspondiente a 16kHz) no debería contener frecuencia de corte, dado que la salida es el *bypass.* Por tanto, podemos decir que el *bypass* de control manual y el de CV son ligeramente diferentes. Obviamente, cabe recordar que esta 5a tabla corresponde a valores de salidas sin resonancia, ya que únicamente podemos aplicar CV a uno de los dos parámetros simultáneamente, al menos si contamos con una única fuente de voltaje externa.

En cuanto al control de resonancia, éste tiene un comportamiento bastante parecido al que veíamos con el control manual. Hasta que el pico no supera los 4 dB, no hay una resonancia bien definida, ya que hay muchos puntos de corte que pueden ser f1. antes de que el nivel aumente y los efectos de la resonancia máxima (oscilación automática) són claramente visibles tanto en los valores como en las gráficas.

No obstante, hay una diferencia importante, que hace referencia al mapeo. Igual que antes, este parámetro recibe 1V/octava y, por tanto, puede recibir un total de 11V. Las correspondencias entre voltios enviados y el punto del potenciómetro seleccionado son triviales y se ven en la primera columna de la tabla 6. Sin embargo,en este caso, cada punto del potenciómetro no coincide con el doble del valor de pico. Si enviamos 1V o 4V, el pico tiene menos que el valor del punto (1V da un pico de menos de 1 dB y 4V dan un pico de menos de 4 dB), y si enviamos 7V o más, entonces el pico es más alto del doble. Esto implica que, con estos voltajes uniformemente separados que elegimos (1V, 4V, 7V, 10V), sólo hay uno (7V) en el que la resonancia está claramente definida y con unos valores correctos (ver Figura 27).

Por otro lado, cabe mencionar que, enviando la *sweep* estática y controlando el parámetro de la resonancia con CV, estos valores se acercan mucho a los obtenidos con la misma señal de entrada y control manual, para la frecuencia de corte de 1024 Hz.

* + Frecuencia de corte y resonancia para diferentes niveles de amplitud de la señal de control.

Enviando la *sweep* variable como señal de entrada, vemos un comportamiento muy similar al obtenido con la misma entrada pero con control manual. El pendiente es muy similar y, de la misma manera, va disminuyendo a medida que el nivel decrece. También la frecuencia de corte real es mucho más precisa y cercana a la teórica a medida que la magnitud se reduce. Además, también vemos mucho ruido a medida que la señal decrece, eso, tal como hemos explicado antes, guarda una estrecha relación con la SNR del filtro.

También observamos un comportamiento y unos valores muy cercanos con el parámetro de la resonancia. Aunque hay algunas diferencias, la primera tiene relación con el mapeo entre el punto del potenciómetro en el que estamos y su pico correspondiente. Igual que con la *sweep* estática, el pico no es el doble del valor del punto sino que es un valor próximo al del punto. Como antes, esto implica que solo hay un voltaje (7V) que causa una resonancia bien definida y con los valores de f1, f2, fcentral, etc correctos.

Con un nivel diferente en la señal de entrada, no hay ningún patrón fijo en el comportamiento del filtro, en algunos voltajes parece que f1 y f2 sean más graves a medida que el señal decrece o que el pico disminuye, y con otros voltajes pasa justo lo contrario o son valores muy próximos. Lo que sí que vemos que se repite es el efecto que tiene el ruido, que se destaca más a medida que la señal decrece y provoca que haya muchos puntos de corte entre la referencia y la salida.

Por último, vemos una clara diferencia con respecto a los resultados con la *sweep*  estática y CV. Antes vimos que con la señal estática, el control de la resonancia con CV correspondía a la frecuencia de corte de 1kHz. Sin embargo, con la señal variable y CV, este control se acerca más a la frecuencia de corte de 512 Hz.

**6. Conclusiones**

El objetivo principal de este trabajo era realizar un análisis exhaustivo y comparar los valores teóricos dados en el manual y en los dos potenciómetros principales del filtro (frecuencia de corte y resonancia) con valores empíricos encontrados con diferentes señales de entrada y manipulando los potenciómetros manualmente y con CV.

Después de haber hecho todo esto, podemos decir que los valores no se corresponden exactamente con los obtenidos con el análisis, sinó que hay ligeras desviaciones. Estas desviaciones pueden deberse a muchos factores, tales como el hecho de que se trata de un sintetizador analógico y muy antiguo por lo que algunos componentes pueden haberse deteriorado con el tiempo. Otro motivo de dichas desviaciones es que el control manual no es del todo preciso, ya que vimos que con el control por voltaje externo, había menos desviaciones, aunque estas nunca desaparecían.

**7. Bibliografia**

[1] “Sintetizador”, 2022. [online]. Disponible en: <https://es.wikipedia.org/wiki/Sintetizador>

(acceso: 8 de Febrero de 2022)

[2] “Proyecto Sintetizador Documentacion Version 1.0”, 2009. [online]. Disponible en: <http://profesores.elo.utfsm.cl/~agv/elo329/1s09/projects/AcevedoBarros/SinteAnalogo.html> (acceso: 18 de Enero de 2022)

[3] C. Meyer, “Learning Modular”, 2016. [online]. Disponible en:

<https://learningmodular.com/glossary/control-voltage/> (acceso: 7 de Febrero de 2022)

[4] “RSF Kobol”, 2021. [online] Disponible en: <https://en.wikipedia.org/wiki/RSF_Kobol>

(acceso: 11 de Setiembre de 2021)

[5] J. Michel Jarre, “RSF Kobol Expander”, 2018. [online]. Disponible en:

<http://www.jarrography.free.fr/details_equipement_audio.php?id_equip=80>

(acceso: 8 de Febrero de 2022)

[6] “Reverb”, 2018. [online]. Disponible en: <https://reverb.com/item/18757272-rsf-kobol-expander-i> (acceso: 8 de Febrero de 2022)

[7] “RSF Kobol Expander”, 2018. [online]. Disponible en: <https://www.vintagesynth.com/misc/rsf_kobol.php> (acceso: 23 de Diciembre de 2021)

[8] “RSF Kobol Manual” , 2017. [online]. Disponible en: <http://yusynth.net/gear/Kobol_RSF_en.html> (acceso: 10 de Enero de 2022)

[9] S. Howell, “RSF Kobol Expander” , 1983. [online]. Disponible en: <http://www.muzines.co.uk/articles/rsf-kobol-expander/5929>

(acceso: 6 de Febrero de 2022)

[10] G. Reid,, “XILS Lab PolyKB”, 2010. [online]. Disponible en: <https://www.soundonsound.com/reviews/xils-lab-polykb> (acceso: 19 de Enero de 2021)

[11] “Edirol\_UA-101\_OM”, 2005. [online]. Disponible en:

<https://www.roland.com/global/support/by_product/ua-101/owners_manuals/39364883-4342-471f-9730-0bf7648914aa/> (acceso: 16 de Febrero de 2022)

[12] A. Farina, “Aurora for Audacity”, 2021. [online]. Disponible en:

<http://www.angelofarina.it/Public/Aurora-for-Audacity/> (acceso: 11 de Setiembre de 2021)

[13] J. Manuel, “Cable balanceado y no balanceado”, 2021. [online]. Disponible en:

<https://www.thomann.de/blog/es/cable-balanceado-y-no-balanceado-en-que-se-diferencian/> (acceso: 19 de Febrero de 2022)

[14] Armelloni, E., Bellini, A., & Farina, A. (2001, Mayo). Not-linear convolution: A new approach for the auralization of distorting systems. In Audio Engineering Society Convention 110. Audio Engineering Society [[PDF]](http://160.78.24.2/Public/AES-110/154-aes110.PDF)

[15] Universidad Miguel Hernandez, “Circuitos electrónicos analógicos”, [online]. Disponible en: <https://sites.google.com/a/goumh.umh.es/circuitos-electronicos-analogicos/transparencias/tema-2> (acceso: 7 de Marzo de 2022)

[16] “Transfer Functions”, 2022. [online]. Disponible en: <https://en.wikipedia.org/wiki/Transfer_function> (acceso: 6 de Abril de 2022)

[17] “Distorsión Armónica Total”, 2017. [online]. Disponible en:

<https://www.seencogroup.com/distorsion-armonica-total-que-paso-con-mi-onda-perfecta/> (acceso: 22 de Marzo de 2022)

[18] “Distorsión Armónica”, 2021. [online]. Disponible en: <https://es.wikipedia.org/wiki/Distorsi%C3%B3n_arm%C3%B3nica>

(acceso: 22 de Marzo de 2022)

[19] X. Lizarraga, “analog digital level calibration”, 2022. [online]. Disponible en:

<https://github.com/xaviliz/analog-digital-level-calibration>

(acceso: 7 de Febrero de 2022)

[20] “MTG: Essentia”, 2006, [online]. Disponible en: <https://github.com/MTG/essentia>

(acceso: 28 de Abril de 2020)

[21] “Rango dinámico”, 2017. [online]. Disponible en:

<https://geekland.eu/que-es-el-rango-dinamico-audio-ventajas/>

(acceso: 28 de Marzo de 2022)

[22] “Pathlib”, 2022. [online]. Disponible en: <https://docs.python.org/3/library/pathlib.html>

(acceso: 28 de Abril de 2022)

[23] “How to do exponential and logarithmic curve fitting in Python?”, 2021. [online]. Disponible en: <https://www.geeksforgeeks.org/how-to-do-exponential-and-logarithmic-curve-fitting-in-python/> (acceso en: 18 de Mayo de 2022)

[24] “Exponential Fit with Python”, 2020. [online]. Disponible en: <https://swharden.com/blog/2020-09-24-python-exponential-fit/>

(acceso: 18 de Mayo de 2022)

[25] “Fit a logarithmic curve to data points and extrapolate out in numpy”, 2018. [online]. Disponible en: <https://stackoverflow.com/questions/49944018/fit-a-logarithmic-curve-to-data-points-and-extrapolate-out-in-numpy> (acceso: 18 de Mayo de 2022)

[26] “sklearn.metrics.mean\_squared\_error”, 2022. [online]. Disponible en:

<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html>

(acceso: 19 de Mayo de 2022)

**8. APÉNDICE**

**FIGURAS**

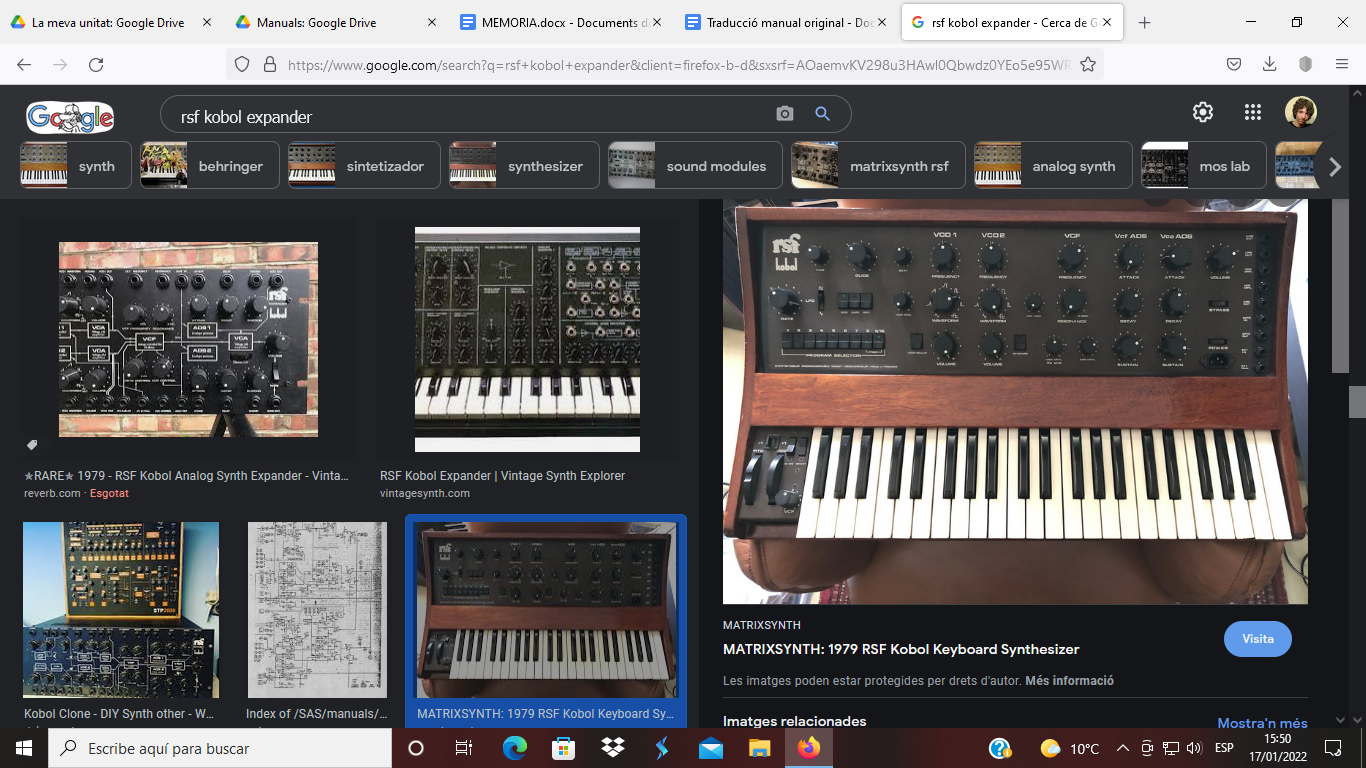


Figura 1: RSF Kobol Expander (completo)



