UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de una aplicación de apoyo a la asignatura de Análisis de Sistemas y Señales

MATERIAL DIDÁCTICO

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A

Juan José Cortés González

ASESOR DE MATERIAL DIDÁCTICO

Dr. Jesús Manuel Álvarez López

Ciudad Universitaria, Cd. Mx., 2017

AGRADECIMIENTOS

A la Universidad Nacional Autónoma de México por abrirme las puertas del conocimiento y la cultura que habitan dentro de ella.

A la Facultad de Ingeniería por formarme como profesionista y darme una perspectiva distinta del mundo.

A M.I. Juan Manuel Gómez González quien me dio la oportunidad de trabajar a su lado y enriqueció en gran medida este proyecto.

A los miembros del jurado M.I. Daniel Martínez Gutiérrez, Dr. Jesús Manuel Álvarez López, DR. Paul Rolando Maya Ortiz, Ing. Jorge Alberto Solano Gálvez, Ing. Jaime Héctor Díaz Osornio, por su interés en ser sinodales y formar parte de este momento en vida.

A los profesores que tuve durante la carrera, el estudiante es el resultado de la suma de todos los maestros que tuvo.

A mis compañeros de carrera con quienes compartí buenos y malos momentos pero sobre todo muchas desveladas, tareas y proyectos.

A mi madre y mi hermana quienes llenan mi vida.

Para una loca y peligrosa amante de los gatos...

Índice

1 Introducción

En el presente trabajo se describe la elaboración de **Asise** (contracción de *Análisis de Sistemas y Señales*). Es una aplicación para dispositivos móviles desarrollada en el sistema operativo Android y su finalidad es aportar material didáctico de apoyo para la materia de *Análisis de Sistemas y Señales.*

La asignatura *Análisis de Sistemas y Señales* está en la categoría de *ciencias de la ingeniería* y se imparte en el cuarto semestre de la carrera ingeniería eléctrica electrónica, por lo tanto, su papel radica aplicar los conocimientos adquiridos en ciencias básicas hacia las bases de ingeniería. Por lo anterior, es de suma importancia ya que muchas asignaturas de semestres posteriores dependen de los conceptos aprendidos en *Análisis de Sistemas y Señales.* Cabe mencionar que hasta el plan de estudios 2010 esta asignatura era impartida en las otras dos carreras de la División de Ingeniería Eléctrica (ingeniería en computación e ingeniería en telecomunicaciones), actualmente con el plan 2016 estas dos ingenierías llevan otra asignatura de nombre *Sistemas y Señales* cuyo programa de estudio es similar al de *Análisis de Sistemas y Señales*. Por ello, **Asise** también puede ser útil para esa nueva asignatura.

1.1 Educación y nuevas tecnologías

En la última década de siglo pasado Internet se volvió un recurso de fácil acceso a nivel mundial, en el caso de México el crecimiento de este medio se dio con mayor relevancia con la llegada del Siglo XXI. En la Figura 1.1 puede verse el crecimiento de los usuarios de Internet desde 2001 hasta 2014(INEGI, 2015). De acuerdo con otro documento de INEGI (2016) existe un uso diferenciado de Internet dependiendo de la edad y la escolaridad (Figuras 1.2 y 1.3). Como puede observarse en dichas gráficas los jóvenes entre 18 y 24 años que están cursando la educación superior son los usuarios más activos. Se muestran estos datos por dos razones:

1. Internet ha tenido un fuerte impacto social y la educación no es la excepción (Greaves et al., 2012; Tenner, 2013).
2. Derivado de Internet han surgido los dispositivos móviles inteligentes donde el acceso a material educativo es relevante (Tan y Teo, 2015).

El segundo punto es el de mayor interés en el presente trabajo. Los celulares y tablets han tenido una penetración en el mercado sumamente importante debido a la portabilidad de los mismos. De acuerdo con INEGI (2016) hay 77.7 millones de ususarios de celulares, de los cuales el 66.3% (51.51 millones) usan un teléfono inteligente (Figura 1.4). Por lo tanto, es de esperar que el impacto que tuvo Internet en la educación universitaria se vea reflejado en los dispositivos móviles. Este planteamiento puede ser contatado al observar el mercado de Google Play donde existen 41 categorías y educación es la que tiene mayor cantidad de aplicaciones.

Existe la percepción de que las aplicaciones orientadas a redes sociales dominan el mercado y es cierto bajo determinada visión. Facebook, Instagram, Snapshot, Twitter, Whatsapp se encuentran entre las aplicaciones más usadas, pero la categoría social en Google Play no está en el Top 10 (Figura 1.5) porque son pocas las aplicaciones relevantes y las necesidaes son homogéneas. Educación está en primer lugar dado el total de aplicaciones desarrolladas con este propósito, pero en la educación las necesidades son heterogéneas, es decir, una aplicación para niños que están aprendiendo a leer no es relevante para un estudiante de química que requiere una tabla periódica. Dicho de otra forma, hay menos aplicaciones sociales porque el mercado ya está saturado, las principales necesidades de socialización ya están resueltas. En oposición, hay más aplicaciones educativas porque éstas son mucho más específicas y no están resueltas todas las necesidades. Se han mostrado los datos para Google Play dado que **Asise** es un aplicación de Android pero los datos de Apple para Iphone son similares.

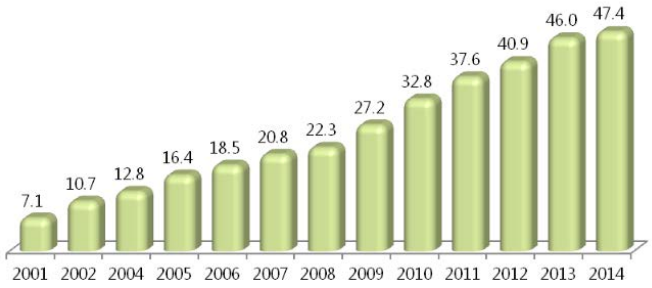


Fig. 1.1 Usuarios de Internet en México 2001-2014 (imagen tomada de INEGI, 2015)

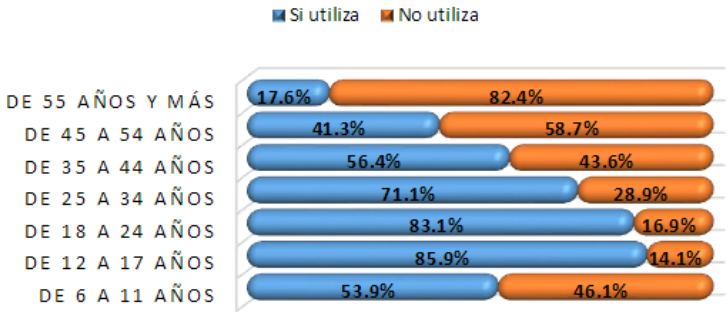


Fig. 1.2. Usuarios de Internet por grupos de edad en México, 2015 (imagen tomada de INEGI, 2016)

