UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Desarrollo de materiales para el aprendizaje online y la autocorrección en Procesado Digital de Sonido.

Aurora María Fernández Martínez

Tutor: José Javier López Monfort

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2017-18

Valencia,4 de septiembre de 2018

Escuela Técnica Superior de Ingeniería de Telecomunicación
UniversitatPolitècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS Gestión ambiental verificada REG.NO. ES-CV- 000030

Resumen

El objetivo de este proyecto es desarrollar materiales para el aprendizaje de Procesado Digital de Sonido de manera online a través de plataformas web.
En los últimos años los cursos MOOC (Masivo Online Open Course) se han popularizado mucho a través de plataformas como Coursera y edX. Como su propio nombre indica, estos cursos son “Masivos”, es decir son seguidos por miles de estudiantes. Es por ello que los ejercicios y problemas que se proponen al estudiante deben ser corregidos automáticamente por un ordenador para que sean viables. En el ámbito de la ingeniería de telecomunicación y en concreto en el procesado digital de sonido, muchos ejercicios se pueden resolver de diversas formas y resulta difícil encontrar una manera de realizar esta corrección automática. En este proyecto se pretende aprovechar la plataforma *Matlab Grade* de *MATLAB* para desarrollar una serie de ejercicios en el campo mencionado que puedan ser autocorregidos por la máquina. Para ello, en este trabajo se estudiarán las posibles alternativas y soluciones a diferentes casos a la vez que se prepararán los materiales de aprendizaje para el estudio de los filtros digitales en procesado de audio.

Resum

L'objectiud'aquestprojecteésdesenrotllarmaterials per a l'aprenentatge de Processat Digital de So de manera online a través de plataformes web. En elsúltimsanysels cursos MOOC (Massiu Online Open Course) s'hanpopularitzatmolt a través de plataformescom Coursera i edX. Com el seu propi nom indica estos cursos són "Massius", és a dirsónseguits per milersd'estudiants. És per això que elsexercicis i problemes que es proposen a l'estudiant han de ser corregitsautomàticament per un ordinadorperquè siguen viables. En l'àmbit de l'enginyeria de telecomunicació i en concret en el processada digital de so, moltsexercicis es poden resoldre de diverses formes i resulta difícil trobar una manera de realitzar esta correccióautomàtica. En aquestprojecte es preténaprofitar la plataforma *Matlab Grade* de MATLAB per a desenrotllar una sèried'exercicis en el campmencionat que puguen ser autocorregits per la màquina.

Per a això en aquesttreballs'estudiaran les possiblesalternatives i solucions a diferents casos al mateixtemps que es prepararan elsmaterialsd'aprenentatge per a l'estudidels filtres digitals en processatd'àudio."

Abstract

The objective of this project is development materials for online learning in Digital Sound Processing.

In recent years, the MOOC (Massive Online Open Course) courses have become very popular through platforms such as Coursera and edX. As its name suggests, these courses are "Massive", that is, they are followed by thousands of students. That is why the exercises and problems that are proposed to the student must be corrected automatically by a computer to be viable.

In the field of telecommunication engineering and in particular in digital sound processing, many exercises can be solved in different ways and it is difficult to find a way to perform this automatic correction, so in this project we intend to take advantage of the MATLAB *Matlab Grade* platform to develop a series of exercises in which they can be self-corrected by the machine.
For this purpose, in this work the possible alternatives and solutions to different cases will be studied while the learning materials for the study of digital filters in audio processing will be prepared.

Índice

Capítulo 1. Introducción y objetivos

1.1 Introducción

A medida que va pasando el tiempo, las tecnologías avanzan a pasos agigantados, por lo que, para la vida profesional es muy importante no quedarse obsoleto. Con esta idea fueron creados los cursos MOOC ("Massive Online Open Courses") en 2008 gracias a George Siemens y Stephen Downes[1], creadores del considerado primer MOOC, denominado "Connectivism and Connective Knowledge" (CCK08)"[2].

Estos cursos se definen como cursos online destinados a un número ilimitado de participantes, de acceso abierto ya que todos los materiales están disponibles en la web y su realización es gratuita, aunque en algunos casos, si se quiere un certificado, se pagan unas tasas impuestas.

Dado el gran interés social por estos cursos, han ido apareciendo plataformas para soportar estos cursos. Las más conocidas son *Coursera* y *Edx*, desarrolladas por académicos de las universidad más prestigiosas del mundo como son la Universidad de Stanford y el Instituto Tecnológico de Massachussets, respectivamente. En España está la plataforma denominada MiriadaX, impulsada por Telefónica y Universia[3] (en la que están integradas 1341 universidades de 20 países iberoamericanos)[4].

1.2 Objetivos

Los objetivos que se intentan alcanzar en este proyecto son:

- Crear un curso en plataformas web que sirva de aprendizaje, en paralelo a las clases, para el estudiante. Este aprendizaje tiene la característica de ser un aprendizaje libre, en el que cada estudiante pueda gestionar su tiempo para reforzar lo aprendido en clase.

- Una característica principal del curso que se va a desarrollar es que los problemas tienen que autocorregirse, por lo que el software para implementar estos cursos tiene que ser un software que dé la opción de poder implementar soluciones de referencia, con el fin de que el estudiante no tenga que esperar a la corrección del profesor para poder seguir avanzando en la realización del curso.

- Comprobar si este tipo de cursos basados en el aprendizaje online pueden ser utilizados en las aulas como refuerzo de los contenidos de algunas asignaturas de grado.

1.3 Software utilizado.

Para el desarrollo del curso se ha utilizado la plataforma de *MATLAB, Matlab Grade*. Las ventajas que tiene esta plataforma son la autocorrección de los problemas subidos por los usuarios, lo que provoca que cada usuario pueda marcarse unos objetivos y no dependa del instructor para que le corrija los ejercicios y, en el caso del desarrollo de este proyecto, la ayuda directa del personal de la empresa para resolver cualquier duda técnica o a nivel logístico.

En la nueva versión aparece al iniciar la página un vídeo guía en el que explican los primeros pasos a seguir en la plataforma y, el link a la documentación de *MATLAB* tanto para la vista de instructor como para la vista de estudiante.

Es una plataforma muy completa y fácil de usar y se va a proceder a explicar las funcionalidades que han sido utilizadas de esta plataforma para el desarrollo del curso:

- Se pueden crear proyectos en blanco o crear colecciones de problemas que luego pueden ser copiados a otros proyectos. Dentro de los proyectos se crean assignments, cuya función es organizar los contenidos del proyecto, estos assignments contienen los problemas y, pueden ser creados desde cero o se pueden copiar los problemas de ejemplo que facilita la plataforma o los problemas de otros cursos que el instructor haya creado anteriormente.

- Cuando ya se ha creado el proyecto, se indican las fechas en las que estará disponible para los alumnos, además de las herramientas que van a ser necesarias como *Simulink* y *Signal Proccesing Toolbox*, entre otros.

- Los cursos están divididos por assignments y, dentro de ellos están los problemas. Al igual que en el proyecto, se indican las fechas en la que estará visible para los estudiantes y el número de envíos permitidos, siendo posible hacer envíos ilimitados o poner un número exacto.

- A la hora de crear un problema hay que proporcionarle un título junto con la descripción e instrucciones necesarias para que el estudiante pueda realizar el problema. La implementación de los problemas tiene dos partes: solución de referencia y creación de los test. A su vez, la solución de referencia puede ser de dos tipos: tipo "script" o tipo "function".
  Al lado de la solución de referencia existe un cuadro de comandos denominado *Learner Template*, éste sirve como ayuda al estudiante para realizar los ejercicios. Es una característica opcional y que se puede usar o no dependiendo de la dificultad del problema en cuestión.
  En cuanto a los test se debe destacar que, si se elige como solución de referencia un script, hay cuatro formas de implementar los diferentes test: *Variable Equals Reference Solution*, compara el valor de de la variable indicada; *Function or Keyword is Present*, comprueba que las funciones indicadas están presentes en el código del estudiante; *Function or Keyword is Absent*, al contrario que la anterior, comprueba que las funciones no estén y, *MATLAB Code*, que tiene la ventaja de poder implementar las otras tres formas.
  Si se elige como solución de referencia el tipo function, solo se pueden implementar los test utilizando *MATLAB Code*.