Figura 2: RSF Kobol Expander (sin teclado)

 Figura 3: VCF



Figura 4: PolyKB II



Figura 5: Configuración inicial



Figura 6: Loopback



Figura 7: Balanceado vs No Balanceado



Figura 8: Montaje de grabación

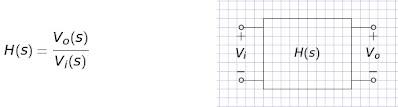


Figura 9: Función de transferencia de un filtro

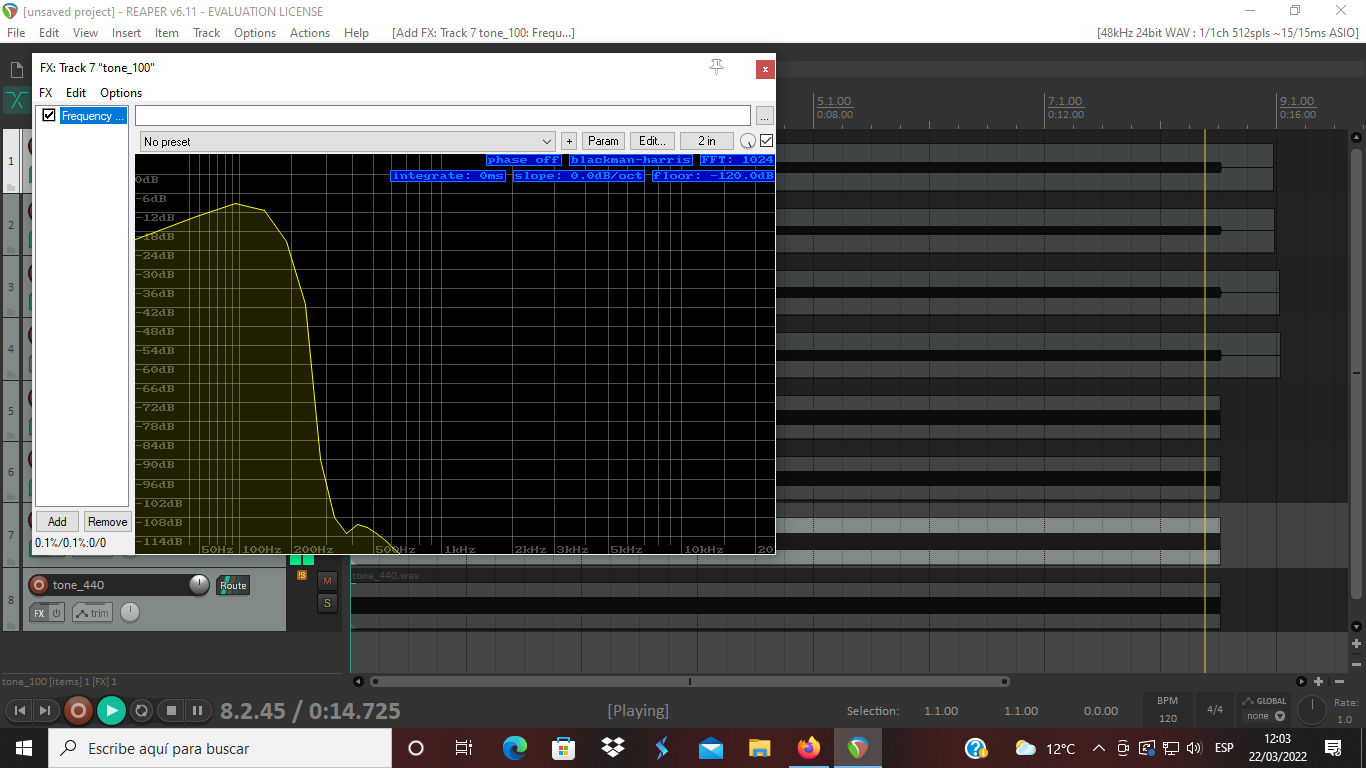
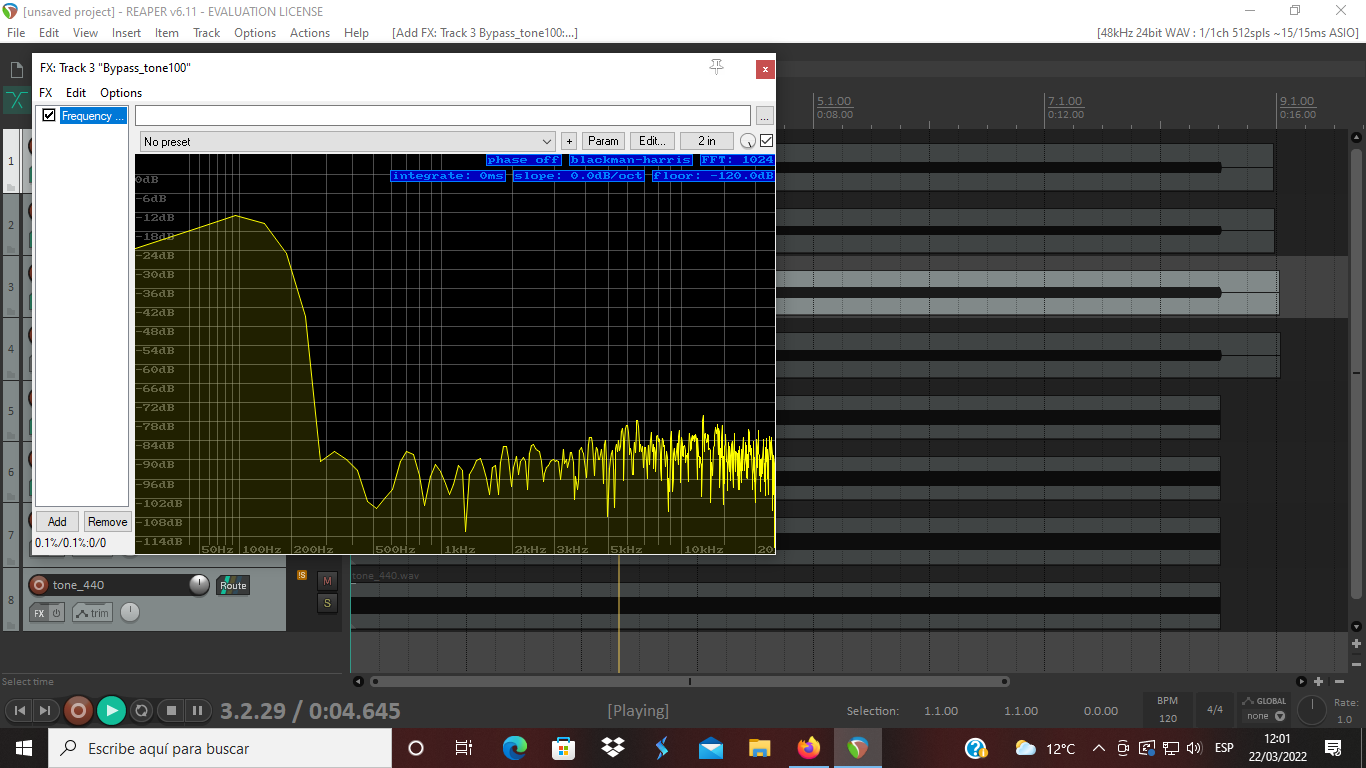
 

Figura 10: tone100 (input) Figura 11: Bypass\_tone100 (output)

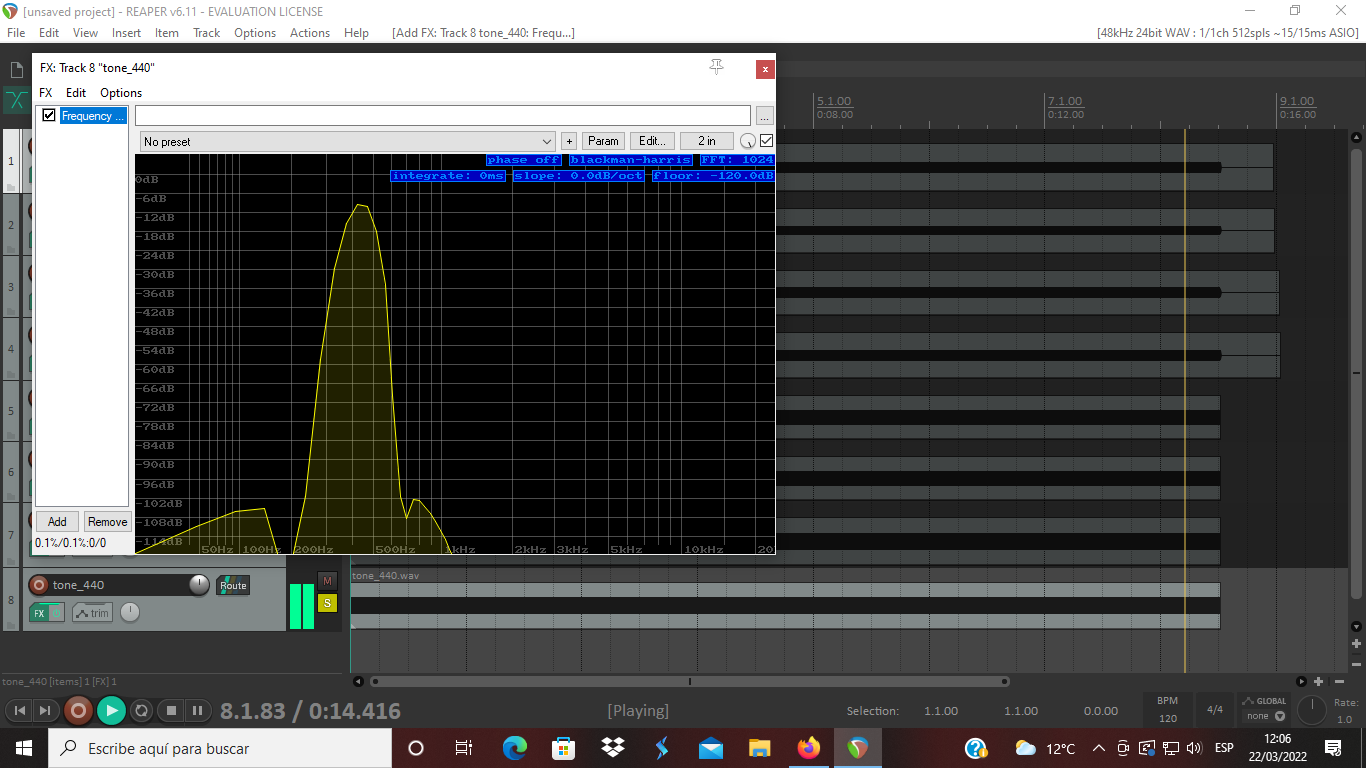
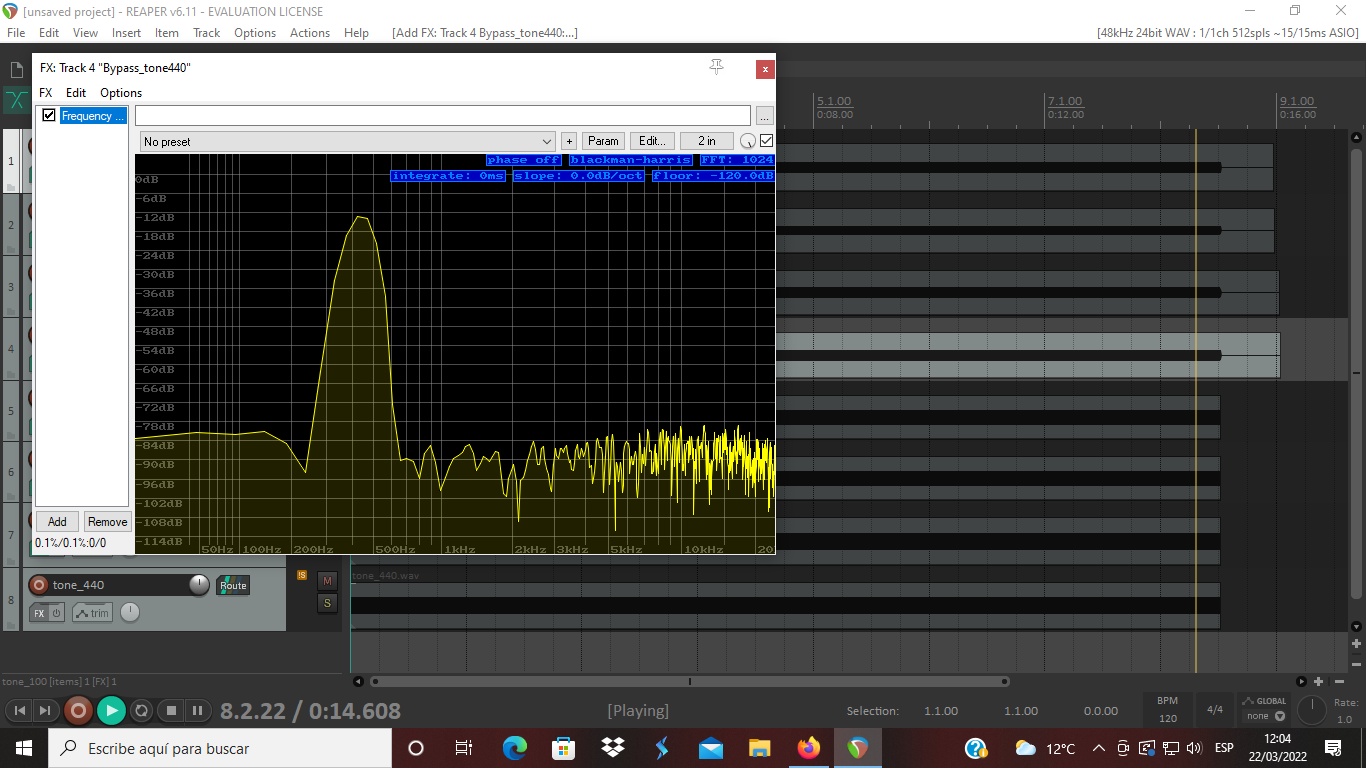
 