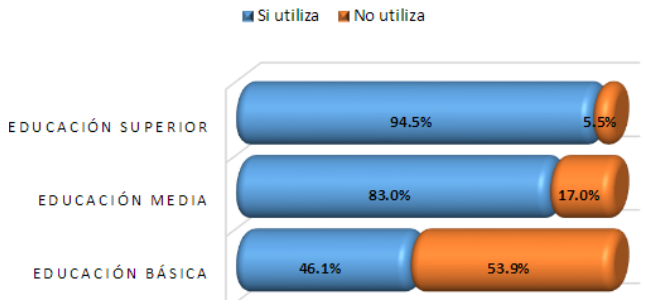


Fig. 1.3. Usuarios de Internet por nivel de escolaridad en México, 2015 (imagen tomada de INEGI, 2016)

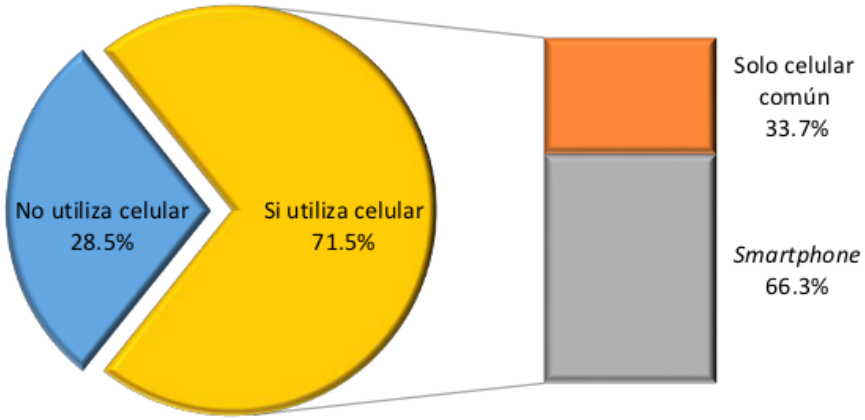


Fig. 1.4. Población según condición de uso de celular, por tipo de equipo (imagen tomada de INEGI, 2015)

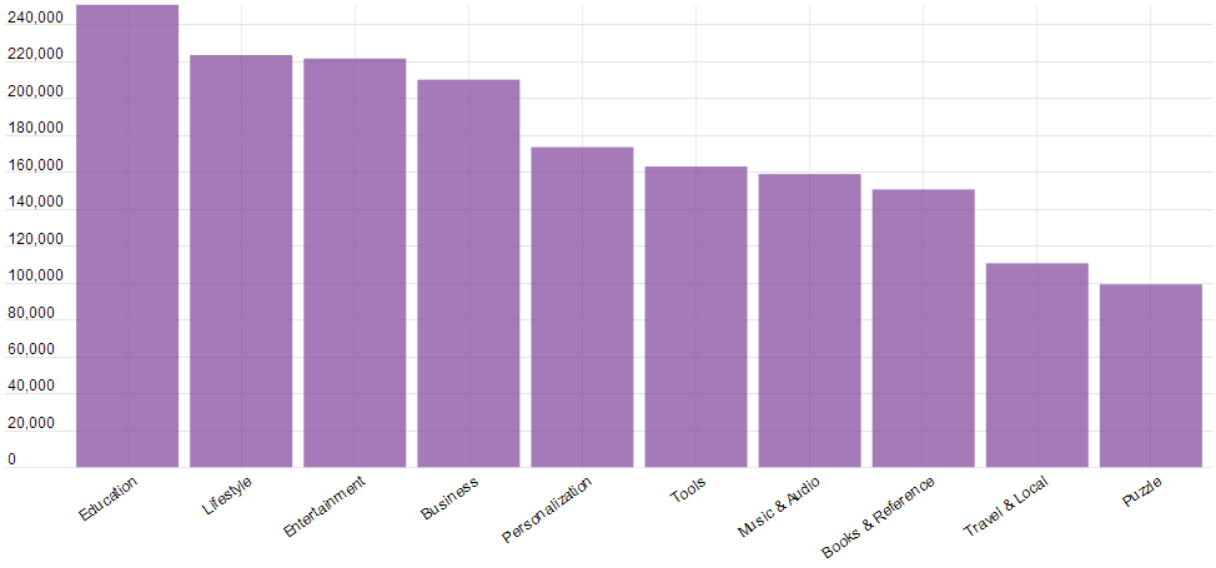


Fig. 1.5. Top 10 de las categorías con más aplicaciones en Google Play actualizado al 17 de julio de 2017 (imagen tomada de AppBrain (1), 2017)

1.2 Aplicaciones móviles educativas

Como se mencionó previamente la categoría Educación es la que contiene más aplicaciones, esta tendencia va a continuar creciendo en los próximos años (Chang, 2017). Dicho crecimiento se debe a diversos factores entre ellos están:

1. Portabilidad
2. Especialización
3. Interactividad y dinamismo
4. Contenido estático

La portabilidad es descrita en inglés como "*anywhere, anytime* " (donde sea y cuando sea). Es relevante ya que no es necesario que los usuarios tengan que esperar o desplazarse para consultar contenido o analizar datos.

La especialización es la clave detrás de la proliferación de aplicaciones educativas, el contenido de una aplicación puede ser desarrollado hacia un grupo de usuarios con necesidades muy específicas. Cabe hacer notar, que dentro de esta variedad de aplicaciones existe cierta popularidad de algunos temas, por ejemplo, el aprendizaje de idiomas es el tema más solicitado (AppBrain (2), 2017).

La interactividad y el dinamismo consisten en generar contenidos que dependen del gusto del usuario y que pueden ser modificados en tiempo real, pero también alude a la capacidad de analizar datos, ya sea porque el usuario los ingresa o porque se adquieren a través de los sensores del dispositivo, por ejemplo, un medidor de decibeles.

El contenido estático se refiere a la parte que es similar a un libro, es decir, texto que no puede ser modificado. Este punto es relevante ya que en las aplicaciones de escritorio, en general, los programas interactivos son distintos de los programas de contenido estático.

Por lo mencionado en los párrafos anteriores, se puede destacar que las aplicaciones para dispositivos móviles combinan la parte interactiva y dinámica de un programa para PC con el contenido teórico de un libro. Es necesario resaltar que las aplicaciones se quedan cortas en ambos sentidos, es decir, no tienen la capacidad de procesamiento de un programa dinámico de escritorio y tampoco tienen el formato adecuado para lectura. Por lo anterior, las aplicaciones móviles no son un reemplazo para estas tecnologías sino un complemento de las mismas, representan una forma distinta de ofrecer material educativo. Este planteamiento puede ser visualizado como se muestra en la Figura 1.6.