- Una vez implementada la solución de referencia, los test y el posible *Learner Template*, se procede a la validación del problema para que el instructor pueda

comprobar que no ha cometido ningún error en la programación de la solución o en alguno de los test.

- Existe una ventana denominada *Learner Preview* en la que se puede visualizar cómo se verá el ejercicio en la vista de estudiante.

- Finalmente, se puede guardar el trabajo de dos formas: guardar como borrador, sirve para poder modificar el problema en un futuro y no es visible para los estudiantes y, guardar como final, se utiliza cuando se guarda la versión final y se quiere que sea visible para que el estudiante pueda realizarlo.

- Cuando los estudiantes realizan los problemas, el instructor puede saber cuántas personas lo han hecho resuelto, si ha conseguido hacerlo bien o si ha fallado y, cuántas no lo han intentado gracias a una barra indicadora en la parte de arriba del problema y, si quiere observar con más detalle, existe una vista en la que se comprueba qué estudiante ha realizado el problema, cuándo lo ha realizado, cuántas veces ha corregido su respuesta y el número de test que ha pasado.
  La barra indicadora usa tres colores; color verde, para cuando un estudiante ha resuelto el problema; color rojo, para cuando el estudiante lo ha intentado pero no lo ha resuelto correctamente y, color gris para cuando los estudiantes todavía no han intentado realizar el ejercicio.

Figura 1. Imagen de la barra indicadora del problema 1.

En la Figura1 se observa que sólo una persona ha realizado el problema de los 4 estudiantes y, ha resuelto el problema correctamente. En cambio, si se observa la Figura2, se observa que 3 personas lo han intentado pero sólo una de ellas lo ha resuelto correctamente.

Figura 2. Imagen de la barra indicadora del problema 3.

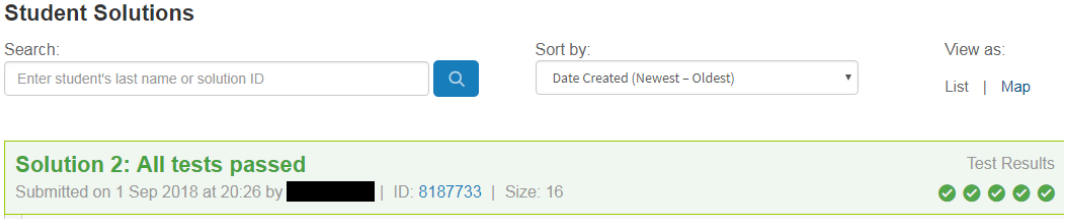

Figura 3. Imagen de la vista en detalle de las respuestas de los estudiantes.

Además de la plataforma *Matlab Grade*, se ha utilizado el programa *MATLAB* con el fin de tener una segunda copia de todos los problemas planteados y porque es más rápido a la hora de compilar que la plataforma online.

Capítulo 2. Marco teórico

2.1 Aprendizaje online

Esta modalidad de aprendizaje también conocida como e-learning, es la modalidad de aprendizaje más novedosa y, para definirla hay una amplia bibliografía de consulta, pero en resumen, e-learning es una modalidad de aprendizaje basada en el uso de las TIC´s como soporte de distribución de contenidos en el que el estudiante se autogestiona.

Dentro del aprendizaje online se encuentran los cursos MOOC[5]. Hay que destacar que un curso MOOC no es lo mismo que un curso online y las diferencias están recogidas en la siguiente tabla:

Curso online	MOOC
Tasas de matriculacón	Gratuito
Apoyo del profesor	No hay apoyo del profesor
Orientado a la adquisición del título	Orientado al proceso de aprendizaje
Tamaño de grupo limitado	Grupos ilimitados
Acceso abierto a los contenidos del curso en concreto	Acceso abierto a todos los contenidos

Tabla 1. Diferencias entre los cursos online y MOOCs.

Los MOOCs se dividen en dos tipos: Network-based MOOCs y Content-based MOOCs:

- Network-based MOOCs. Estos MOOCs no se centran totalmente en la adquisición de competencias o en los contenidos, se centra en ir construyendo el conocimiento mediante conversaciones en las redes de comunicación, dándole máxima importancia al análisis y búsqueda de recursos. Este esquema era el que siguieron los primeros cursos MOOC.

- Content-based MOOCs. En este caso, la adquisición de competencias cobra mucha más importancia, los cursos suelen ser impartidos por profesores de grandes universidades y el sistema de evaluación tiende a ser automático. El curso realizado en este trabajo es de este tipo.

A pesar de todas las ventajas que ofrecen estos cursos también hay desventajas. Estos cursos tienen una alta tasa de abandonos debido principalmente a la falta de motivación de

los estudiantes y, la dificultad para conseguir certificados ya que en la mayoría debes pegar unas tasas.

2.2 Filtros digitales de audio

Los filtros digitales de audio siempre tienen que cumplir las siguientes características[6] para que su implementación sea viable:

- Causalidad. Característica importante para los sistemas que funcionan en tiempo real.
- Estabilidad. Esta característica será importante en los filtros IIR ya que, en circunstancias concretas, pueden llegar a la inestabilidad.
- Cantidad de almacenamiento finita. La cantidad de valores a la entrada debe ser fijo para que el sistema no se colapse.
- Cantidad de operaciones por muestra de salida finita. El número de operaciones por muestra debe ser fijo

Para implementar estos filtros en MATLAB se pueden usar las técnicas de diseño que se van a ver a continuación o la herramienta f*datool*, que va a pasar a llamarse *filterDesigner*. Esta herramienta es muy intuitiva ya que permite indicar mediante opciones qué tipo de filtro se quiere implementar junto con todas sus características. En lasiguiente imagen se puede observar cómo se ha implementado un filtro FIR paso bajo de orden 4 por el método de las ventanas