Figura 12: tone440 (input) Figura 13: Bypass\_tone440 (output)

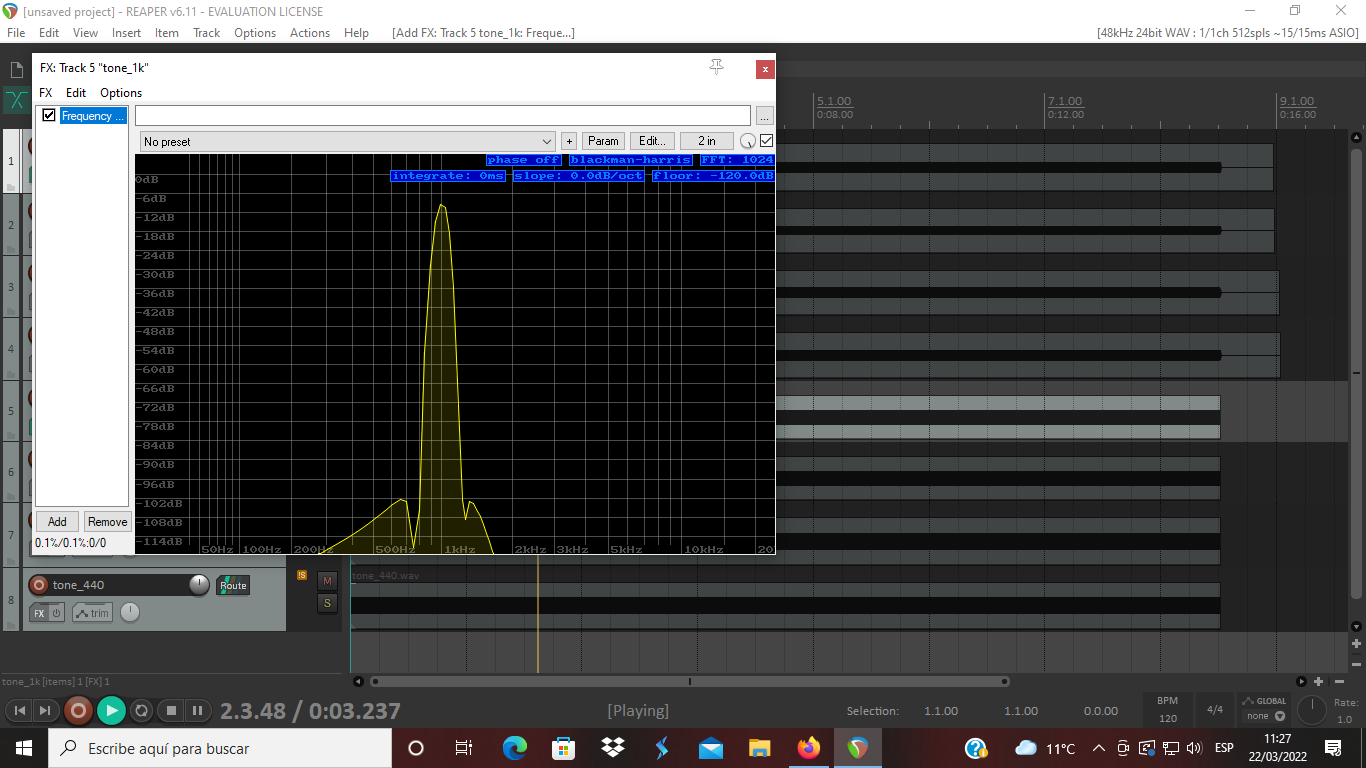
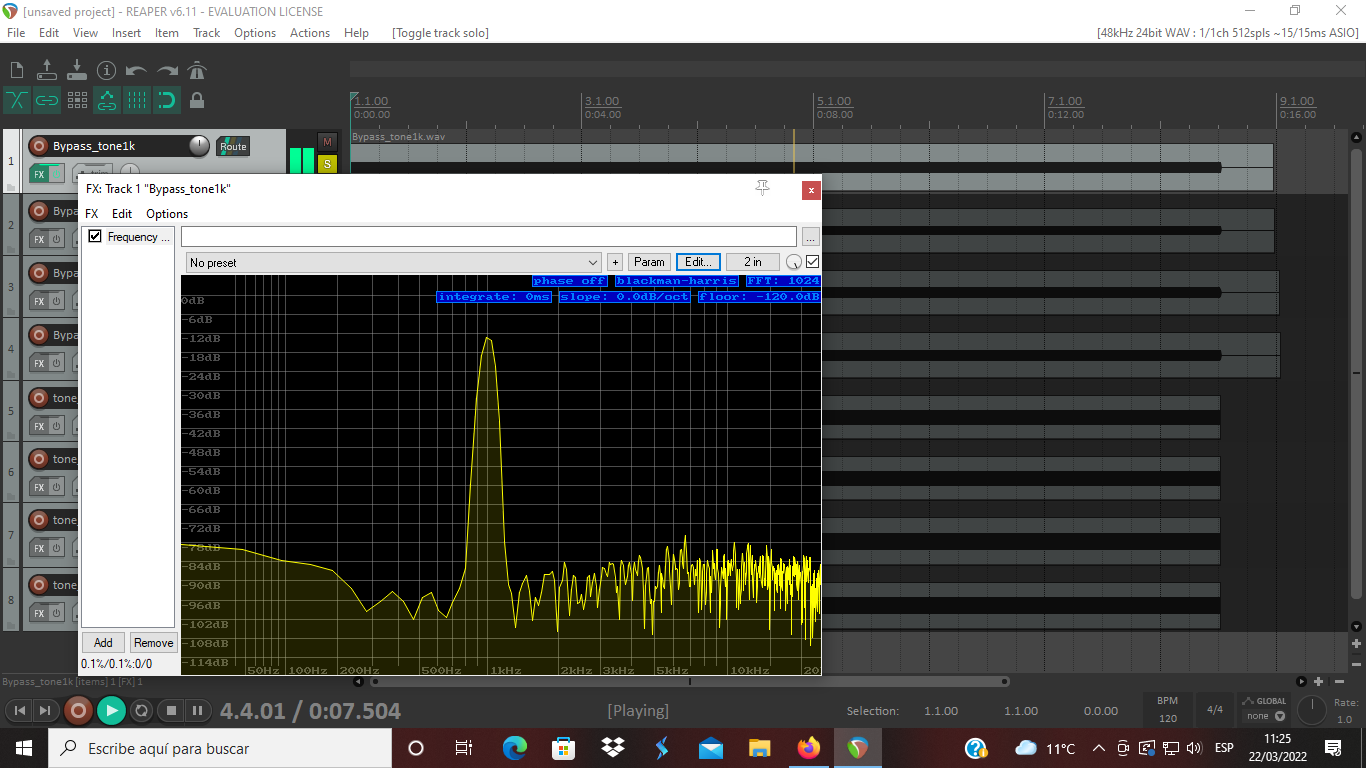
** **

Figura 14: tone1k (input) Figura 15: Bypass\_tone1k (output)