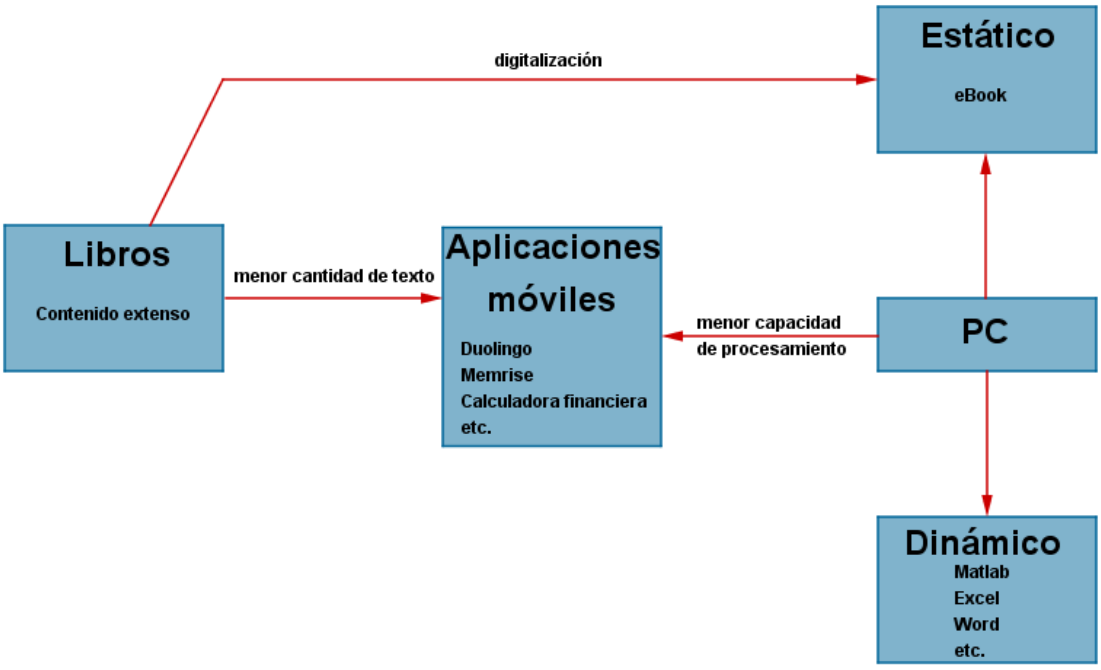


Fig. 1.6. Relación de las aplicaciones móviles con programas para PC y libros.

Otro aspecto de relevancia consiste en las ventajas y desventajas técnicas que requiere el desarrollo de aplicación móvil educativa, a continuación se presentan las más relevantes:

Ventajas

- *Corrección de errores*
- *Incrementar el contenido*
- *Actualizaciones*

Corrección de errores

Cuando existe algún error en la redacción de un texto se puede resolver de una manera muy fácil ya que se realizan las correcciones pertinentes y se publican como actualizaciones.

Incrementar el contenido

Al igual que en el punto anterior, ampliar el contenido de un texto puede realizarse de forma muy sencilla.

Actualizaciones

Este punto se abordó al hablar de corrección de errores y ampliación de contenido pero va más allá de eso, también es posible mejorar la interactividad y dinamismo, agregar nuevas características y dar mantenimiento.

Desventajas

- Lectura en pantalla
- Gama diversa de dispositivos
- Uso de memoria
- Mantenimiento

Lectura en pantalla

Hay muchas personas a las que no les agrada o no les es cómodo leer en pantallas, menos aún cuando dichas pantallas son del tamaño reducido que tienen los dispositivos móviles, particularmente los teléfonos.

Gama diversa de dispositivos

Este aspecto es sobresaliente sobretodo en el sistema operativo Android, ya que a diferencia de IPhone, los dispositivos Android son producidos por una gran variedad de fabricantes donde cada uno utiliza versiones modificadas del sistema operativo. Lo anterior genera que existan incompatibilidades que deben ser tomadas en cuenta al momento de programar.

Uso de memoria

A diferencia de una computadora de escritorio o una laptop los dispositivos móviles tienen una memoria de almacenamiento muy reducida. Lo anterior genera que las aplicaciones deben tener tamaños reducidos.

Mantenimiento

Desde el surgimiento del sistema operativo Android para sistemas móviles en 2008 y hasta la fecha existen 14 versiones de Android hasta la actual que es Android 7.1.2 Nougat. Es decir, 14 versiones en menos de 10 años. Lo anterior, genera que todas las aplicaciones deben estar bajo constante revisión pues con cada versión nueva algunos métodos son depreciados y otros son añadidos además existen cambios más profundos, por ejemplo a partir de Android 6.0 (Marshmallow) la forma de solicitar permisos (conexión a internet, almacenamiento interno, localización, GPS, etc.) cambió radicalmente, lo que provocó que todas las aplicaciones disponibles tuvieran que modificar su código.

En suma, **Asise** no es sustituto para un libro, tampoco es un documento virtual, ni pretende sustituir el uso de software de PC (Matlab, Octave). **Asise** está diseñada para sacar provecho de las tecnologías móviles (teléfonos y tablets), es decir, dar al estudiante otra herramienta de estudio.

Se presenta en los siguientes capítulos el desarrollo de la aplicación **Asise**, primero se muestran los aspectos teóricos que fundamentan el contenido de la aplicación, posteriormente se describen los aspectos técnicos de programación, después se muestran

sus características por medio de un manual de usuario y por último se realizan las conclusiones pertinentes.

La aplicación **Asise** puede ser descargada desde el siguiente:

https://play.google.com/store/apps/details?id=com.labiomedicaunam.fi.asise

1.3 Objetivos

Desarrollar una aplicación para dispositivos Android que contenga material didáctico de apoyo en aspectos conceptuales, prácticos y aplicados, con la finalidad de ayudar a un mejor entendimiento de los temas vistos la asignatura de Análisis de Sistemas y Señales.

2 Aspectos teóricos

En el presente capítulo se detallan los aspectos teóricos detrás de **Asise.** La mayor parte del contenido es muy similar al que se muestra en a aplicación en la sección referente a *Conceptos*. En algunos casos, se modificó el tono del lenguaje para hacerlo más adecuado a una tesina y se agregaron algunas partes que profundizan un poco más en los temas, pero las ecuaciones y los pasos para la resolución de problemas son los mismos.

2.1 Señales en Ingeniería Eléctrica Electrónica

El término señales, en ingeniería eléctrica electrónica, se refiere a la información que obtenemos del mundo físico (analógico) como temperatura, humedad, etc., y que es convertida a fluctuaciones de voltaje, corriente, etc. Todas las señales tienen los mismos elementos en común: Amplitud, frecuencia, fase y una función que depende del tiempo:

$$A\cos(ft+\theta) \qquad (2.1.1)$$

Donde:

A es la amplitud.

cos es la función trigonométrica que depende del tiempo t.

f es la frecuencia.