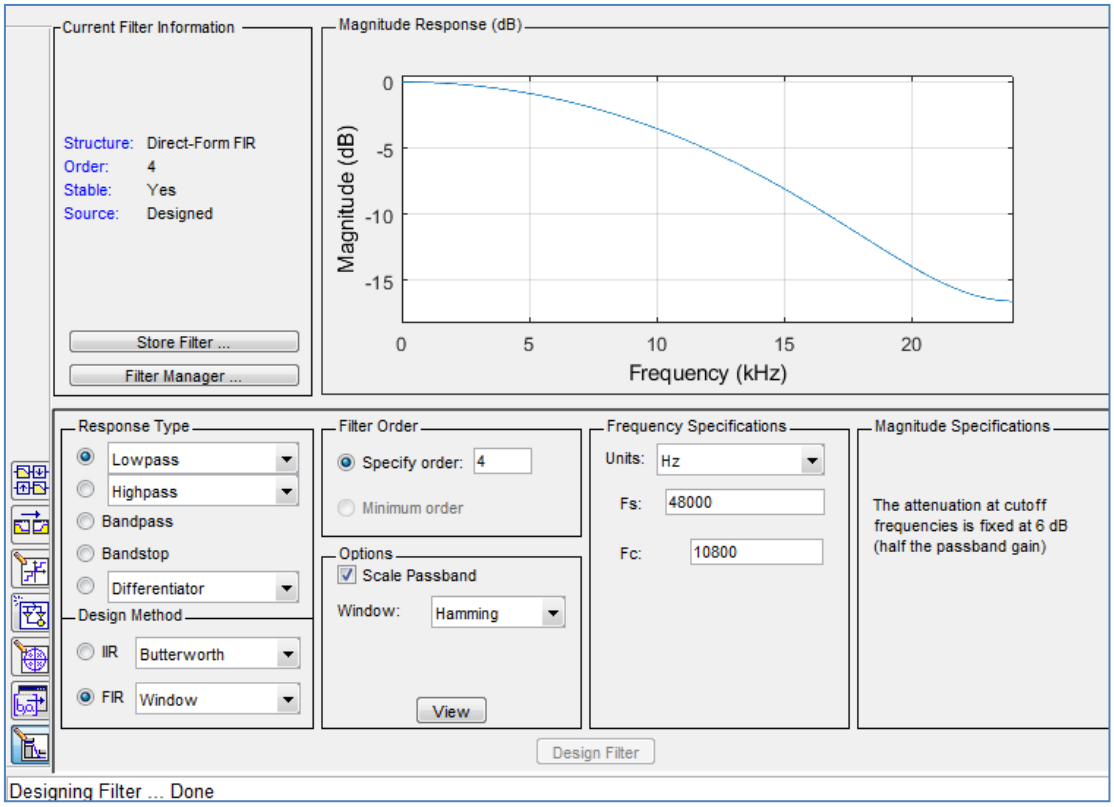

Figura 4. Ejemplo de implementación de un filtro con filterDesigner.

2.2.1 Filtros FIR

Los filtros FIR("Finite Impulse Response") son filtros que no tienen realimentación, por lo que siempre son estables y, son no recursivos, es decir, que los coeficientes a_i son todos cero.

En la siguiente imagen se observa el esquema de un filtro FIR de orden N:

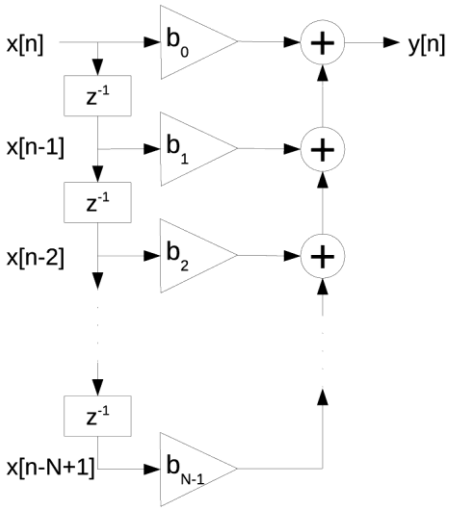

Figura 5. Esquema de un filtro FIR.

x[n] es la señal a procesar por el filtro.

Z^{-1} corresponde con el retardo de una muestra.

x[n-N] es la señal retrasada N muestras.

b_n son los diferentes coeficientes no recursivos del filtro.

y[n] señal de salida ya filtrada.

Observando el esquema se puede obtener la respuesta al impulso de este filtro como:

$$y[n] = \sum_{k=0}^{N} b_k * x[n-k] \text{[7]} \quad (1)$$

Una vez obtenida la respuesta al impulso se llega a otra de las características del filtro FIR, esta respuesta coincide con el vector de coeficientes b_i. Esta característica se puede observar en el siguiente ejemplo:

La señal de entrada es una delta, δ[n], cuya forma es:

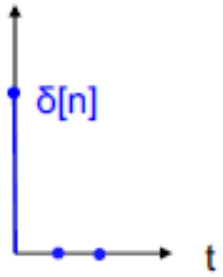

Figura 6.*δ*[n].

Cuando la señal de entrada, x[n]= *δ*[n], pasa por el filtro, se obtiene la siguiente señal de salida:

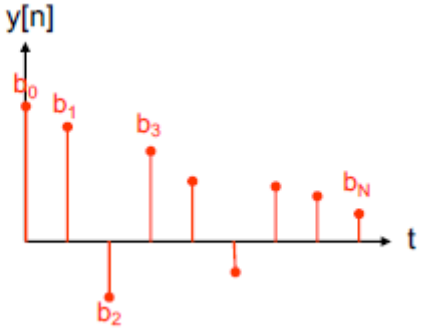

Figura 7. y[n].

Las ventajas[8] de estos filtros son: la facilidad de obtener su respuesta al impulso a través del vector de coeficientes, tal y como se ha explicado anteriormente; se puede utilizar la transformada de Fourier para analizar fácilmente su respuesta en frecuencia y, usando la respuesta en frecuencia junto con la transformada inversa, su diseño es fácil.

El problema[9] que tienen estos filtros es que necesitan un gran número de coeficientes para tener las mismas prestaciones que un filtro IIR con un orden menor o para obtener una buena resolución a bajas frecuencias, lo que se traduce en un mayor retardo a mayor número de coeficientes.

2.2.1.1 Técnicas de diseño

Para el diseño de estos filtros en MATLAB se puede elegir entre tres tipos dependiendo de las características que se quieran.

2.2.1.1.1 Método de enventanado

Para utilizar este método hay que tener en cuenta que es un método que a frecuencias bajas no es el ideal y que la resolución en frecuencia vendrá determinada por el tamaño de la ventana y por el número de muestras de la FFT("Transformada rápida de Fourier") respecto de la frecuencia de muestreo.

Para crear un filtro con este método se debe seguir una serie de pasos:

- Crear una plantilla en el dominio frecuencial con las características deseadas.
- Realizar la transformada inversa de Fourier.
- Realizar el enventanado de la secuencia resultante de hacer la transformada inversa de Fourier.

Se debe tener en cuenta que hay diferentes tipos de ventana, pero las más utlizadas son: Hanning, Hamming, Blackman y Bartlett.

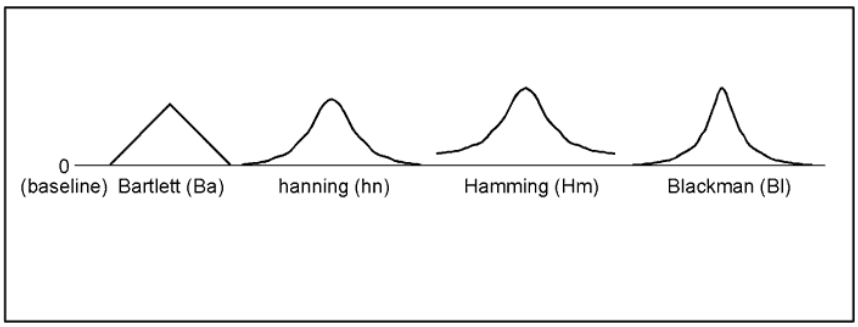

Figura 8. Esquema de las ventanas típicas.

Para implementar filtros FIR con este método en MATLAB existen dos instrucciones:

- fir1(n,f,ventana)[10]: calcula el vector de coeficientes en función del orden, la frecuencia de corte normalizada y el tipo de ventana. Puede implementar filtros paso alto, paso bajo, paso banda, elimina banda y notch.

- fir2(n,f,m)[11]: permite crear un filtro arbitrario en función del orden, un vector de frecuencias normalizadas y un vector de magnitudes.

2.2.1.1.2 Rizado constante

Este método está basado en el algoritmo Parks-McCellan[12]. Es un algoritmo iterativo cuya finalidad es crear filtros FIR óptimos y eficientes basándose en una variación del algoritmo de intercambio de Remez y de la teoría de aproximación de Chebyshev.

Los filtros creados por este método son denominados óptimos ya que minimiza el error máximo entre la respuesta en frecuencia del filtro obtenido y del descrito.

MATLAB tiene una función que implementa este método denominada FIRPM(N,F,A)[13], siendo N el orden, F las frecuencias indicadas por parejas, es decir, indicando el tramo en el que se cumple el valor del vector de ganancias A.

2.2.1.1.3 Error cuadrático mínimo

Este método se utiliza cuando se tiene una plantilla de puntos en una frecuencia y se quiere obtener una respuesta en frecuencia determinada, por lo que el error entre la respuesta del filtro obtenido y la del descrito será el error cuadrático mínimo.