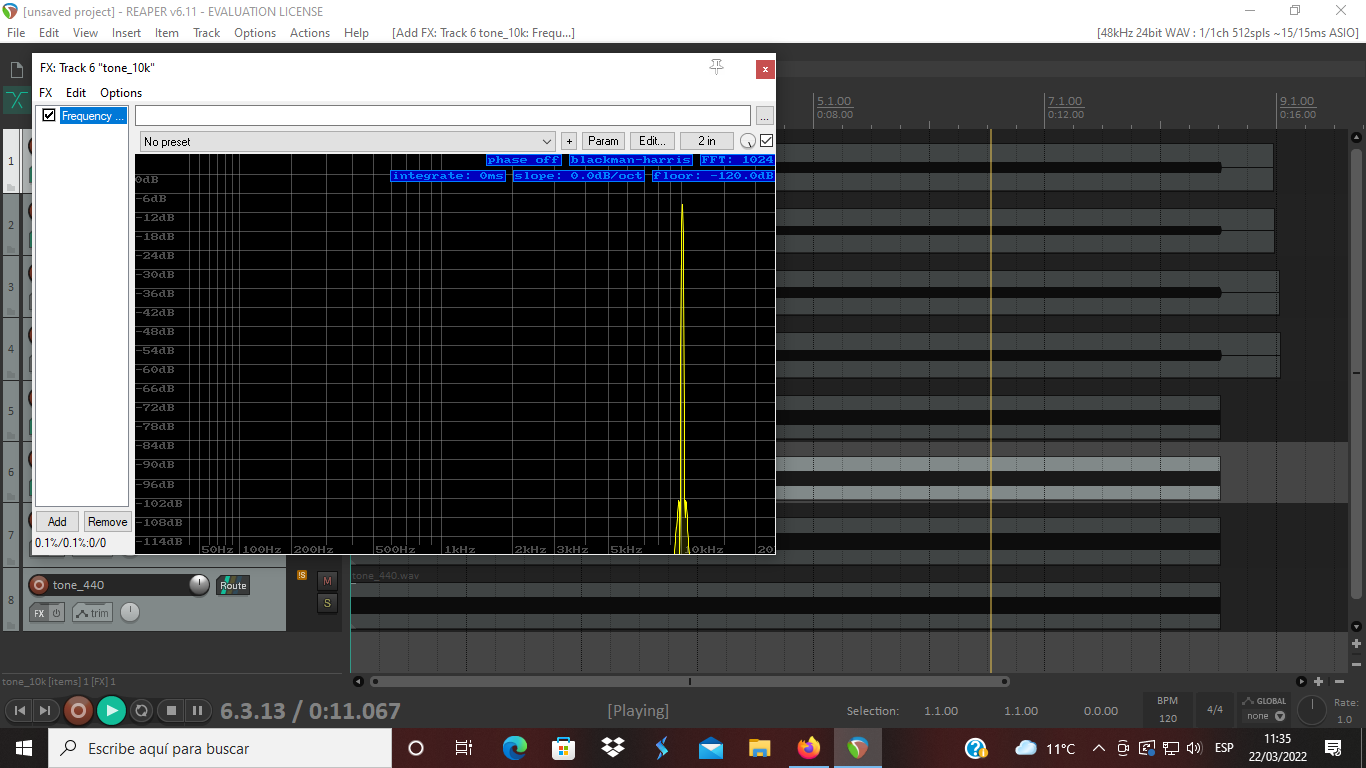
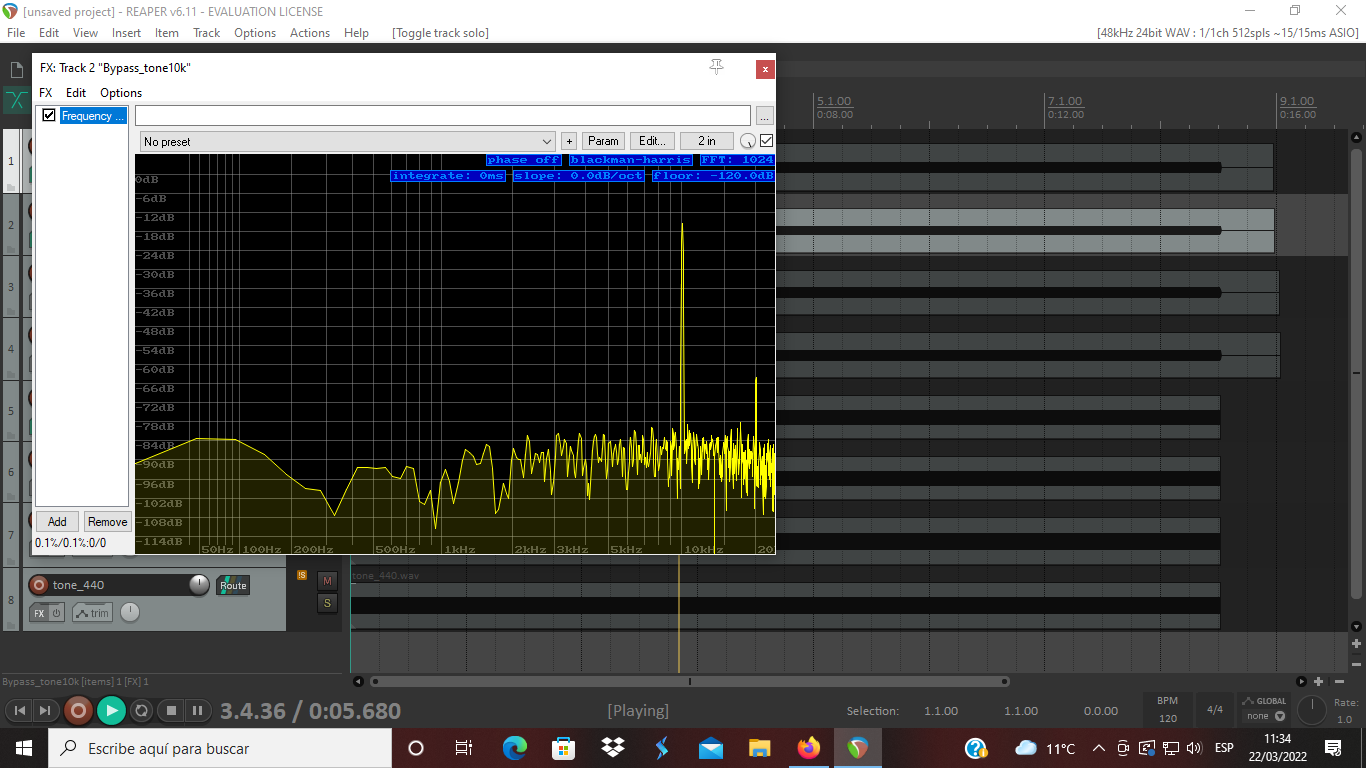
 

Figura 16: tone10k (input) Figura 17: Bypass\_tone\_10k (output)

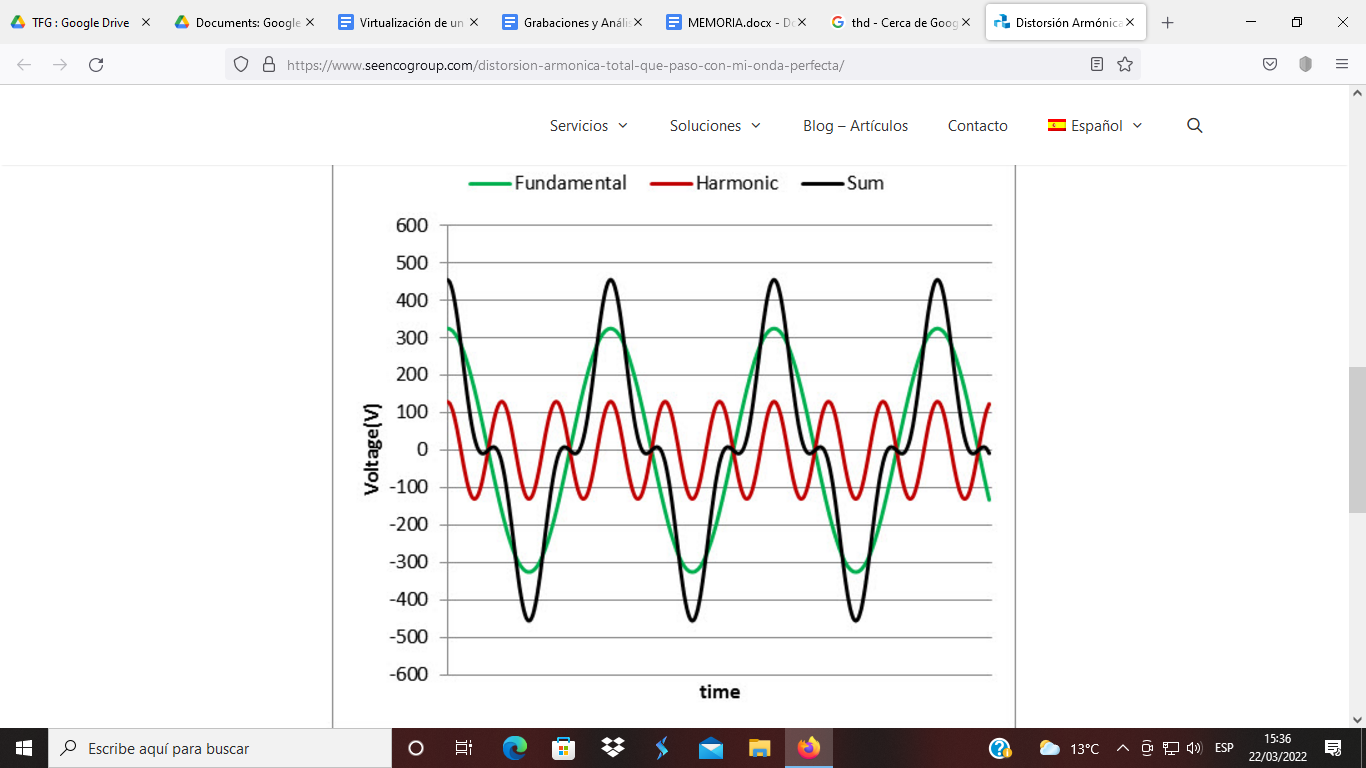


Figura 18: THD

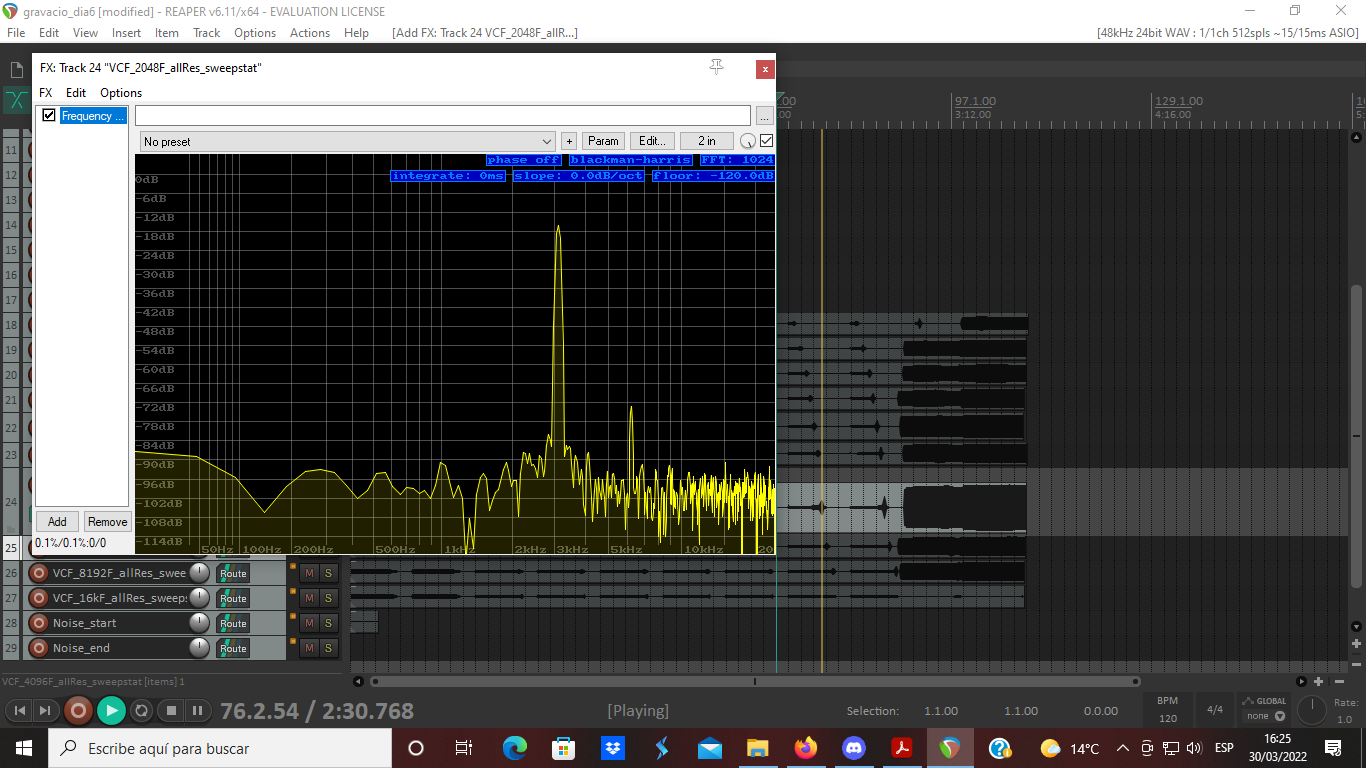
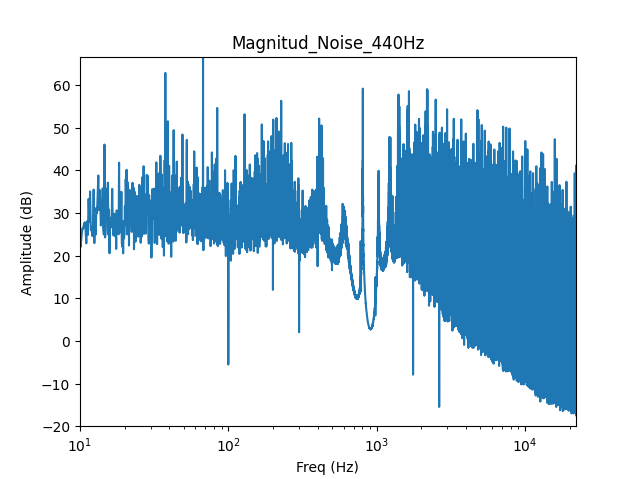
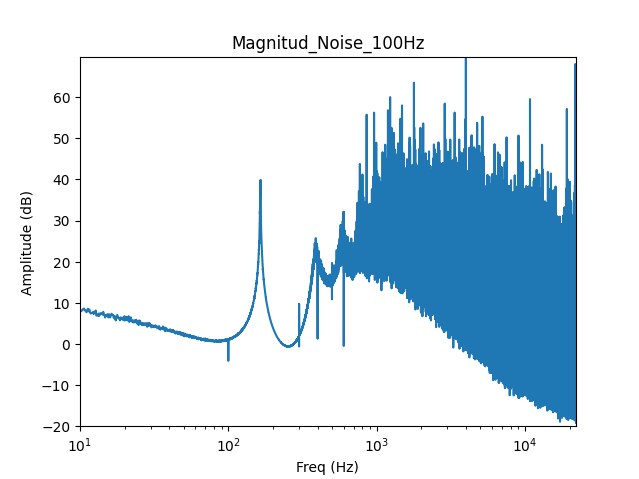