θ es la fase.

La amplitud es una medida de la energía que contiene la señal. La frecuencia describe la oscilación en el tiempo, puede medirse en Hertz o en radianes por segundo. La fase es un ángulo que indica un atraso o adelanto de la señal, puede indicarse en grados o radianes. Ejemplos de señales son:

$$5\cos(60t+45°) \qquad (2.1.2)$$

Amplitud: 5; frecuencia: 60[Hz]; fase:45°

$$\cos\left(4\pi t+\frac{\pi}{2}\right) \qquad (2.1.3)$$

Amplitud: 1; frecuencia: $4\pi\left[\frac{rad}{s}\right]$; fase: $\frac{\pi}{2}$

Existen muchas formas de clasificar las señales dependiendo de distintas características. Entre dichas clasificaciones están: analógicas y digitales, determinísticas y aleatorias, periódicas y no periódicas. A continuación de da una breve descripción de cada una.

Analógicas: son aquellas que tiene valores para cualquier instante del tiempo, es decir la función que las describe es continua.

Digitales: son aquellas que tiene valores para instantes de tiempo específicos, es decir la función que las describe no es continua.

Determinísticas: son aquellas para las cuales sabemos su valor en cualquier momento, es decir, podemos describirla con una función ya sea continua o discontinua.

Aleatorias: son aquellas para las cuales no sabemos su valor en cualquier momento, por lo tanto, no podemos describirlas con una función.

Periódicas: son aquellas que tienen un patrón ("forma") que se repite cada cierto tiempo.

Aperiódicas: son aquellas que NO tienen un patrón ("forma") que se repite cada cierto tiempo.

Transformaciones y operaciones con señales

Las transformaciones en las señales siempre se llevan a cabo sobre la variable independiente, ya sea t para señales continuas o n para señales discretas. Obviamente al modificar t o n significa que las transformaciones ocurren en el dominio del tiempo.

Escalamiento en el tiempo

Implica multiplicar o dividir la variable independiente por una constante mayor a uno. Cuando se multiplica el resultado es una contracción, cuando se divide el resultado es una expansión.

$$x(t) \xrightarrow{\text{contracción}} x(at) \qquad a > 1 \qquad (2.1.4)$$

$$x(t) \xrightarrow{\text{expansión}} x\left(\frac{t}{a}\right) \qquad a > 1 \qquad (2.1.5)$$

Otra forma de plantear esta transformación consiste en manejar sólo la multiplicación por una constante mayor a cero. En este caso se tendrá una contracción cuando la constante es mayor a uno y una expansión cuando es menor a uno y mayor a cero.

$$x(t) \xrightarrow{\text{contracción}} x(at) \qquad a > 1 \qquad (2.1.6)$$

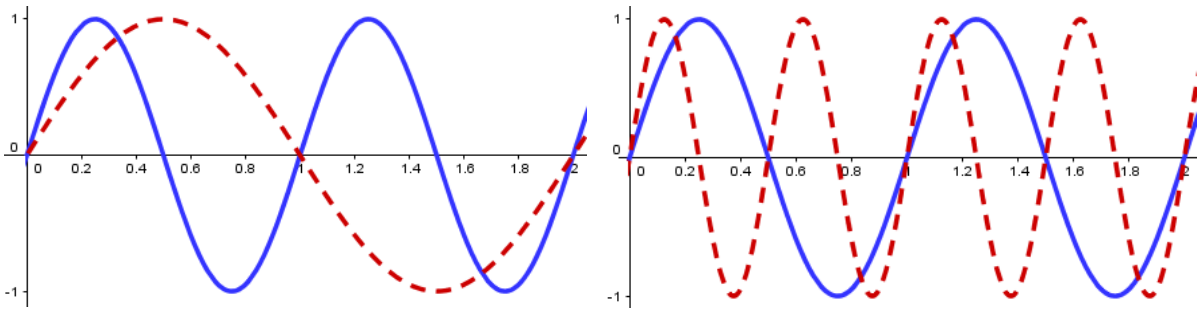
$$x(t) \xrightarrow{\text{expansión}} x(at) \qquad 0 < a < 1 \qquad (2.1.7)$$

Cabe preguntarse ¿Es posible que la constante a se negativa? Sí, pero además del escalamiento se presentará otra transformación: inversión en el tiempo, ésta es explicada más adelante en el texto.

En las Figuras 2.1 (a) y (b) se muestra el escalamiento en el tiempo de una señal $\text{sen}(t)$.

Desplazamiento en el tiempo

Implica sumar o restar a la variable independiente una constante t_0 distinta de cero. Cuando se suma el resultado es un adelanto en el tiempo, cuando se resta el resultado es un atraso en el tiempo.



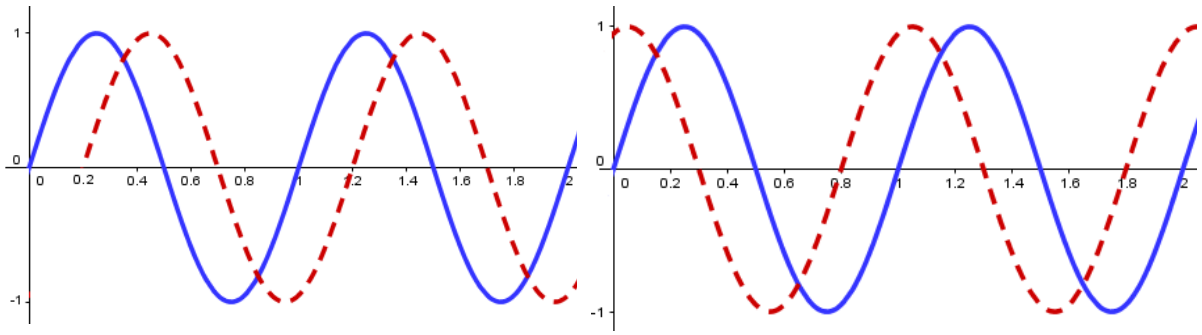
(a) (b)

Fig. 2.1. Escalamiento (líneas discontinuas) en el tiempo de $\text{sen}(t)$(línea continua). (a) $\text{sen}\left(\frac{t}{2}\right)$ expansión,(b) $\text{sen}(2t)$ contracción.

$$x(t) \xrightarrow{adelanto} x(t+t_0) \qquad t_0 \neq 0 \qquad (2.1.8)$$

$$x(t) \xrightarrow{atraso} x(t-t_0) \qquad t_0 \neq 0 \qquad (2.1.9)$$

En las Figuras 2.2 (a) y (b) se muestra el desplazamiento en el tiempo de una señal $\text{sen}(t)$. Notar que de forma gráfica un atraso se muestra como desplazamiento hacia la derecha y un adelanto como un desplazamiento hacia la izquierda.