MATLAB tiene una función que implementa este método denominada FIRLS(N,F,A)[14], con las mismas variables que el método de rizado constante.

2.2.2Filtros IIR

Los filtros IIR("*Infinite Impulse Response*") tienen, como su nombre indica, una respuesta al impulso infinita debido a que estos son filtros son recursivos, es decir, su respuesta en un instante depende de la respuesta en instantes anteriores.

Su ecuación en diferencias es la siguiente:

$$y[n] = -\sum_{i=1}^{N} a_k y[n-i] + \sum_{j=0}^{M} b_k x[n-j]^{[15]} \quad (2)$$

Hay cuatro tipos diferentes de esquemas para implementar estos filtros:

- Forma directa I[16]

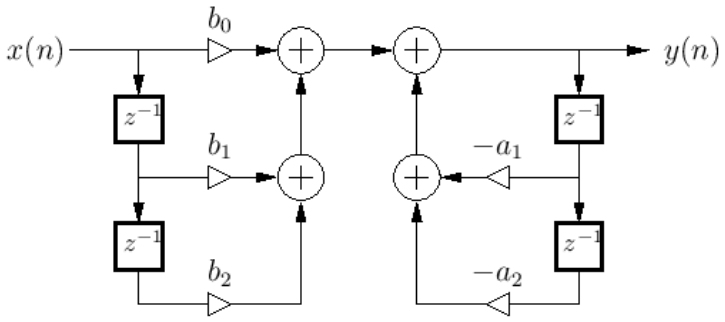

Figura 9. Forma directa I.

Este esquema tiene el doble de delays de los que son necesarios, se puede considerar como una sección todo cero seguida de una sección todo polo, no existe overflow interno y, se usa para implementar filtros de primer y segundo orden. Tiene una gran sensibilidad a errores producidos por los redondeos de los coeficientes, sobretodo en funciones de transferencia con orden alto.

- Forma directa II[17]

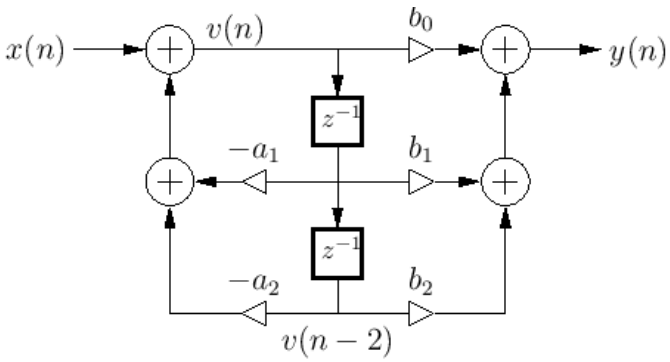

Figura 10. Forma directa II.

Se puede considerar como una sección todo polo seguida de una sección todo cero, es canónico ya que tiene la mínima cantidad de delays, puede tener overflow interno. En todas las formas directas, existe una gran sensibilidad a errores producidos por los redondeos de los coeficientes, sobretodo en funciones de transferencia con orden alto.

- Formas directas traspuestas[18]

Está basada en la fórmula de ganancia de Mason, método utilizado para encontrar la relación entre dos variables cuando se dispone del diagrama de flujo de la señal. Este método no cambia la función de transferencia.

Lo que más destaca de esta forma es que la señal de entrada está a la derecha y la señal de salida está a la izquierda.

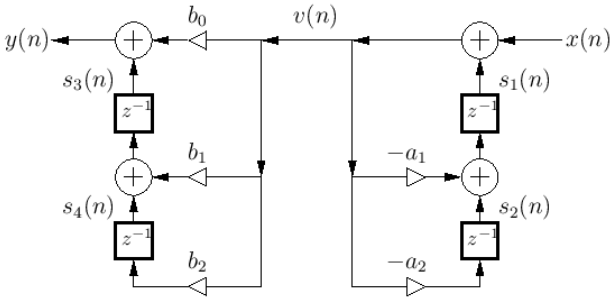

Figura 11. Forma directa I traspuesta.

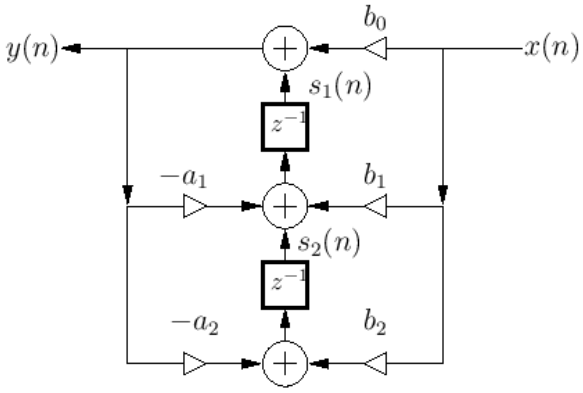

Figura 12. Forma directa II traspuesta.

La forma directa II traspuesta tiene robustez numérica frente al overflow debido a que los ceros preceden a los polos.

El estudio de la estabilidad en estos filtros es muy importante ya que, con determinadas características, se pueden tener inestabilidades. Para ello, se utiliza la transformada Z:

La definición de la transformada Z es la siguiente:

$$X[z] = \sum_{n=-\infty}^{\infty} x[n] \cdot z^{-n} \tag{3}$$

y si se aplica la definición de la transformada Z a la ecuación en diferencias de un filtro IIR (2), se obtiene la función de transferencia de un sistema LTI(Linear Time Invariant):

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 Z^{-1} + \cdots + b_M Z^{-M}}{1 + a_1 Z^{-1} + \cdots + a_N Z^{-N}} \tag{4}$$

Los polinomios de la ecuación anterior se pueden expresar en función de sus raíces:

$$H(z) = k \frac{(z-c_1)\cdot(z-c_2)\cdot\cdots\cdot(z-c_M)}{(z-p_1)\cdot(z-p_2)\cdot\cdots\cdot(z-p_N)} \tag{5}$$

La estabilidad de los filtros IIR depende de la posición en la que se encuentren los polos, siendo estable cuando todos los polos se encuentran dentro del círculo unidad e inestable cuando al menos uno de los polos está fuera del círculo unidad.

Para evitar esta inestabilidad en filtros con órdenes grandes, se utiliza la descomposición en Secciones de Segundo Orden(S.O.S). La descomposición puede seguir la estructura en serie o en paralelo, siendo la estructura en paralelo menos problemática en los redondeos. Al realizar esta descomposición, la ecuación (5), función de transferencia de un LTI en función de sus raíces quedaría de la siguiente forma:

$$H(z) = k\frac{(z-c_1)\cdot(z-c_2)}{(z-p_1)\cdot(z-p_2)}\cdot\frac{(z-c_{M-1})\cdot(z-c_M)}{(z-p_{N-1})\cdot(z-p_N)} \quad (6)$$

En *MATLAB* se puede hacer directamente con el comando *tf2sos*[19] .

Las ventajas[20] de estos filtros son la buena resolución a bajas frecuencias junto con el retardo nulo, provocando que estos filtros tengan mayores prestaciones que los FIR con un menor número de coeficientes; permite usar metodología de diseño de filtros analógicos como Butterworth y, tienen analogía con filtros analógicos basados en condensadores e inductancias.

Esos filtros tienen presencia de polos, lo que se traduce en posibles inestabilidades; el redondeo le afecta mucho más que a los filtros FIR y su implementación hardware es mucho más compleja. Otro de los inconvenientes[21] de estos filtros es que no garantizan que la fase de la función de transferencia sea lineal.

2.2.2.1 Técnicas de diseño

Para el diseño de estos filtros en *MATLAB* se puede elegir entre tres tipos dependiendo de las características que se quieran.