Figura 19: VCF\_4096F\_allRes\_sweepstat



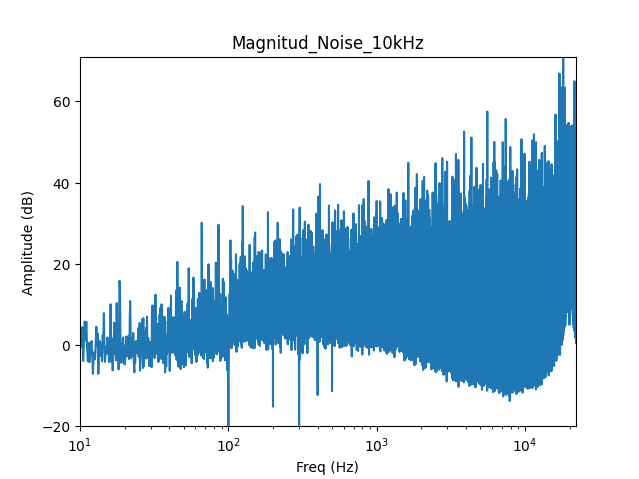
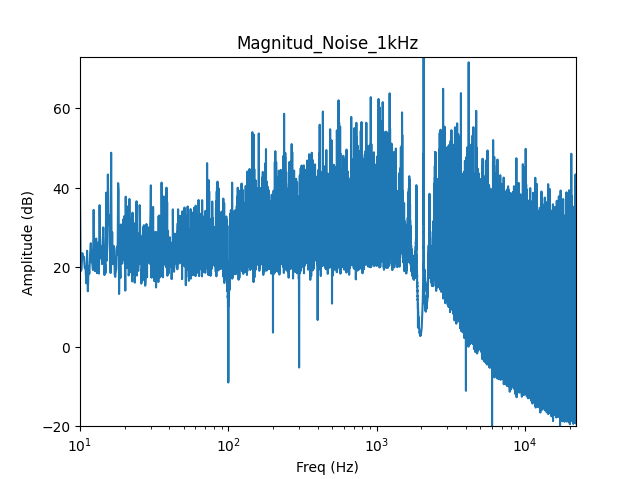
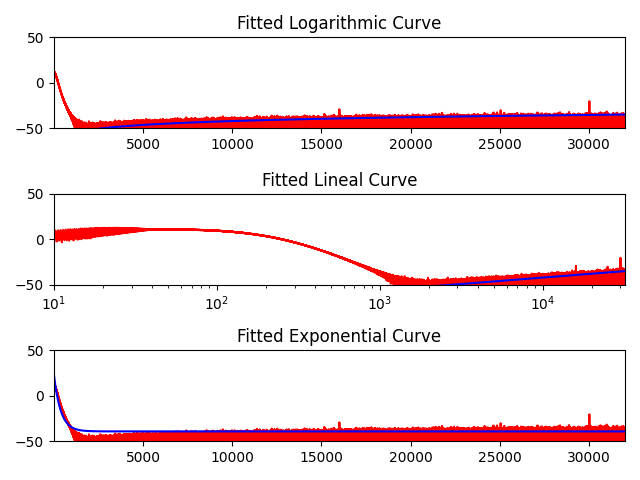
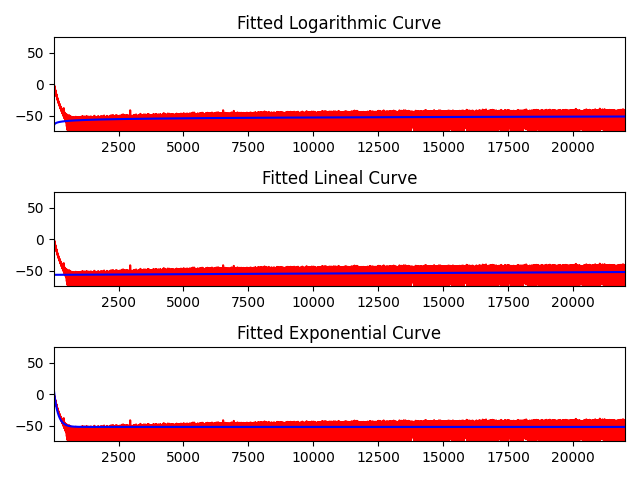
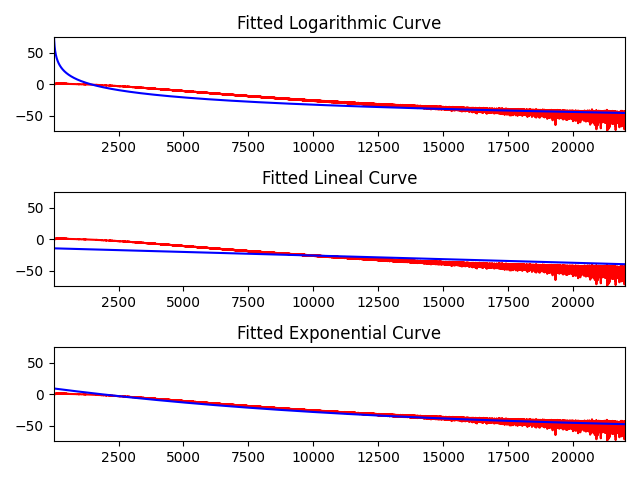
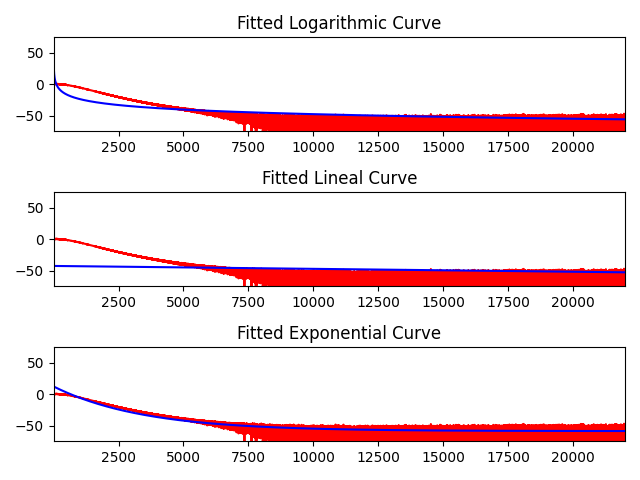


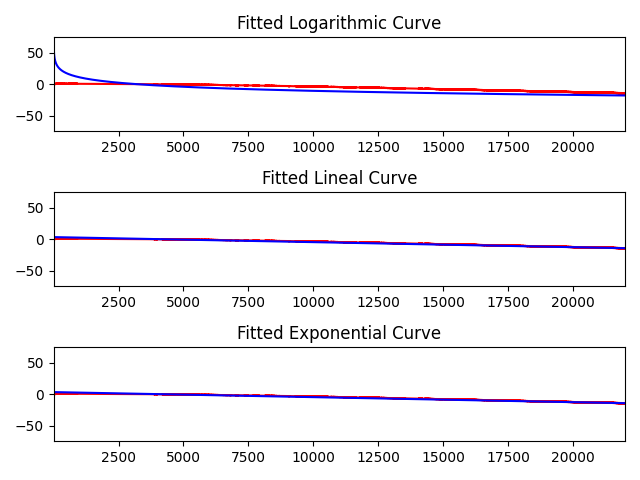
Figura 20: FFT\_Noise\_Tones

****

*16F\_0R 64F\_0R*

****

*256F\_0R 1024F\_0R*

****

*4096F\_0R*

Figura\_21:Filter\_Response

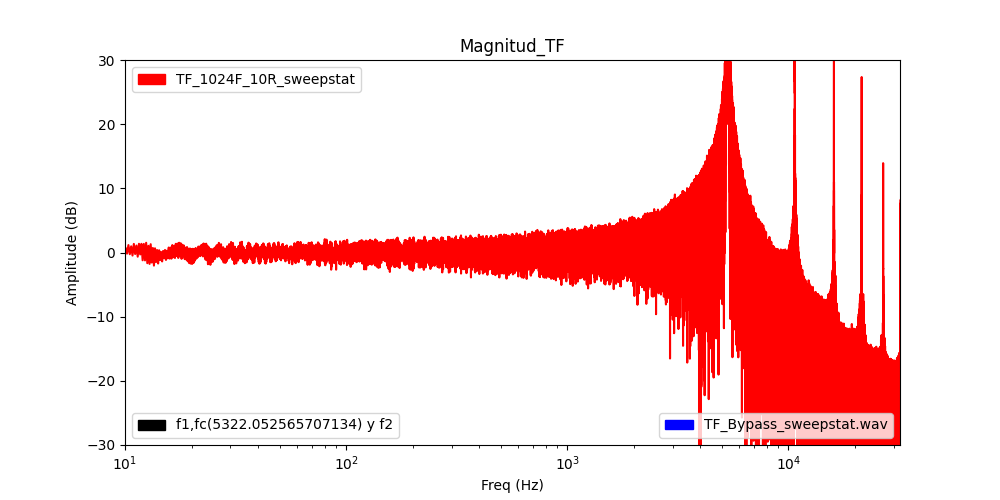
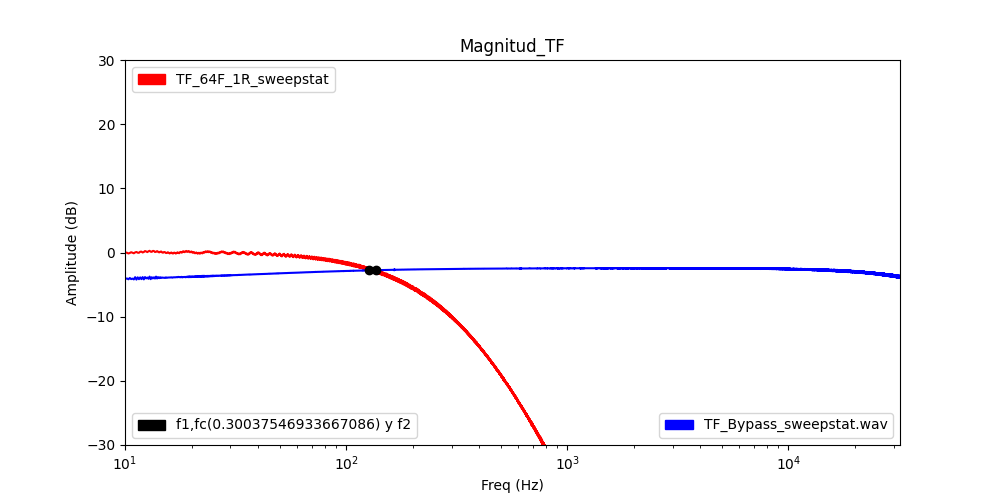
****