(a) (b)

Fig. 2.2. Desplazamiento (líneas discontinuas) en el tiempo de $\text{sen}(t)$(línea continua). (a) $\text{sen}(t-0.2)$atraso (b) $\text{sen}(t+0.2)$ adelanto.

Inversión en el tiempo

Implica multiplicar la variable independiente por menos uno. El resultado es una reflexión de la señal para $t = 0$. En la Figura 2.3 se muestra la inversión en el tiempo de una señal $\text{sen}(t)$.

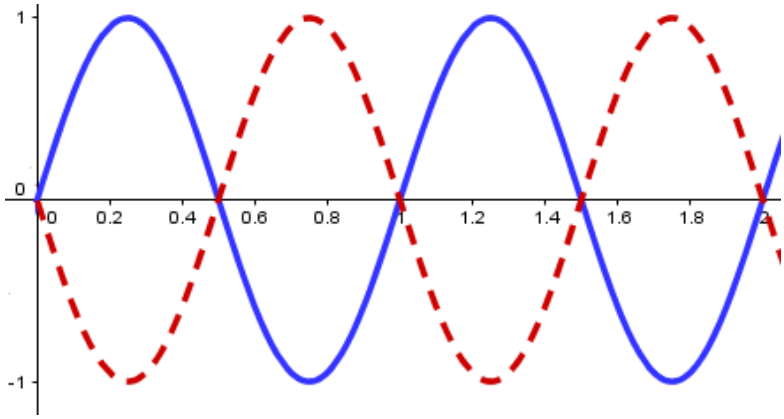


Fig. 2.3. Inversión $\text{sen}(-t)$ (línea discontinua) en el tiempo de $\text{sen}(t)$(línea continua).

Otro aspecto importante en el manejo de señales son las operaciones que se pueden realizar entre dos o más señales. En este sentido, se pueden sumar, restar, multiplicar y dividir entre ellas. El resultado de estas operaciones depende los elementos que cambian entre las señales. Ejemplos:

$$6\cos(2t) + 6\cos(2t) = 12\cos(2t) \qquad (2.1.10)$$

$$6\cos(2t) - 2\cos(2t) = 4\cos(2t) \qquad (2.1.11)$$

$$6\cos(2t) * 4\cos(2t) = 24\cos^2(2t) \qquad (2.1.12)$$

$$\frac{6\cos(2t)}{3\cos(2t)} = 2 \qquad (2.1.13)$$

$$\text{sen}(t) + \cos(t) = \sqrt{2} * \text{sen}\left(\frac{\pi}{4} + t\right) \qquad (2.1.14)$$

En el último caso se tuvo que usar una identidad trigonométrica. Pero lo relevante es que al operar entre señales se modifica el resultado de salida ya que las distintas amplitudes, funciones, frecuencias y fases, interactúan entre sí.

2.2 SLIT (Sistema Lineal Invariante en el Tiempo)

El término sistema tiene distintas acepciones debido a que se utiliza muchas áreas de estudio. Para nuestra área de interés nos referimos a los sistemas como procesos que tienen una señal entrada y una señal salida, donde la salida es una versión modificada de la entrada. La modificación generalmente se indica como una caja cerrada:

$$\xrightarrow{entrada} \boxed{Modificar} \xrightarrow{salida}$$

Luego, a las señales de entrada y salida se les describe como:

$$x(t) \text{ o } x[n]$$

$$y(t) \text{ o } y[n]$$

Donde t se usa para señales continuas mientras que n se emplea en señales discretas. Por lo tanto, el esquema planteado queda:

$$x(t) \longrightarrow \boxed{Modificar} \longrightarrow y(t)$$

$$x[n] \longrightarrow \boxed{Modificar} \longrightarrow y[n]$$

El primer sistema es puramente continuo, mientras que el segundo sólo es discreto. En la práctica es común pasar de señales continuas a discretas y viceversa. Dichos sistemas se llaman convertidores ya sea analógico-digital (ADC) o digital-analógico (DAC). Un ejemplo común de este proceso ocurre al grabar con el celular:

$$x(t) \longrightarrow \boxed{ADC} \longrightarrow y[n]$$

Donde:

$$x(t) = Sonido\ que\ entra\ por\ el\ micrófono$$

$$ADC = Conversión\ Analógica\ Digital$$

$$y[n] = bits\ almacenados\ en\ el\ celular$$

Luego, cuando se reproduce la grabación

$$x[n] \longrightarrow \boxed{DAC} \longrightarrow y(t)$$

Donde:

$$x[n] = bits\ almacenados\ en\ el\ celular$$

$$DAC = Conversión\ Digital\ Analógica$$

$$y(t) = Sonido\ que\ sale\ por\ las\ bocinas$$

Los sistemas tienen distintas características y se clasifican de acuerdo a ellas. Entre estas características están: memoria, causalidad, linealidad, invariante en el tiempo, etc., los sistemas que se prefieren son aquellos que son lineales e invariantes en el tiempo. Un sistema lineal es aquel que cumple con el principio de superposición. Esta es una propiedad matemática que se conoce de otras áreas como el cálculo. Cabe recordar que la integral es un operador lineal, porque la integral de una suma es igual a la suma de las integrales. Efectivamente:

$$\int x^2 + 2x\ dx = \int x^2\ dx + \int 2x\ dx \qquad (2.2.1)$$

Es decir, cumple con la homogeneidad y aditividad. De igual manera, un sistema lineal es aquel para el cual la respuesta de una suma de señales es igual a la suma de las respuestas de cada señal. Efectivamente:

$$a(t) + b(t) \longrightarrow \boxed{ADC} \longrightarrow c[n]$$

Es igual a:

$$a(t) \longrightarrow \boxed{ADC} \longrightarrow a[n]$$

$$b(t) \longrightarrow \boxed{ADC} \longrightarrow b[n]$$

$$a[n] + b[n] = c[n]$$

Un sistema invariante en el tiempo es aquel donde un desplazamiento en la entrada se corresponde con un desplazamiento en la salida. Si:

$$x(t) = y(t)$$

Entonces:

$$x(t - t_0) = y(t - t_0)$$

Por ejemplo, en un sistema cuyo proceso consiste en elevar al cuadrado la señal de entrada, se tiene lo siguiente:

$$x(t) \longrightarrow \boxed{\text{Elevar al cuadrado}} \longrightarrow y(t) = x(t)^2$$

Ahora se realiza un desplazamiento en el tiempo para ello se coloca el desplazamiento posterior al proceso:

$$x(t) \longrightarrow \boxed{\text{Elevar al cuadrado}} \xrightarrow{x(t)^2} \boxed{\text{Desplazamiento}} \longrightarrow y(t) = x(t - t_0)^2$$

Para verificar si es invariante en el tiempo se invierte el orden y se realiza el proceso posterior al desplazamiento:

$$x(t) \longrightarrow \boxed{\text{Desplazamiento}} \xrightarrow{x(t-t_0)} \boxed{\text{Elevar al cuadrado}} \longrightarrow y(t) = x(t - t_0)^2$$