2.2.2.1.1 Basados en prototipos analógicos

Estos filtros pasan por una transformación del dominio ‘s’ al dominio ‘z’ provocando una alteración en algunas de sus propiedades. Hay 5 tipos de filtros:

- Filtro Butterworth:
  Este tipo de filtros no tiene rizado en ninguna de las bandas y su función en MATLAB es butter(n,f)[22], siendo n el orden del filtro y f la frecuencia de corte normalizada.

- Filtro elíptico:
  Tienen rizado en ambas bandas.
  La función en MATLAB que implementa este filtro es ellip(n,Rp,Rs,f)[23], siendo n, el orden del filtro; Rp, el rizado en la banda de paso; Rs, el rizado en la banda atenuada y, f la frecuencia de corte normalizada.

- Filtros Chebyshev:
  Los filtros Chebyshev se dividen en dos, chebyshev tipo I, con rizado en la banda de paso y Chebyshev tipo II, con rizado en la banda atenuada.
  La función para implementar chebyshev de tipo I es cheby1(n,Rp,f)[24] y, para implementar chebyshev tipo II es cheby2(n,Rs,f)[25].

- Filtro Bessel:
  Este filtro se caracteriza por tener únicamente polos provocando que su retardo de grupo sea constante y que tenga una mayor zona de transición en la banda pasante y no pasante. Su función en MATLAB es: besself(n,f)[26].

2.2.2.1.2 Aproximación de mínimos cuadrados

Para hacer un filtro IIR arbitrario hay dos métodos: mediante la combinación de filtros IIR típicos en cascada o paralelo o mediante un método que minimice el error entre la respuesta obtenida y la deseada, error cuadrático mínimo de Yulewalk.

El error cuadrático mínimo de Yulewalk[27] es un método que permite tener un filtro de respuesta arbitraria haciendo una aproximación de mínimos cuadrados en 5 fases. La función en *MATLAB* para calcular este filtro es:

$$[b, a] = yulewalk(n, f, m)^{[28]} \quad (4)$$

Siendo n el orden del filtro; f, el vector de frecuencias normalizadas y m, el vector de magnitudes a dichas frecuencias.

2.2.2.1.3 Filtros de 2º orden

Estos filtros proceden de modelos analógicos y se utilizaban en equipos analógicos como los ecualizadores. Son comúnmente conocidos como filtros paramétricos debido a que están definidos por una serie de parámetros: G(ganancia), fc(frecuencia de corte), Q(factor de calidad). Hay cuatro tipos de filtros:

- Filtros Shelving:

Existen filtros Shelving de graves, modifican las amplitudes de las bajas frecuencias (G=1 en las altas frecuencias) y filtros Shelving de agudos, modifican las amplitudes de las altas frecuencias(G=1 en las bajas frecuencias).Estos filtros tienen el factor de calidad fijo a 0.7, mientras que la ganancia puede ser ajustada por cualquier usuario y la frecuencia de corte solo en equipos profesionales.

- Filtros Peak:

Son muy utilizados en ecualizadores paramétricos y en mesas de mezclas debido a que permiten un control total de los 3 parámetros. Modifican una zona del espectro en función del valor del factor de calidad. El cálculo del factor de calidad depende de cada equipo provocando que, para calcular el ancho de banda haya dos procedimientos: ancho de banda calculado a ganancia mitad o ancho de banda calculado a ganancia -3dB

- Filtro Notch:

Este tipo de filtros se utiliza para eliminar una banda muy estrecha, como la banda de 50 Hz de la red eléctrica.

- Filtro Paso Todo:

Son filtros que solo alteran la fase por lo que el módulo de la respuesta en frecuencia es 1. Se utilizan para obtener ecos aleatorios en la reverberación o para corregir la fase de los filtros IIR.

Capítulo 3. Metodología de trabajo

3.1 Gestión del proyecto

Para tener una mejor gestión del proyecto se ha llevado a cabo una distribución de la tareas siguiendo un orden temporal, tal y como aparece en el punto 3.2 de este documento; tutorías con el profesor para que pudiese validar los avances realizados; así como anotaciones de todas las sugerencias y de los plazos de entrega de las diferentes tareas.

3.2 Distribución en tareas

3.2.1 Introducción a *Matlab Grade*

La primera tarea a realizar es conocer el funcionamiento del software utilizado para implementar el curso de aprendizaje, junto con todos sus recursos adicionales como son los catálogos de ejercicios ya implementados, en lo que se puede aprender, por ejemplo, cuál es la mejor manera de implementar los test y, la documentación de *MATLAB*. Con la nueva versión de la plataforma se ha añadido un vídeo guía de 4 minutos para que el aprendizaje de cómo crear los assignments, los test o de cómo saber manejar todas las opciones que tiene la plataforma de una manera mucho más sencilla que en la versión anterior.

3.2.2 Planteamiento de ejercicios

En paralelo con aprender a utilizar la plataforma software, se ha realizado un documento con todos los enunciados de los ejercicios del curso. Estos ejercicios se han dividido en introducción, filtros FIR y filtros IIR. En el caso de los filtros, se han dividido en las diferentes metodologías utilizadas para diseñarlos.

Filtros FIR	Filtros IIR
Método de enventanado	Basados en prototipos analógicos
Error cuadrático mínimo	Aproximación de mínimos cuadrados
Rizado constante	2º orden

Tabla 2. Metodologías para la implementación de filtros.

- **Introducción**

En este apartado se pretende que el estudiante aprenda a cargar señales y a trabajar con ellas, ya sea visualizando su respuesta, tanto en frecuencia como en el tiempo, o modificando su amplitud para comprobar cómo le afecta. Contiene 6 problemas.

- **Filtros FIR**

Tal y como indica la Tabla 1, los ejercicios de filtros FIR se dividen en 3 categorías siguiendo el esquema del marco teórico anterior. Este apartado contiene 11 problemas entre las 3 categorías. La realización de este apartado le aporta al estudiante todos los conocimientos necesarios sobre los filtros de respuesta al impulso finita además de familiarizarse a programar en *MATLAB*.

- **Filtros IIR**

Al igual que con los filtros FIR, los filtros IIR están divididos en tres categorías y, el apartado tiene un total de 14 problemas entre las tres categorías. Este es el último apartado del curso por lo que tiene un grado de complejidad mayor que los otros dos apartados.

3.2.3 Implementación de los ejercicios y sus test en *Matlab Grade*

Esta tarea consiste en programar en la plataforma web los problemas del curso así como los test de validación necesarios para asegurar que la solución del estudiante es la correcta.

Además, se ha hecho uso de la vista *Learner Template* que sirve de gran ayuda para el estudiante.

3.2.4 Mejoras

Para las mejoras se ha pedido ayuda a 4 estudiantes de la asignatura *Tratamiento digital de audio*. Estos voluntarios realizaron el curso e indicaron las cosas buenas y las cosas que se podían mejorar del curso.

3.3 Diagrama temporal

Tareas	Fecha inicio	Días trabajados	Fecha fin
Introducción a Cody Coursework	13-abr	3	16-abr
Planteamiento de ejercicios	17-abr	10	30-abr
Implementación de ejercicios y test	03-may	25	25-jun
Memoria	02-jul	42	22-ago
Manual_Cody_coursework	18-jul	1	18-jul
Manual_Matlab_Grade	06-ago	1	06-ago

Figura 13. Diagrama temporal.

Capítulo 4. Desarrollo del trabajo y resultados

4.1 Desarrollo

4.1.1 Planteamiento de ejercicios

Como ya se ha descrito anteriormente en este documento, el principal objetivo de este trabajo es crear un curso, con el fin de que el estudiante tenga la posibilidad de complementar en casa lo aprendido en la asignatura. Es por ello que ha habido un estudio previo de los boletines de problemas y de prácticas utilizados en esa asignatura, así el estudiante tiene muchos ejercicios con los que practicar y poner a prueba sus conocimientos para los futuros exámenes. Estos enunciados han sido ordenados de menor a mayor dificultad, siguiendo la misma forma de trabajar que el profesor de la asignatura.