Figura 22: TF\_1024F\_10R\_sweepstat



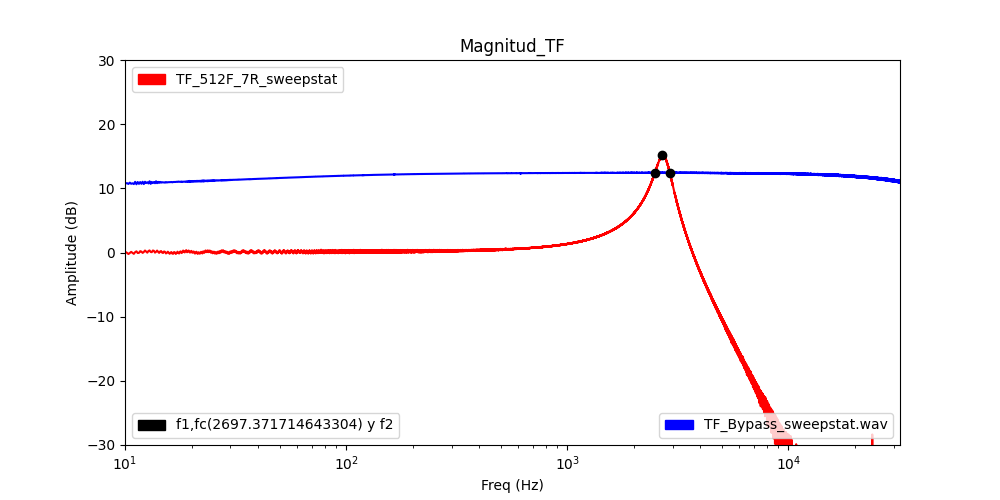
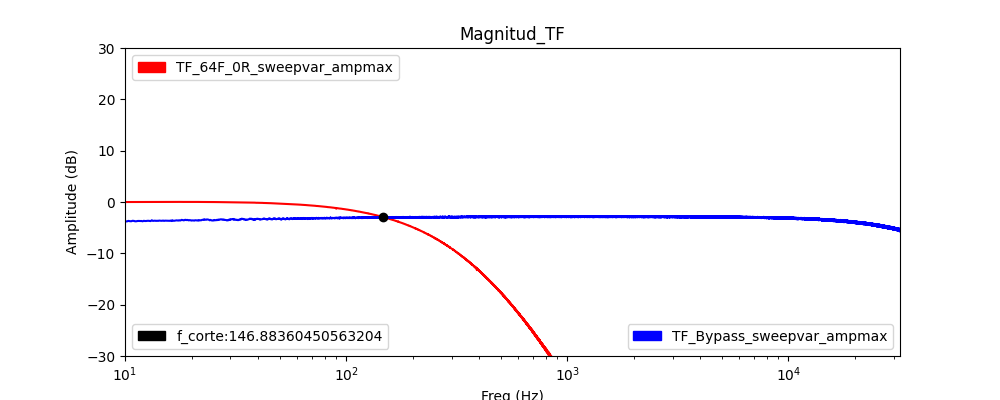
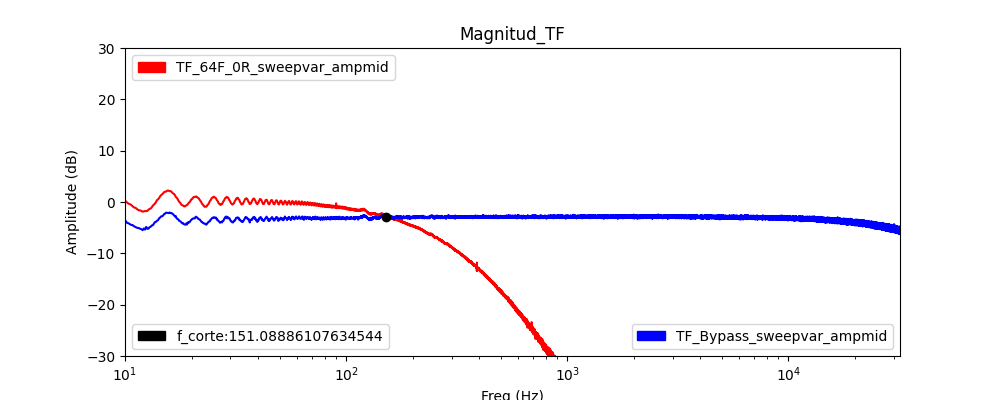


Figura 23: Rango\_Resonancia\_sweepstat





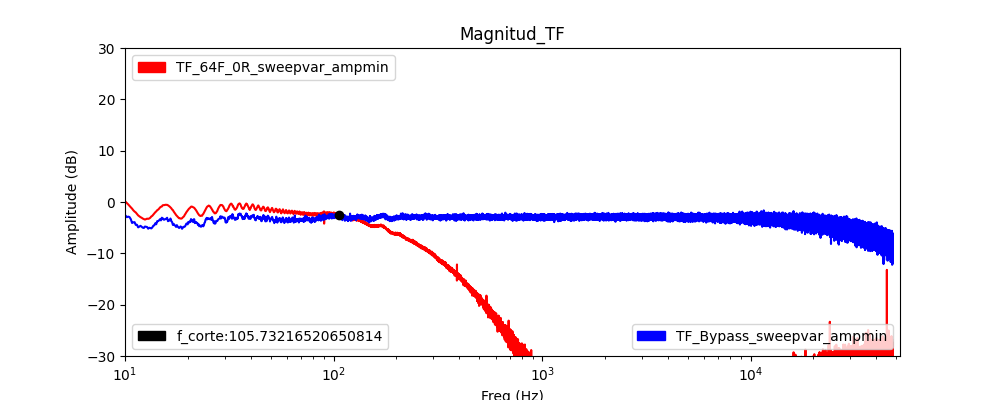
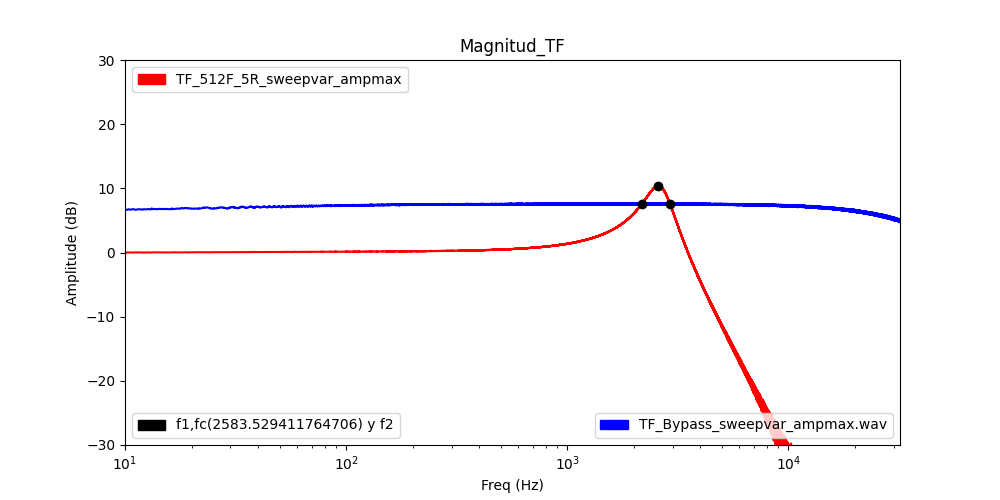
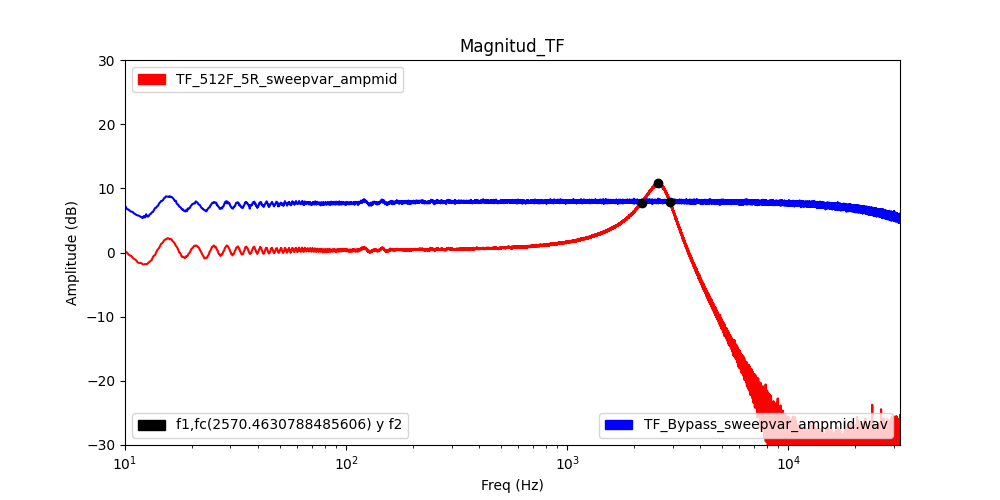


Figura 24: TF\_64F\_0R\_sweepvar





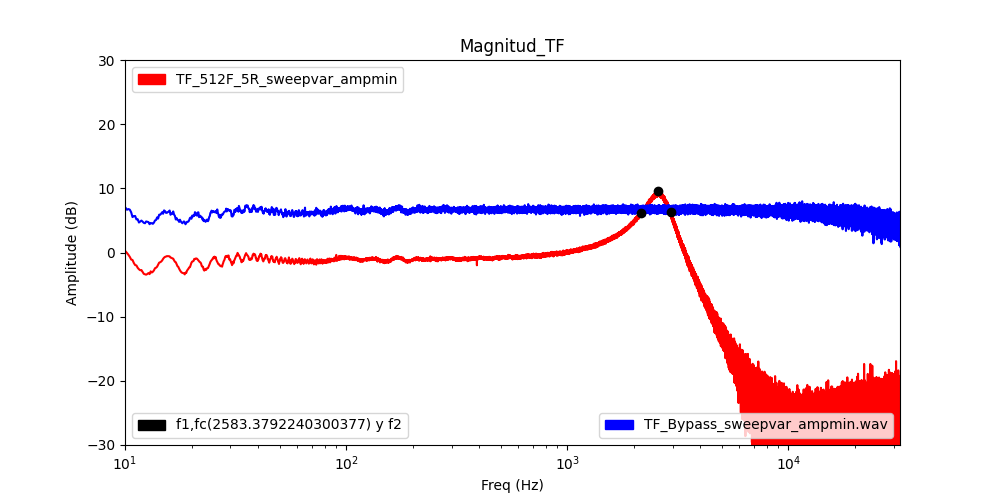


Figura 25: TF\_512F\_5R\_sweepvar

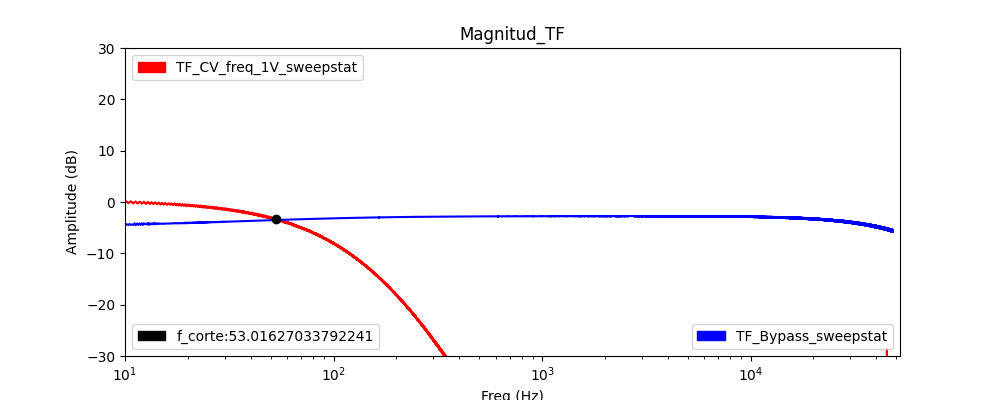


Figura 26: TF\_CV\_freq\_1V\_sweepstat

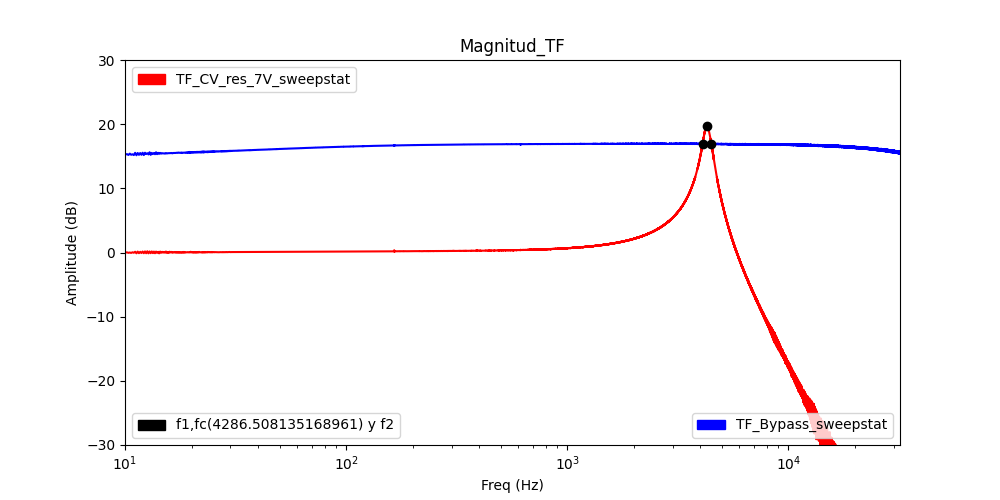


Figura 27: TF\_CV\_res\_7V\_sweepstat

**TABLAS**

| **Resultados** | **Frecuencia de corte (Hz)** | **Pendiente**  **(dB/octava)** |
| --- | --- | --- |
| *16F\_0R* | 30.64 | 7.81 |
| *32F\_0R* | 43.40 | 7.87 |
| *64F\_0R* | 135.47 | 8.96 |
| *128F\_0R* | 404.91 | 10.32 |
| *256F\_0R* | 735.92 | 10.93 |
| *512F\_0R* | 1370.61 | 11.13 |
| *1024F\_0R* | 2431.69 | 11.36 |
| *2048F\_0R* | 4186.18 | 11.45 |
| *4096F\_0R* | 8936.47 | 11.62 |
| *8192F\_0R* | 25935.62 | 3.95 |