Puede observarse que las salidas son iguales. Es decir, la salida no depende del tiempo en que se aplique la entrada. Por lo tanto, el sistema sí es invariante en el tiempo. Otro

ejemplo: un sistema discreto donde el proceso consiste dividir entre n. Es decir, dividir entre el propio tiempo discreto:

$$x[n] \longrightarrow \boxed{\text{Dividir entre } n} \longrightarrow y[n] = \frac{x[n]}{n}$$

Primero el desplazamiento posterior al proceso:

$$x[n] \longrightarrow \boxed{\text{Dividir entre } n} \xrightarrow{\frac{x[n]}{n}} \boxed{\text{Desplazamiento}} \longrightarrow y[n] = \frac{x[n-n_0]}{n-n_0}$$

Luego, el proceso posterior al desplazamiento:

$$x[n] \longrightarrow \boxed{\text{Desplazamiento}} \xrightarrow{x[n-n_0]} \boxed{\text{Dividir entre } n} \longrightarrow y[n] = \frac{x[n-n_0]}{n}$$

Puede observarse que las salidas no son iguales. Es decir, la salida depende del tiempo en que se aplique la entrada. Por lo tanto, el sistema no es invariante en el tiempo.

2.3 Tiempo discreto y Teorema del muestreo

2.3.1 Tiempo discreto

El término tiempo discreto, está relacionado con señales que tienen valores discretos, es decir, tienen valores sólo en ciertos instantes de tiempo y su dominio siempre está en los números naturales. Si el tiempo es una línea continua:

Entonces el tiempo discreto es una línea discontinua:

Donde la distancia entre los intervalos de tiempo es constante.

Como se mencionó en el apartado anterior. la notación para describir señales que están en tiempo continuo o discreto se distinta. Donde el cambio está en la variable independiente:

t Para tiempo continuo

n Para tiempo discreto

Una de las implicaciones más importantes del tiempo discreto consiste en preguntarse ¿Qué distancia debe existir entre los intervalos de tiempo? Por ejemplo, las siguientes líneas son discontinuas:

Si ambas representan tiempo discreto en un caso dichos los intervalos están más cercanos que en el otro. Es decir, mientras se trabaja en el tiempo continuo puede decirse que este parámetro es único pero al cambiar al tiempo discreto entonces se abre la posibilidad de tener distintos tiempos dependiendo de la distancia entre cada intervalo. La diferencia entre ambos conceptos puede plantearse de la siguiente forma:

- El tiempo continuo es un magnitud que se usa como parámetro y que de acuerdo con la física está presente en todo el universo.
- El tiempo discreto es una magnitud que se usa como parámetro y que de acuerdo con la ingeniería se determina para un sistema en particular.

Por lo tanto, para responder a la pregunta ¿Qué distancia debe existir entre los intervalos de tiempo? Es necesario plantear el teorema del muestreo.

2.3.2 Teorema del muestreo (Nyquist-Shannon)

Este teorema tuvo un desarrollo histórico interesante, de hecho, en la actualidad aún se debate quién o quiénes fueron los primeros en plantearlo y demostrarlo de manera formal. Se respeta en este texto la autoría más aceptada que involucra a Harry Nyquist y Claude Shannon. Para una muy breve, pero interesante revisión histórica consultar Lüke (1999).

Entrado en materia, se puede plantear el teorema del muestreo a partir del siguiente ejemplo: señal: $\cos(t)$ que está en tiempo continuo (Figura 2.4).

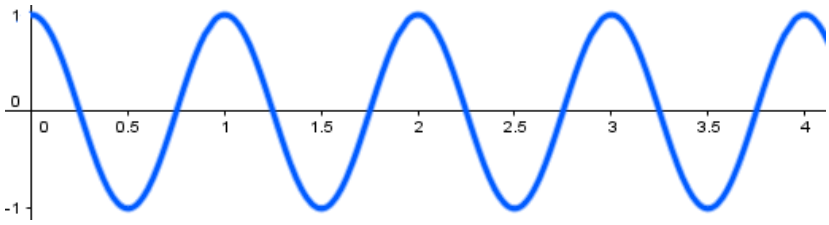


Fig. 2.4 Gráfica de la señal $\cos(t)$

Si se busca representar la misma señal en un tiempo discreto es necesario establecer un intervalo de tiempo tal que la señal no pierda su forma, es decir, información. El teorema del muestreo establece que esa condición se cumple cuando:

$$f_s \geq 2f_{máx} \qquad (2.3.1)$$

Donde f_s es la frecuencia de muestreo y $f_{máx}$ es la frecuencia máxima de la señal. En el ejemplo que se usó:

$$\cos(t) \qquad (2.3.2)$$

La frecuencia máxima está en (t). Por lo tanto:

$$f_s \geq 2[Hz] \qquad (2.3.3)$$

En la Figura 2.5 se muestra donde estarían ubicados los puntos de muestreo para una $f_s = 2[Hz]$.

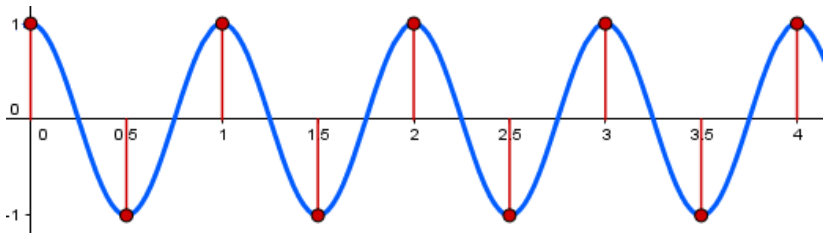


Fig. 2.5 Puntos de muestreo para una $f_s = 2[Hz]$. sobre la señal $\cos(t)$

Al unir los puntos de muestreo se obtiene la forma que adquiere la señal de salida (Figura 2.6).

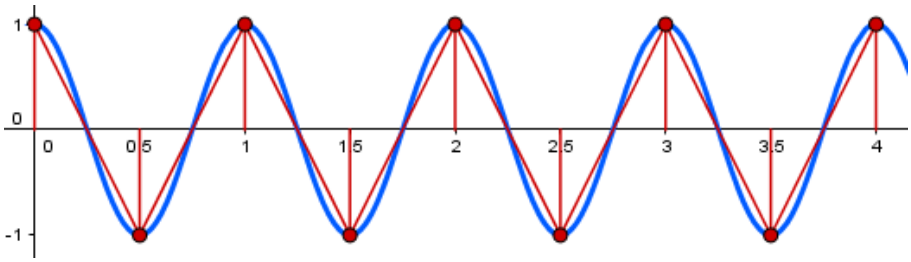


Fig. 2.6 En color azul señal de entrada $\cos(t)$. En color rojo señal de salida $\cos[n]$

Otro concepto importante en el teorema del muestreo es el periodo o tasa de muestreo que es el inverso de la frecuencia de muestreo:

$$T_s = \frac{1}{f_s}[s] \qquad (2.3.4)$$

Por lo tanto, en el ejemplo mostrado:

$$f_s = 2[Hz] \rightarrow T_s = \frac{1}{2}[s] \qquad (2.3.5)$$

En la Figura 2.6 puede notarse que los puntos de muestreo están localizados cada 0.5 [s]. Por otro lado, si la frecuencia de muestreo es menor a la frecuencia máxima se presenta un fenómeno denominado aliasing. Si la frecuencia de muestreo es mucho mayor a la frecuencia máxima ocurre sobre muestreo.