La principal característica del planteamiento es el grado de sencillez y especificación de la redacción de los enunciados, para que el estudiante no tenga dudas de qué es lo que pide el ejercicio y pueda pasarlo sin ninguna dificultad.

El planteamiento de estos ejercicios está en el capítulo 7. Anexos.

4.1.2 Implementación de los ejercicios y test en *Matlab Grade*

En la implementación de estos ejercicios se han creado 7 assignments y dentro de ellos, sus problemas correspondientes, siguiendo el planteamiento de ejercicios creado previamente.

Los 7 assignments mencionados son los siguientes:

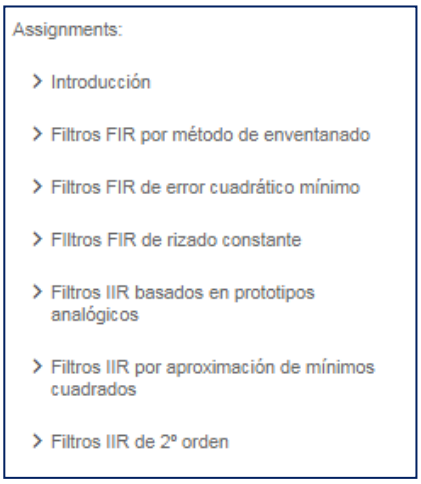

Figura 14. Imagen de la distribución de los assignments del curso.

Se observa que, como se ha explicado en el capítulo 3, el temario de los filtros FIR e IIR están divididos en 3 categorías cada uno y, dentro de cada categoría hay entre 1 y 8 problemas que el estudiante debe realizar.

Se ha creado un anexo a este documento en el que se explica con detalle cómo crear un proyecto junto con sus diferentes assignments y problemas en *MATLAB Grade* denominado "Manual_Matlab_Grade.pdf."

La vista de instructor es la que permite implementar los ejercicios y test, así como las fechas en las que estará visible cada assignment. Para demostrar el desarrollo de la implementación, se va a seguir la creación paso a paso de uno de los ejercicios, en concreto, el problema 1 del assignment "*Filtros IIR de 2º orden*".

El enunciado de este problema es: "*Diseña un filtro shelving paso bajo cuya frecuencia de corte es 100Hz, las ganancias son 2,4,6,-2,-4 y -6 y la frecuencia de muestreo es 44100Hz. Utiliza la función shelving_low.m y recuerda pasar las ganancias a lineal*."

- Lo primero ha sido añadir el archivo *shelving_low.m* que se ha creado previamente a la implementación del ejercicio y, se ha decidido utilizar un script para la solución de referencia. En la imagen siguiente se observa una de las partes del código en la que se pasan todas las ganancias a lineal y se calcula el filtro shelving con las ganancias 2 y -2:

```
fs=44100;
g1=10^(2/20);
g2=10^(-2/20);
g3=10^(4/20);
g4=10^(-4/20);
g5=10^(6/20);
g6=10^(-6/20);

[b1,a1]=shelving_low(g1,44100,100);
[h,f]=freqz(b1,a1,2000,fs);
hdB=20*log10(abs(h));
semilogx(f,hdB,'k');
hold on
[b2,a2]=shelving_low(g2,44100,100);
[h,f]=freqz(b2,a2,2000,fs);
hdB=20*log10(abs(h));
semilogx(f,hdB,'k');
hold on
```

Figura 15. Ejemplo de solución de referencia.

- Para hacerle más fácil la programación al estudiante y conseguir pasar los diferentes test de validación, se ha añadido una parte de código en *Learner Template*. En este caso, se le ha indicado cómo se pasa a lineal una ganancia y la forma de observar su respuesta en frecuencia ya que es más complicada que en otros casos y el curso está orientado a saber implementar e interpretar la respuesta, no a saber cuál es la mejor solución para observar la respuesta en frecuencia.

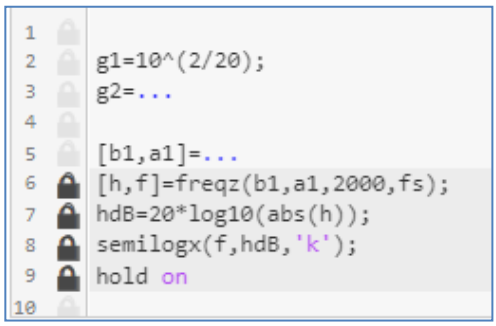

Figura 16. Ejemplo de Learner Template.

Al observar la Figura15, se advierte que en algunas líneas de código aparece el icono de unos candados, lo que indica que la línea de código no puede ser modificada por el estudiante. El instructor es libre de utilizar los candados o no, en este problema en particular se decidió que sí se iba a usar para las líneas de código que implementan la visualización de la respuesta en frecuencia.

- Una vez están implementadas tanto la solución de referencia como el *Learner Template*, se procede a crear los diferentes test que serán los encargados de verificar si la solución del estudiante es correcta o no. Para este problema se han creado 3 test y, han sido programados siguiendo la cuarta forma, denominada *MATLAB Code*, cuyas ventajas se han descrito en el *capítulo 1*, *apartado 3*, *software utilizado*. El test1 comprueba que las ganancias han sido pasadas correctamente a lineal, el test2 comprueba los valores de los coeficientes de los 6 filtros creados y, finalmente, el test3 verifica que las funciones necesarias para poder implementar este problema, están presentes.

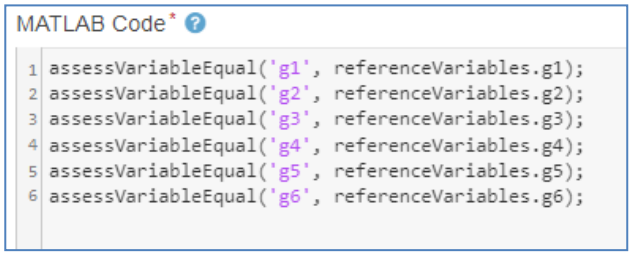

Figura 17. Test1 del ejercicio de ejemplo.

- Cuando ya está todo programado, se valida tanto la solución de referencia como los test, para comprobar que el instructor no ha cometido ningún error a la hora de programar el código del problema o de alguno de los test.

- Finalmente, se guarda el problema como versión final, en la que el estudiante puede ver el problema o, se guarda como borrador para posibles modificaciones futuras.

Este ha sido el proceso seguido para crear cada uno de los problemas que aparecen en el curso.

Otra de las funcionalidades que se han utilizado de la plataforma, ha sido el límite de envíos en los problemas, siendo configurado para que cada problema tenga un límite de dos envíos.

Una vez el estudiante ha escrito la solución, puede utilizar la opción denominada "*Run script*" que permite visualizar lo que se ha programado antes de enviar la respuesta para compararla con la solución de referencia, esto es muy útil para comprobar si el código no está bien escrito o si falta alguna función importante en él. Cuando el estudiante está seguro

de su respuesta, puede enviarla con la opción "*submit*". Cuando el estudiante hace click en *submit*, es cuando la plataforma procede a pasar cada uno de los test a la solución del estudiante y, al finalizar el proceso, al estudiante le aparece cuántos test ha pasado su código.

Si el instructor lo quiere, puede añadir un comentario que aparezca cuando el estudiante ha fallado algún test, con el fin de ayudarle a encontrar el error. En este curso se ha decidido añadir este tipo de comentarios en los problemas con un mayor gado de dificultad.

4.2 Resultados

En este apartado se va a analizar la estructura final del curso según la vista del instructor y según la vista del estudiante, ya que se ha realizado una demo con 4 de los estudiantes de la asignatura *Tratamiento Digital de Audio* del curso 2017/2018.