Tabla 1: Frecuencia\_sweepstat

| **Resultados** | **f1** | **f2** | **f. central** | **f. de resonancia** | **Pico** | **Factor Q** |
| --- | --- | --- | --- | --- | --- | --- |
| *16F:*  *1R*  *2R*  *3R*  *4R*  *5R*  *6R*  *7R*  *8R*  *9R*  *10R* | 20.28  32.59  38.15  38.60  40.55  41.15  4.35  17.27  34.39  42.95 | 21.18  34.54  38.15  40.10  40.55  41.15  42.50  43.40  43.55  43.25 | 0.30  0.90  19.52  19.52  23.73  24.33  27.48  31.09  43.10  43.25 | 20.58  33.49  38.15  39.20  40.55  41.15  13.52  27.33  38.60  42.95 | 2.08  0.48  0.62  1.24  1.83  2.67  3.79  6.23  15.56  51.27 | 22.99  17.18  *inf*  26.19  *inf*  *inf*  0.36  1.05  4.22  143.50 |
| *32F:*  *1R*  *2R*  *3R*  *4R*  *5R*  *6R*  *7R*  *8R*  *9R*  *10R* | 41.001  72.24  78.85  13,21  27.63  37.25  47.01  59.77  81.10  87.86 | 43.25  73.89  80.20  81.85  82.30  83.65  85.31  86.66  88.01  88.01 | 0.30  39.95  52.41  56.62  62.63  64.43  68.18  73.59  88.01  88.01 | 42.05  72.99  79.45  32.74  47.61  55.72  63.23  71.94  84.41  87.86 | 0.89  1.45  3.60  4.91  6.002  7.15  8.65  10.88  25.11  62.34 | 18.69  44.22  58.83  0.48  0.87  1.20  1.65  2.68  12.23  585.50 |
| *64F:*  *1R*  *2R*  *3R*  *4R*  *5R*  *6R*  *7R*  *8R*  *9R*  *10R* | 126.76  15.77  154.85  189.69  209.51  233.54  255.02  325.91  325.91  335.52 | 135.90  290.01  304.88  311.64  314.64  319.00  321.40  326.21  326.21  335.67 | 0.30  211.46  243.30  260.12  269.14  278.75  290.01  326.06  326.21  335.67 | 131.11  67.58  217.17  243.004  256.67  272.89  286.26  325.91  325.91  335.52 | 0.30  4.06  6.56  8.09  9.36  11.09  13.25  20.59  64.15  73.11 | 14.33  0.24  1.45  1.99  2.44  3.19  4.31  1085.50  1085.50  2234.50 |
| *128F:*  *1R*  *2R*  *3R*  *4R*  *5R*  *6R*  *7R*  *8R*  *9R*  *10R* | 374.87  20.58  526.11  599.40  662.33  705.28  761.30  852.16  915.39  943.48 | 383.88  820.18  868.99  881.15  892.11  898.12  906.23  915.85  915.54  943.63 | 56.62  621.48  729.76  757.25  784.13  810.11  834.44  915.69  915.54  943.63 | 379.22  129.76  676.14  726.61  768.66  795.84  830.54  883.40  915.39  943.48 | 0.58  3.98  7.29  8.78  10.53  12.05  14.65  21.29  71.02  76.09 | 42.097  0.16  1.97  2.58  3.34  4.13  5.73  13.87  6095.50  6282.50 |
| *256F:*  *1R*  *2R*  *3R*  *4R*  *5R*  *6R*  *7R*  *8R*  *9R*  *10R* | 756.34  20.57  942.73  1073.69  1164.85  1254.52  1337.72  1599.50  1600.40  1653.27 | 758.90  1438.20  1523.05  1546.20  1561.50  1576.22  1586.43  1600.85  1600.55  1653.42 | 108.28  1109.74  1288.76  1336.07  1388.03  1423.78  1465.83  1600.25  1600.55  1653.42 | 757.58  171.96  1198.20  1288.31  1348.54  1406.06  1456.67  1600.10  1600.40  1653.27 | 0.73  3.95  7.28  8.93  10.41  12.28  14.56  52.18  72.63  76.66 | 296.73  0.12  2.06  2.73  3.40  4.37  5.86  1183.83  10656.50  11008.50 |
| *512F:*  *1R*  *2R*  *3R*  *4R*  *5R*  *6R*  *7R*  *8R*  *9R*  *10R* | 1225.68  8.41  1728.66  1977.52  2151.14  2315.14  2491.46  2944.13  2944.13  3025.23 | 1230.19  2659.82  2784.18  2831.94  2860.48  2887.36  2911.09  2945.18  2944.73  3025.53 | 116.10  2039.40  2354.04  2454.97  2543.58  2623.48  2697.37  2944.73  2944.73  3025.38 | 1227.78  149.44  2193.79  2366.36  2480.50  2585.33  2693.02  2944.58  2944.28  3025.23 | 0.84  4.25  7.18  8.89  10.46  12.38  15.16  58.73  72.42  77.74 | 272.53  0.06  2.08  2.77  3.50  4.52  6.42  2800.93  4901.25  10072.00 |
| *1024F:*  *1R*  *2R*  *3R*  *4R*  *5R*  *6R*  *7R*  *8R*  *9R*  *10R* | 2087.31  5.11  3040.70  3478.35  3817.17  4101.33  4369.26  5184.33  5184.33  5321.00 | 2104.88  4686.31  4909.34  4986.23  5042.55  5086.56  5120.05  5185.23  5185.38  5322.20 | 177.97  3557.80  4147.58  4295.97  4456.67  4601.15  4761.70  5184.93  5185.38  5322.05 | 2096.02  154.69  3863.58  4164.55  4387.28  4567.36  4729.71  5184.63  5185.78  5321.45 | 0.92  4.28  7.13  8.84  10.62  12.55  14.97  62.05  73.11  77.73 | 119.28  0.03  2.07  2.76  3.58  4.64  6.30  5753.67  4931.79  4429.12 |
| *2048F:*  *1R*  *2R*  *3R*  *4R*  *5R*  *6R*  *7R*  *8R*  *9R*  *10R* | 3580.77  11.86  5276.09  6012.31  6537.97  7001.90  7392.54  8899.82  8901.63  9131.26 | 3628.84  8035.34  8440.40  8581.13  8677.40  8745.28  8801.45  8900.73  8902.08  9132.77 | 209.66  6129.01  7128.51  7457.42  7670.09  7898.52  8091.06  8900.42  8901.93  9132.77 | 3604.66  308.64  6673.14  7182.73  7532.06  7825.08  8066.28  8900.12  8901.78  9131.86 | 0.93  4.13  7.23  8.98  10.60  12.39  14.36  64.73  74.95  78.43 | 75.00  0.04  2.11  2.80  3.52  4.49  5.72  9876.83  19757.17  6080.40 |
| *4096F:*  *1R*  *2R*  *3R*  *4R*  *5R*  *6R*  *7R*  *8R*  *9R*  *10R* | 10912.64  10.96  10747.88  12552.99  13557.30  14624.98  15619.52  18793.74  18793.59  19298.82 | 11124.40  17291.26  18092.21  18363.30  18489.31  18595.94  18722.10  18794.64  18793.89  19303.48 | 661.58  13034.79  14875.19  15788.94  16274.04  16686.46  17235.24  18794.49  18793.74  19303.18 | 11017.92  435.39  13944.63  15182.63  15832.34  16491.36  17100.53  18794.04  18793.59  19301.08 | 0.76  4.06  6.94  8.87  10.29  12.19  14.54  72.57  76.73  76.98 | 52.03  0.02  1.90  2.61  3.21  4.15  5.51  20856.33  62567.50  4145.60 |
| *8192F:*  *1R*  *2R*  *3R*  *4R*  *5R*  *6R*  *7R*  *8R*  *9R*  *10R* | 31675.49  8.11  26080.40  30218.22  33142.23  36559.30  39621.48  43612.11  45151.09  44661.78 | 34342.53  16988.79  27896.92  31406.66  34939.07  37453.22  40684.35  44044.66  47994.44  47997.60 | 1350.94  34771.76  39772.41  41878.05  43264.58  45001.20  45705.28  47578.72  47912.89  47984.08 | 32981.98  371.11  26973.27  30806.66  34028.79  37003.55  40149.39  43827.78  46550.99  46299.57 | 0.80  4.24  7.20  8.77  10.47  12.43  14.80  19.20  23.07  21.88 | 12.34  0.02  14.85  25.92  18.94  41.39  37.77  101.32  16.37  13.88 |
| *16384F:*  *1R*  *2R*  *3R*  *4R*  *5R*  *6R*  *7R*  *8R*  *9R*  *10R* | 41981.23  20.58  0.00  0.00  0.00  0.00  0.00  0.60  0.00  1.35 | 47999.10  20575.72  25848.06  26493.42  27379.67  26720.65  25079.85  24706.33  38189.14  33436.60 | 1913.69  47142.28  47482.30  47428.84  47958.55  47957.20  47840.95  47546.43  47965.91  47996.99 | 44889.31  650.61  0.00  0.00  0.00  0.00  0.00  121.80  0.00  212.51 | 1.04  3.90  4.57  4.67  4.74  4.67  4.42  4.25  4.89  4.37 | 7.46  0.03  0.00  0.00  0.00  0.00  0.00  0.01  0.00  0.01 |

Tabla 2: Resonancia\_sweepstat

| **Resultados** | **Frecuencia de corte (Hz)** | **Pendiente**  **(dB/octava)** |
| --- | --- | --- |
| *16F\_0R:*  *Ampmax*  *Ampmid*  *Ampmin* | 29.45  29.59  18.17 | 7.75  7.71  3.79 |
| *64F\_0R*  *Ampmax*  *Ampmid*  *Ampmin* | 146.88  151.09  105.73 | 9.08  9.23  6.57 |
| *512F\_0R*  *Ampmax*  *Ampmid*  *Ampmin* | 1407.11  1441.05  1008.91 | 11.20  11.73  7.73 |
| *4096F\_0R*  *Ampmax*  *Ampmid*  *Ampmin* | 10180.32  9996.04  7128.81 | 11.56  11.73  8.56 |