¿Qué es el aliasing? Este término está en inglés pero puede ser fácilmente explicado en español. Alias es sinónimo de apodo, por ello, es común escuchar x persona, alias "*el* y." Con las señales pasa algo parecido cuando se presenta el aliasing una señal toma una forma distinta. Por ejemplo, si se muestrea $\cos(t)$ con una $f_s = \frac{4}{3}[Hz]$ se obtiene el resultado mostrado en la Figura 2.7.

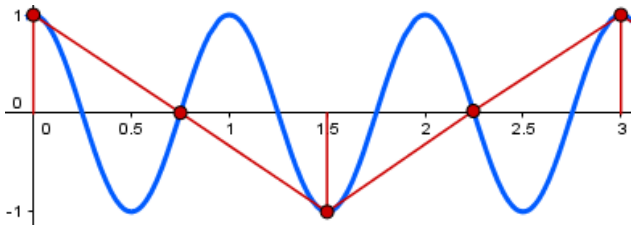


Fig. 2.7 Puntos de muestreo para una $f_s = \frac{4}{3}[Hz]$ sobre la señal $\cos(t)$

Para hacer más evidente la ubicación de los puntos de muestreo que se observan en la Figura 2.7, es necesario considerar la tasa de muestreo:

$$f_s = \frac{4}{3}[Hz] \rightarrow T_s = \frac{3}{4} = 0.75\ [s]$$

Al observa T_s es evidente que los puntos de muestreo están cada 0.75 $[s]$ lo cual coincide con la Figura 2.7. Por otro lado, para ver de forma más notoria el aliasing es necesario mostrar la señal de salida con curvas más suaves (Figura 2.8). En dicha figura se observa que la señal de salida sigue siendo un coseno pero su frecuencia ya es distinta. De hecho, la frecuencia de la señal de salida es:

$$\cos\left[\frac{1}{3}n\right] \qquad (2.3.6)$$

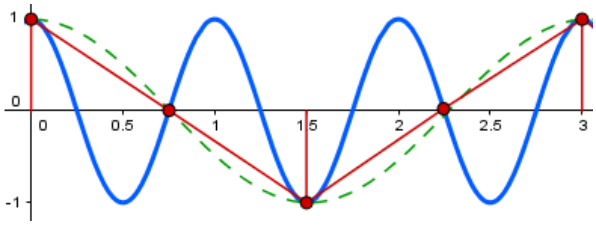


Fig. 2.8 En color azul señal de entrada $\cos(t)$. En color rojo señal de salida $\cos\left[\frac{1}{3}n\right]$. En color verde $\cos\left[\frac{1}{3}n\right]$ con curvas más suaves.

¿Cómo se obtiene la frecuencia de la señal de salida?

Se tiene un sistema de dos ecuaciones:

$$f_o = |f_e - mf_s| \ \ m = 0,1,2,3,\ldots,n \qquad (2.3.7)$$

$$f_o \leq \frac{f_s}{2} \qquad (2.3.8)$$

Donde:

$$f_o: frecuencia\ de\ salida$$

$$f_e: frecuencia\ de\ entrada$$

$$f_s: frecuencia\ de\ muestreo$$

En el ejemplo planteado se tiene:

$$f_e = 1[Hz] \qquad (2.3.9)$$

$$f_s = \frac{4}{3}[Hz] \qquad (2.3.10)$$

$$T_s = \frac{3}{4}[s] \qquad (2.3.11)$$

Aplicando la ecuación (2.3.7):

$$f_o = \left|1 - m\frac{4}{3}\right| \qquad (2.3.12)$$

Para $m = 0$:

$$f_o = |1 - 0| = 1 \qquad (2.3.13)$$

Aplicando la ecuación (2.3.8)

$$f_o \leq \frac{\frac{4}{3}}{2} \qquad (2.3.14)$$

$$1 \leq 0.66 \qquad (2.3.15)$$

No se cumple, Por lo tanto, es necesario probar para $m = 1$:

$$f_o = \left|1 - \frac{4}{3}\right| = 0.33 \qquad (2.3.16)$$

$$0.33 \leq 0.66 \qquad (2.3.17)$$

Sí cumple. Por lo tanto, la señal de salida es:

$$\cos[0.33n] = \cos\left[\frac{1}{3}n\right] \qquad (2.3.18)$$

El resultado puede verificarse de forma gráfica en la Figura 2.8, donde se puede observar que por cada tres ciclos de la señal de entrada se tienen un ciclo de la señal de salida.

2.4 El número de Euler

Este número es de vital importancia en el análisis de sistemas y señales, aparece recurrentemente en distintos cálculos y está involucrado en todas las transformadas que se usan en la asignatura, por ello, se incluye en el presente trabajo una forma para deducir esta constante y ayudar a un mejor entendimiento de su importancia.

El número de Euler fue usado por primera vez por John Napier quien buscaba un proceso para simplificar la multiplicación y la división, Napier usa el número *e* como base para publicar una serie de logaritmos basados en dicho número, otros matemáticos como Leibniz también mencionan el valor de *e*, pero fue Euler quien demostró muchas de sus propiedades y le dio un sentido matemático formal (Tognetti, 1998; Glaz, 2017)

El número e se puede deducir de distintas formas. A continuación se muestra una manera de obtenerlo basado en el planteamiento hecho originalmente por Jacob Bernoulli y solucionado por Euler (Tognetti, 1998; Andreou y Lambright, 2012).