4.2.1 Resultados según la vista del instructor

Para evaluar esta parte de los resultados se han tenido en cuenta las indicaciones de cómo debería ser el curso según los requisitos del profesor. Como se ha comentado en capítulos anteriores, el curso se ha dividido en 7 secciones o assignmnets, en los que tienen dentro varios problemas ordenados de menor a mayor dificultad. Todos los problemas tienen su enunciado, solución de referencia y test para comprobar la solución del estudiante.

Se han podido implementar todos los requisitos de problemas del profesor salvo los ejercicios en los que se pedía comprobar la señal filtrada mediante la instrucción *sound*[29], que sirve para escuchar una señal, ya que no está implementada esta opción en la plataforma web, por lo que el curso tiene varios ejercicios menos que en las primeras indicaciones del profesor.

En cuanto a los test, todos han sido programados utilizando la cuarta forma, *MATLAB code*, ya que permite unificar varios test en uno cuando tratan de lo mismo, como por ejemplo, comprobar el valor de varias variables.

La mayoría de los problemas tienen un *Learner Template* para ayudar al estudiante, aunque hay algunos en los que se ha decidido no implementarlo, debido al nivel básico del problema

4.2.2 Resultados según la vista del estudiante

Como se ha comentado antes, para este apartado se realizó una demo con 4 voluntarios que habían cursado la asignatura en el curso anterior. La demo consistía en realizar los 6 ejercicios del primer apartado, *Introducción*, del curso denominado *Filtros digitales de audio*, que se encuentra en la plataforma web *Matlab Grade*. Los estudiantes pueden acceder a este curso mediante una invitación enviada por el instructor del curso. Para el estudio de los resultados, este apartado se ha dividido en 2: la puntuación de cada uno de los ejercicios y el feedback recibido sobre los enunciados, estructura, entre otros.

Los estudiantes tuvieron desde el 19 de julio hasta el 3 de septiembre para poder realizar los ejercicios, es un gran período de tiempo para realizar 6 ejercicios debido a que casi todos tenían que desarrollar sus propios trabajos finales de grado.

- En cuanto a la puntuación de los ejercicios se puede observar que, en la mayoría de los ejercicios, todos los estudiantes han sabido resolver los problemas sin ninguna dificultad y sin ayuda del instructor. Esto es debido a que los

enunciados son específicos y en ellos aparece toda la información necesaria para realizar cada uno de los problemas.
Otra característica importante es que no hay tiempo, por lo que los estudiantes se sienten más relajados y confiados para obtener la solución correcta.
El uso de la opción *Run script*, cuya funcionalidad está explicada anteriormente, es de gran ayuda al estudiante para previsualizar su respuesta y poder comprobar que no se ha equivocado en algún punto, antes de enviarlo para que se compare con la solución de referencia y perder un envío.
En las siguientes dos imágenes se observa mediante el uso del botón *Learner Preview*, cuya funcionalidad se ha explicado en el punto ***1.3. Software utilizado,*** la vista de estudiante de uno de los problemas.

Figura 18. Vista del estudiante I.

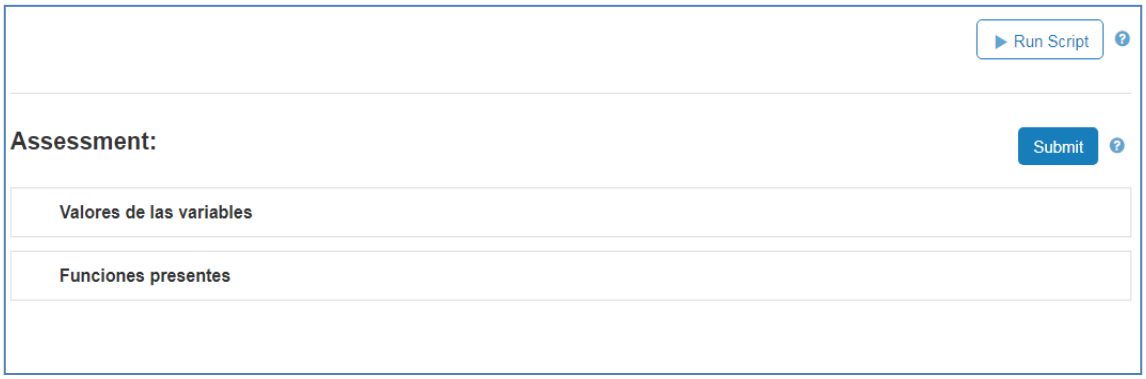

Figura 19. Vista del estudiante II.

En la parte de arriba de la Figura18 aparece el enunciado del ejercicio así como el acceso a la documentación de *MATLAB* y la opción de hacer reset. Dentro del cuadro indicado por ***Your Script*** es donde el estudiante debe implementar el código, siendo las líneas sombreadas las marcadas por el instructor con los candados para que el estudiante no pueda modificarlas.

En la Figura19 aparecen los Assessments, que son los test que ha programado el instructor y se indican para que el alumno sepa qué es lo que se va a comprobar para dar por válida o errónea la solución.

- Para obtener el feedback de los estudiantes se les ha hecho varias preguntas una vez terminada la demo. Las respuestas han sido de gran ayuda y se han tenido en cuenta para en un futuro modificar el curso, debido a que *Matlab Grade* no permite hacer ningún tipo de cambio en los ejercicios sin perder los resultados de los estudiantes.

Capítulo 5. Conclusiones y propuestas de trabajo futuras

En cuanto a las conclusiones, se pueden dividir en dos: sobre la creación de cursos MOOC en la plataforma *Matlab Grade* y la necesidad de implementar este tipo de cursos en asignaturas de la universidad.

- *Matlab Grade* es una plataforma muy completa para el desarrollo de estos cursos y fácil de usar, gracias a toda la documentación que pone a disposición del usuario, tanto si es instructor como estudiante y, además se ha creado otro manual de usuario para instructores, tal y como se ha comentado en el apartado ***4.1.2*** de este documento, como complemento a la documentación oficial de la plataforma. El problema de usar esta plataforma web, es que hay algunas asignaturas que no podrían realizar cursos MOOC basados en la autocorrección de esta plataforma ya que no usan *MATLAB*.

  En este curso en concreto no ha sido necesario pero, hay cursos en los que los instructores realizan vídeos explicativos de lo que se va a preguntar en cada uno de los módulos, lo que supone un gran esfuerzo por el instructor y eso no siempre es algo atractivo.

- Al tener todos los resultados de los voluntarios, se ha comprobado que la mayoría han tenido un desarrollo favorable de los ejercicios, ya que los resultados de cada uno de los problemas son buenos, esto se traduce en que, en la asignatura *Tratamiento Digital de Audio* los estudiantes aprenden los conocimientos propuestos en la guía docente. Por lo que incluir este tipo de cursos en esta asignatura sería una buena idea para la gente a la que le cuesta más afianzar los conocimientos.

Por tanto, se puede concluir que la creación de estos cursos es una buena idea como forma de apoyo de contenidos de algunas asignaturas específicas, porque para realizar estos cursos los contenidos y enunciados tienen que estar lo suficientemente claros para que el estudiante no tenga problemas a la hora de comprenderlos y, para asignaturas que no tengan demasiada carga de trabajo, ya que la filosofía de un MOOC es que el estudiante sea independiente y que aprenda todo lo que se enseña en el curso.

Como propuesta de trabajo futura sería integrar este curso en plataformas de aprendizaje como *Sakai* o *Moodle*, gracias a que la nueva versión de la plataforma, lanzada en julio de 2018, tiene en la visa de instructor una opción denominada *LMS Integration*, con la que poder hacer el proceso de integración. El problema de esta propuesta es que, para poder

utilizar la opción de integrar un curso en una plataforma como *Moodle*, se debe pagar una licencia aparte de la que actualmente tiene la universidad.