Tabla 3: Frecuencia\_sweepvar

| **Resultados** | **f1** | **f2** | **f. central** | **f. de resonancia** | **Pico** | **Factor Q** |
| --- | --- | --- | --- | --- | --- | --- |
| 16F:  2R  Ampmax  Ampmid  Ampmin  5R  Ampmax  Ampmid  Ampmin    8R  Ampmax  Ampmid  Ampmin  10R  Ampmax  Ampmid  Ampmin | 32.14  25.68  0.30  39.05  2.10  3.90  17.27  19.37  19.82  41.90  42.05  41.90 | 32.44  25.68  10.81  39.05  35.29  32.59  41.90  40.85  39.35  42.20  42.20  42.20 | 4.66  15.62  7.21  22.53  15.62  7.36  31.54  29.29  32.44  42.20  42.20  42.05 | 32.14  25.68  1.80  39.05  8.56  11.26  26.88  28.08  27.78  41.90  42.05  41.90 | 0.15  1.80  2.65  1.38  3.07  2.24  5.72  6.54  5.48  51.89  67.47  55.23 | 107.50  *inf*  0.17  *inf*  *0.26*  *0.39*  1.09  1.31  1.43  140.00  280.50  140.00 |
| 64F:  2R  Ampmax  Ampmid  Ampmin  5R  Ampmax  Ampmid  Ampmin  8R  Ampmax  Ampmid  Ampmin  10R  Ampmax  Ampmid  Ampmin | 312.39  0.30  1.20  227.83  230.99  228.13  306.23  308.03  307.28  361.95  361.95  361.95 | 312.99  312.54  314.49  340.93  340.48  341.68  350.69  351.29  352.64  362.10  362.10  362.10 | 225.73  220.48  244.05  290.91  291.36  293.77  328.01  330.26  328.76  362.10  362.10  362.10 | 312.54  9.62  19.37  278.60  280.40  279.05  327.56  328.91  329.06  361.95  361.95  361.95 | 3.52  3.91  2.53  9.24  9.71  8.29  17.58  18.12  16.66  69.70  76.65  74.34 | 520.50  0.03  0.06  2.46  2.56  2.46  7.37  7.60  7.26  2410.50  2410.50  2410.50 |
| 512F:  2R  Ampmax  Ampmid  Ampmin  5R  Ampmax  Ampmid  Ampmin  8R  Ampmax  Ampmid  Ampmin  10R  Ampmax  Ampmid  Ampmin | 0.30  0.00  3.30  2169.91  2173.52  2167.21  2744.98  2759.01  2773.22  41865.43  1821.33  1592.59 | 2699.92  2706.23  2731.61  2920.70  2926.71  2936.92  2987.23  2990.99  2994.44  42763.10  1821.48  1592.74 | 2032.79  2001.10  2048.71  2583.53  2570.46  2583.38  2850.86  2879.40  2911.09  41865.58  1821.48  1592.74 | 28.38  0.00  94.92  2517.45  2522.10  2522.85  2863.48  2877.75  2881.65  42311.79  1821.33  1592.59 | 3.83  4.19  2.89  10.42  10.83  9.62  20.84  21.78  20.58  92.41  86.85  97.45 | 0.01  0.00  0.03  3.35  3.35  3.28  11.82  12.96  13.03  47.13  12127.50  10604.50 |
| 4096F:  2R  Ampmax  Ampmid  Ampmin    5R  Ampmax  Ampmid  Ampmin  8R  Ampmax  Ampmid  Ampmin  10R  Ampmax  Ampmid  Ampmin | 7802.85  7835.89  7181.83  14110.74  13993.14  13790.09  18016.67  18154.39  17949.54  20745.73  20739.72  20735.52 | 19506.83  19597.70  21625.23  18489.31  20309.29  20644.81  18794.64  20318.90  20513.99  19303.48  20743.18  20738.67 | 14740.17  14423.28  15148.84  16274.04  17243.65  17625.38  18794.49  19204.35  19203.75  19303.18  20740.17  20737.92 | 12337.17  12392.14  12462.28  15832.34  16857.82  16872.84  18794.04  19206.16  19188.89  19301.08  20741.38  20737.02 | 4.98  5.26  5.55  10.37  10.87  11.16  19.43  20.56  19.92  83.33  91.87  99.92 | 1.05  1.05  0.86  2.74  2.67  2.46  8.13  8.87  7.48  6006.24  6004.50  6574.98 |

Tabla 4: Resonancia\_sweepvar

| **Resultados** | **Frecuencia de corte (Hz)** | **Pendiente**  **(dB/octava)** |
| --- | --- | --- |
| *freq\_1V (32 Hz)* | 53.02 | 7.97 |
| *freq\_4V (256 Hz)* | 426.83 | 10.38 |
| *freq\_7V (2048 Hz)* | 2795.14 | 11.29 |
| *freq\_10V (16 kHz)* | 31801.80 | 0.19 |

Tabla 5: Frecuencia\_CV\_sweepstat

| **Resultados** | **f1** | **f2** | **f. central** | **f. de resonancia** | **Pico** | **Factor Q** |
| --- | --- | --- | --- | --- | --- | --- |
| *res\_1V (1R)* | 1880.95 | 1887.71 | 165.06 | 1884.25 | 1.45 | 278.81 |
| *res\_4V (4R)* | 151.84 | 3963.15 | 2931.96 | 775.72 | 3.47 | 0.20 |
| *res\_7V (7R)* | 4093.51 | 4491.66 | 4286.51 | 4287.86 | 19.68 | 10.77 |
| *res\_10V (10R)* | 4775.67 | 4775.97 | 4775.82 | 4775.67 | 76.89 | 15899.50 |

Tabla 6: Resonancia\_CV\_sweepstat

| **Resultados** | **Frecuencia de corte (Hz)** | **Pendiente**  **(dB/octava)** |
| --- | --- | --- |
| *freq\_1V (32 Hz)*  *Ampmax*  *Ampmid*  *Ampmin* | 53.77  55.57  36.19 | 7.75  7.85  5.39 |
| *freq\_4V (256 Hz)*  *Ampmax*  *Ampmid*  *Ampmin* | 428.03  439.90  316.60 | 10.55  10.69  7.62 |
| *freq\_7V (2048 Hz)*  *Ampmax*  *Ampmid*  *Ampmin* | 2524.81  2881.35  2037.00 | 10.04  11.78  8.20 |
| *freq\_10V (16 kHz)*  *Ampmax*  *Ampmid*  *Ampmin* | 15943.93  16728.21  13.685.26 | 5.63  7.86  5.74 |

Tabla 7: Frecuencia\_CV\_sweepvar

| **Resultados** | **f1** | **f2** | **f. central** | **f. de resonancia** | **Pico** | **Factor Q** |
| --- | --- | --- | --- | --- | --- | --- |
| *res\_1V (1R)*  *Ampmax*  *Ampmid*  *Ampmin* | 1171.76  774.97  0.30 | 1197.00  903.23  468.59 | 173.62  15.62  7.21 | 1184.23  836.55  11.71 | 0.65  2.32  2.44 | 46.94  6.53  0.02 |
| *res\_4V (4R)*  *Ampmax*  *Ampmid*  *Ampmin* | 59.92  0.45  0.30 | 2494.32  2498.07  2527.21 | 1847.01  1741.88  7.21 | 386.58  33.49  27.48 | 3.31  3.70  2.47 | 0.16  0.01  0.01 |
| *res\_7V (7R)*  *Ampmax*  *Ampmid*  *Ampmin* | 2537.72  2555.74  2564.01 | 2796.65  2802.20  2800.40 | 2664.33  2682.65  2643.45 | 2664.03  2676.04  2679.50 | 19.56  20.22  19.20 | 10.29  10.86  11.33 |
| *res\_10V (10R)*  *Ampmax*  *Ampmid*  *Ampmin* | 2438.45  2438.30  2212.26 | 2438.75  2438.45  2212.42 | 2438.60  2438.45  2212.42 | 2438.45  2438.30  2212.26 | 75.95  89.07  98.32 | 8118.50  16235.50  14730.50 |

Tabla 8: Resonancia\_CV\_sweepvar

**Resumen Extendido**

En este trabajo me propongo analizar el funcionamiento del filtro pasa bajos controlado por tensión de una unidad del sintetizador analógico RSF Kobol Expander. El RSF Kobol es un sintetizador monofónico (sólo puede producir una nota a la vez) francés lanzado en 1978 por los hermanos Ruben y Serge Fernandez. El RSF Kobol Expander fue una versión de este modelo original lanzada tres años más tarde y era, básicamente, una versión en rack del RSF Kobol.

Un sintetizador es un instrumento musical de tipo electrónico que, a través de circuitos, genera señales eléctricas que son convertidas en sonidos. Una propiedad que diferencia al sintetizador de otros instrumentos electrónicos es que sus sonidos pueden ser creados y modificados. Además, pueden imitar otros instrumentos o generar nuevos. En el caso del sintetizador analógico, este usa circuitos analógicos para generar estos sonidos. Este instrumento tiene la característica de ser imperfecto sónicamente, ya que padece de desviaciones, ruidos y otros aspectos técnicos que normalmente no se consideran en el audio digital, y eso es lo que nos interesa analizar, ya que las especificaciones técnicas no siempre son las que, realmente, tiene el módulo. Uno de los elementos más característicos y diferenciadores de un sintetizador es su filtro, a menudo de tipo pasa-bajos. Por ello, este trabajo se centra en él.

Un filtro pasa-bajos es, esencialmente, un proceso que se aplica a una señal para mantener los sonidos graves y eliminar los agudos. El hecho de que se pueda controlar por tensión significa que se pueden regular sus parámetros mediante un aparato externo, por ello nos referimos a él como VCF (Voltage Controlled Filter). La idea general es que los niveles de tensión analógicos producidos por otro instrumento o por una fuente externa de voltaje se pueden utilizar para controlar las funciones y los parámetros de un módulo, en este caso, del filtro.

Para lograr este objetivo, hemos dividido el proyecto en una serie de pasos, todos ellos imprescindibles para llegar al último. Primero revisamos la documentación relacionada con la unidad para conocer las especificaciones técnicas de cada control, su historia y su funcionamiento.

A continuación, creamos una base de datos que podemos analizar para conocer el comportamiento real del filtro. Esta base consiste en una serie de grabaciones, que se han obtenido enviando señales de prueba, previamente creados y preparados, desde un ordenador hasta el filtro. Estas señales son sweep tones (señales que contienen un barrido de frecuencias, es decir, que recorren todo el espectro frecuencial) de amplitud fija y de amplitud variable y diversos tonos puros (señales con una sola frecuencia) y se graba la salida del filtro habiendo manipulado los potenciómetros de éste con diversas combinaciones, tanto manualmente como con una fuente de voltaje externa.

Aparte de las grabaciones manipulando los potenciómetros, hace falta una grabación con el filtro abierto, es decir, con el potenciómetro de la frecuencia de corte al máximo y el de la resonancia al mínimo, para tener una referencia de la respuesta sin que el filtro actúe (también denominada bypass).

Nos sirve para comparar con los niveles de las tomas procesadas y obtener geométricamente los valores de los parámetros. También hace falta una grabación del paso de la señal por la tarjeta de sonido utilizada para conectar el ordenador con el sintetizador, sin llegar al filtro, para tener la señal que le llegará al VCF. Éste contiene tanto la señal de control como el ruido y coloración aportados por la tarjeta. Esta salida (llamada loopback) también contiene el retardo introducido por la conversión DA/AD (Digital-Analógico/Analógico-Digital) que se produce en la tarjeta.

Estas grabaciones sirven para conocer los valores exactos de los parámetros, así como el comportamiento del filtro con diferentes señales de entrada. Estos parámetros hacen referencia a los 2 potenciómetros con los que hemos trabajado, uno que determina la frecuencia de corte del filtro (a partir de que frecuencia se empieza a eliminar sonido) y otro que determina el tamaño de la resonancia (la cual consiste, básicamente, en un aumento del nivel de la señal de salida, en forma de pequeña campana de Gauss, justo antes de empezar a disminuir).

En la tercera y última parte del proyecto, analizamos detalladamente estas salidas, usando diversos códigos escritos en Python. Para este análisis, nos basamos en ciertas preguntas sobre su comportamiento con las diferentes señales y sus respectivas salidas. Por ejemplo, hay preguntas como:

¿Hay diferencias en la respuesta en frecuencia del filtro para diferentes niveles de señal de audio de entrada? ¿Cuál es el rango de la frecuencia de corte? ¿Cuál es el rango de la resonancia? ¿Cuál es la pendiente real? ¿Responden igual los parámetros controlados por una fuente de voltaje externa?

Como hemos mencionado, el VCF tiene dos parámetros principales, la frecuencia de corte del filtro y la resonancia. No obstante, hay diversos parámetros secundarios, pero no por eso menos importantes. Estos son la pendiente del filtro (cuando se llega a la frecuencia de corte, el sonido no es repentinamente eliminado, sino que es un proceso gradual), el ancho de banda de la resonancia (que frecuencias ocupa la campana), la ganancia (la diferencia entre el máximo nivel de esta resonancia y el nivel previo del sonido), la frecuencia central o de resonancia (en que frecuencia se encuentra el nivel máximo de la resonancia) y, finalmente, el factor Q (indica la forma de la señal en la resonancia, y cuanto más alto es este valor, más estrecha es la forma). Todos estos valores se pueden conocer geométricamente usando diversas librerías de Python (como matplotlib, numpy, scipy o Essentia) para hacer cálculos con los datos de las grabaciones obtenidas.

Otro dato importante es que para el análisis usamos las respuestas impulsionales de las salidas. Estas consisten en señales donde el nivel es 0 en todo el dominio, excepto en el final, cuando hay un pico. Básicamente nos dicen cómo reaccionan las señales a un impulso (un sonido repentino y fuerte). Al pasarlo a dominio frecuencial y aplicar algunas operaciones (explicadas con detalle en el trabajo) estos impulsos nos dan la respuesta frecuencial de las salidas con respecto a la señal de entrada (llamada función de transferencia). Y eso es lo que podemos usar para calcular los parámetros que hemos mencionado antes.

Con este análisis, podemos comparar los valores que dan en las especificaciones técnicas con los valores encontrados empíricamente y conocer el comportamiento real del filtro, de manera que dicho comportamiento real pueda quedar caracterizado paramétricamente.

1. en inglés conocido como *Voltage Controlled Filter* (VCF) [↑](#footnote-ref-0)
2. en inglés conocido como *Voltage Controlled Filter* (VCF) [↑](#footnote-ref-1)
3. en inglés conocido como *Voltage Controlled Filter* (VCF) [↑](#footnote-ref-2)