También sería muy interesante estudiar la posibilidad de que con esta versión e integrado el curso en una de las plataformas, se pueda usar el comando *sound* de *MATLAB* ya que es un comando muy utilizado en el procesamiento de audio porque permite escuchar las señales y, comprobar las diferencias entre una señal filtrada y sin filtrar, de una manera más dinámica y más efectiva que visualizando las respuestas en frecuencia.

Capítulo 6. Bibliografía

[1],[2][3] http://mooc.es/que-es-un-mooc

[4] http://www.universia.es

[5] http://elearningeuropa.info/en/article/MOOC-Design-Principles.-A-Pedagogical-Approach-from-the-Learner%E2%80%99s-Perspective

[6] Antonio Albiol, Valery Naranjo y Josep Prades. "*Tratamiento Digital de la Señal. Teoría y Aplicaciones*"

[7], [8],[9],[[12],[15],[20],[21],[27],[28] Apuntes de la asignatura Tratamiento Digital de Audio.

[10],[11],[13],[14],[19],[22],[23],[24],[25],[26] https://es.mathworks.com/help/index.html. Documentación online de MATLAB.

[16], [17], [18] Julius O. Smith III "*Introduction to Digital Filters: with Audio Applications*"

[27]Friedlander, B., and Boaz Porat. "The Modified Yule-Walker Methodof ARMA Spectral Estimation." IEEE® Transactions on Aerospace Electronic Systems. Vol. AES-20, Number 2, 1984, pp. 158–173.

Capítulo 7. Anexos

7.1 Planteamiento de los ejercicios

7.1.1 Introducción

1. Aprende a cargar una señal, ploteala y muestra la frecuencia de muestreo. Debes guardar el resultado de plot en la variable p.
2. Carga una señal y ploteala pero cambiando el eje x a segundos.
3. Carga una señal y haz un plot de las 60000 primeras muestras. Recuerda que la primera muestra está en la posición 1.
4. Carga una señal y haz un plot entre el segundo 3 y el segundo 10. Haz una conversión de segundos a número de muestras.
5. Carga una señal, amplifícala 1 dB y haz un plot de la señal original y la amplificada y comprueba la diferencia.
6. Amplifica la señal 50 dB y comprueba la diferencia con un plot.

7.1.2 Filtros FIR

7.1.2.1 Filtros FIR por método de enventanado

1. Diseña un filtro paso bajo de orden 25 con una frecuencia de corte de 2000Hz utilizando la función fir1 y ventana Hamming.
2. Diseña un filtro paso bajo con una fc=2000Hz y de orden 25 con fir1 y ventana Hanning.
3. Comprueba las respuestas en frecuencia de las señales de los ejercicios anteriores entre 0dB y -50dB. Utiliza dos figures para visualizar las respuestas correctamente.
   Utiliza la función freqz(b,a,1024,fs) y el comando set(gca,'xscale','log').
4. Diseña un filtro paso alto con un orden de 60 y una fc=6000Hz con fir1 y ventana Barlett. Visualiza su respuesta en frecuencia entre 20dB y -40dB
5. Diseña un filtro elimina banda con un orden de 800 , una frecuencia de corte inferior de 300Hz y una superior de 900Hz utilizando ventana Hanning. Visualiza su respuesta en frecuencia entre 10dB y -150dB.
6. Diseña un filtro paso banda de orden 400 con una frecuencia de corte inferior de 300Hz y una frecuencia de corte superior de 700Hz. Visualiza su respuesta en frecuencia entre 10dB y -180dB.

7. Diseña un filtro paso alto de orden 40 a una frecuencia de corte=4000Hz con ventana Blackman. Visualiza la respuesta entre 10 dB y -150dB

7.1.2.2 Filtros FIR por error cuadrático mínimo

1. Diseña un filtro FIR utilizando la funciófirls junto con el vector de frecuencias y de magnitudes:
   F=[0 0.15 0.2 0.3 0.4 0.5 0.6 0.64 0.65 0.7 0.8 0.9]
   A=[0.3 0.5 0.3 0.3 0.1 1 0.0 0.3 0.3 0.3 0.7 0.2]
2. Diseña un filtro de orden 20 y que cumpla con las siguientes características:

- [0-0.15] esté en 0.5dB
- [0.2-0.3] caiga a 0.1dB
- [0.4-0.5] aumente 0,25dB
- [0.6 0.64] se mantenga en 0.35dB
- [0.65-0.7] aumente hasta 1dB
- [0.8-0.9] caiga 0,5dB

7.1.2.3 Filtros FIR de rizado constante

1. Diseña un filtro FIR de rizado constante utilizando la función firpm junto cn el vector de frecuencias y el vector de magnitudes:
   F=[0 0.15 0.2 0.3 0.4 0.5 0.6 0.64 0.65 0.7 0.8 0.9]
   A=[0.3 0.5 0.3 0.3 0.1 1 0.0 0.3 0.3 0.3 0.7 0.2]

7.1.3 Filtros IIR

7.1.3.1 Filtros IIR basados en prototipos analógicos

1. Diseña un filtro butterworth paso alto con una frecuencia de corte de 5000Hz y orden 6 y observa su respuesta en frecuencia utilizando freqz.
   Recuerda que para visualizar la respuesta con los ejes logarítmicos en el eje x debes usar set(gca,'xscale','log').
2. Diseña un filtro chebysev paso bajo con un rizado en la banda de paso de 6 dB, una frecuencia de corte de 300Hz y orden 8. Utiliza una fs=44100Hz
3. Diseña un filtro chebyshev paso bajo con un rizado en la banda atenuada de -20 dB, una frecuencia de corte de 300Hz y orden 8. Utiliza una fs=44100Hz
4. Comprueba la respuesta en frecuencia de los filtros de los ejercicios 2 y 3. Usa un axis ([20 10000 -70 5]).
5. Diseña un filtro elíptico paso bajo con diferentes órdenes(2, 4 y 6) y una frecuencia de corte de 700Hz, un rizado en la banda de paso de 4 dB y un rizado en la banda atenuada de -60 dB. Visualiza la respuesta en frecuencia. Utiliza una fs=48000Hz
6. Comprueba la respuesta de los cuatros tipos de filtros anteriores (butter,cheby1,cheby2 y ellip), realizando un paso bajo con una frecuencia de corte de 500Hz, orden 6, rizado en la banda atenuada -40dB y rizado en la banda de paso 3 dB. Utiliza fs=48000Hz
7. Diseña un filtro bessel paso alto con orden 9, frecuencia de corte 2000Hz. Utiliza una fs de 48000Hz

7.1.3.2 Filtros IIR por aproximación de números cuadrados

1. Diseña u filtro yulewalk de orden 200 utilizando el vector de frecuencias y el de magnitudes siguientes:
   f=[0 200 750 800 1000 2000 2500 5000 8000 9200 12000 16000 18000 24000];
   m=[4 4 2 2 3.5 3.5 -3.2 -3.2 -1.5 -1.5 2 2 -3 0];
   Los ejes deben estar en Herzios(eje x) y en dB(eje y).
2. Observa el filtro de la imagen Ejercicio1_IIR e intenta reproducirlo. Utiliza la función yulewalk.

7.1.3.3 Filtros IIR de 2º orden

1. Diseña un filtro shelving paso bajo con una fc=100Hz y ganancias 2,4,6,-2,-4 y -6. fs=44100. Utiliza la función shelving_low.m y recuerda pasar las ganancias a lineal. G1=2dB, G2=-2dB
2. Implementa una función que se llame shelving_high(G,fs,fc) con la formulación vista en clase.
3. Implementa la formulación de un filtro peak paso banda estricto.
4. Diseña un filtro notch para filtrar la señal a 300 Hz con una Q=0,98. Visualiza la respuesta en frecuencia de la señal filtrada.
5. Implementa la formulación de un filtro paso todo de 2º orden.