Considerando el siguiente caso:

1) Un sujeto x hace un préstamo por la cantidad de $1 peso a un sujeto y

2) y promete pagarle x a más tardar un mes después

3) Si y paga a mitad del mes pagará el peso más 50 centavos extra

4) Si y paga a fin de mes pagará el peso más $1 peso extra

El planteamiento parece correcto, pero de hecho, no lo es ¿Por qué? Porque a mitad del mes el sujeto y ya debe 1.5 pesos, es decir:

$$1 + \frac{1}{2} \qquad (2.4.1)$$

Quiere decir que el $1 peso ha crecido 50% en medio mes, si a fin de mes paga $2 pesos implica que la segunda mitad del mes el dinero creció menos porque debería aumentar el 50% de lo que ya debía a mitad del mes. Para hacerlo más adecuado, se tendría que calcular un aumento de 50% en ambas mitades del mes considerando el interés que se dio a mitad del periodo:

$$1 + \frac{1}{2} = 1.5 \ (acumulado\ 1er\ mitad\ del\ mes) \qquad (2.4.2)$$

$$1.5 + \frac{1.5}{2} = 2.25\ (acumulado\ 2da\ mitad\ del\ mes) \qquad (2.4.3)$$

Todo el mes en fracciones:

$$1 + \frac{1}{2} + \frac{\left(1 + \frac{1}{2}\right)}{2} = 2.25 \qquad (2.3.4)$$

Lo cual se reduce a:

$$1+\frac{1}{2}+\frac{1}{2}+\frac{1}{4} \qquad (2.4.5)$$

Ordenando las fracciones:

$$1+2\left(\frac{1}{2}\right)+\frac{1}{4} \qquad (2.4.6)$$

Se observa que el segundo término es igual a 1 pero hay una razón para ponerlo de esta forma. Es más, aunque parezca inútil se multiplica ese término por otro 1:

$$1+2(1)\left(\frac{1}{2}\right)+\frac{1}{4} \qquad (2.4.7)$$

La razón para escribirlo así es porque, de esta forma, se convierte en un trinomio cuadrado perfecto:

$$a^2+2ab+b^2$$

Donde:

$$a=1 \text{ y } b=\tfrac{1}{2}$$

Efectivamente:

$$1+2(1)\left(\frac{1}{2}\right)+\frac{1}{4}=1^2+2(1)\left(\frac{1}{2}\right)+\left(\frac{1}{2}\right)^2 \qquad (2.4.8)$$

Factorizando el polinomio anterior:

$$\left(1+\frac{1}{2}\right)^2=2.25$$

Euler generalizó y demostró esta fórmula:

$$\left(1+\frac{1}{n}\right)^n \qquad (2.4.9)$$

La cual permite hacer el siguiente planteamiento:

¿Qué ocurre si en vez de dividir el mes en 2 partes se divide en 3, 4 ó quizá en 30 partes?:

$$\left(1+\frac{1}{2}\right)^3=2.37$$

$$\left(1+\frac{1}{2}\right)^4=2.44$$

$$\left(1+\frac{1}{2}\right)^{30}=2.67$$

Se obtiene una tasa de crecimiento que tiene una característica muy importante: Depende de lapsos de tiempo más cortos, es decir, tiende a ser continua. Considerando los minutos en un mes de 30 días:

$$\left(1+\frac{1}{43200}\right)^{43200}=2.718250$$

Los segundos en ese mismo lapso de tiempo:

$$\left(1+\frac{1}{2592000}\right)^{2592000}=2.718281$$

De esta forma, se puede establecer la continuidad cuando el lapso de tiempo tiende a ser infinitamente pequeño. Lo cual permite plantear un límite:

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^{n} \qquad (2.4.10)$$

Al resolver este límite se obtiene el número e:

$$e=2.718281828\ldots$$

Es decir, se obtiene la tasa de crecimiento que tiene el $1 peso si éste comienza a generar intereses desde una fracción de tiempo infinitamente pequeña justo después de que se dio el préstamo. Lo anterior, es muy útil ya que se puede saber exactamente cuánto tiene que pagar el sujeto y dependiendo del momento en que acuda a saldar su deuda con el sujeto x. Esta tasa de crecimiento se considera como natural por las siguientes razones:

1. Describe la razón de cambio de un determinado fenómeno cuando el tiempo es continuo.
2. Dicha razón de cambio depende de sí misma, pero también es igual a sí misma.

a) Depende de sí misma. Esto obedece a que el interés se va acumulando a la cantidad de la cual se obtiene dicho interés, por ello va incrementando (interés compuesto).

b) Igual a sí misma. Esta característica proviene del cálculo diferencial. Cuando se determinó la derivada de la funciones exponenciales, se encontró que en general:

$$\frac{d}{dx}(a^x)=a^x\ln(a) \qquad (2.4.11)$$

Es decir, la derivada de cualquier número a elevado a una variable x es el mismo número elevado a dicha variable x por el logaritmo natural del número a. Por ejemplo:

$$\frac{d}{dx}(2^x)=(2^x)\ln(2) \qquad (2.4.12)$$

$$\frac{d}{dx}(3^x) = (3^x)\ln(3) \qquad (2.4.13)$$

Al calcular los logaritmos en las ecuaciones anteriores se obtiene:

$$\frac{d}{dx}(2^x) = (2^x)(0.693147)$$

$$\frac{d}{dx}(3^x) = (3^x)(1.098612)$$

Los matemáticos observaron estos resultados y notaron que:

$$\ln(2) < 1$$

$$\ln(3) > 1$$

Por lo tanto, debía existir un número entre 2 y 3 cuyo logaritmo fuera igual a 1, por lo tanto, su derivada sería:

$$\frac{d}{dx}(a^x) = a^x(1) \qquad (2.4.14)$$

$$\frac{d}{dx}(a^x) = a^x \qquad (2.4.15)$$

Es decir, la función y su derivada serían iguales. Dicho número es e. Por ello, se dice que la razón de cambio es igual a sí misma. Esta propiedad es conocida sólo para este número.

Un último aspecto que conviene resaltar es que el planteamiento de Bernoulli tiene como objetivo duplicar la cantidad de entrada. Pero es más relevante que sólo eso. Por ejemplo, el crecimiento de una célula, su objetivo es reproducirse, luego, el proceso no estará completo hasta que dicha reproducción se haya completado, es decir, la célula se haya duplicado a sí misma. Lo cual, representa un crecimiento de la masa igual al 100%. Este último planteamiento puede modificarse si lo que se busca es un crecimiento mayor o menor al 100%. Por ejemplo:

$$e^{0.25x} = 25\%\ de\ crecimiento$$

$$e^{0.5x} = 50\%\ de\ crecimiento$$

$$e^{x} = 100\%\ de\ crecimiento$$

$$e^{2x} = 200\%\ de\ crecimiento$$

$$e^{10x} = 1000\%\ de\ crecimiento$$

2.5 Variables complejas en las transformadas

En el análisis de sistemas y señales hay cuatro transformadas que se usan reiteradamente: Laplace, Fourier, z y DFT (Transformada Discreta de Fourier). De estas cuatro transformaciones Laplace y Fourier operan en el dominio del tiempo continuo, mientras que z y DFT son las versiones discretas de Laplace y Fourier respectivamente (Kallolkar y Salunke, 2015; Mata et al., 2002; Oppenheim y Willsky, 1997;). En cuanto a la relación entre Laplace y Fourier, también es muy cercana ya que la segunda es un caso particular de la primera (Anumaka, 2012; Kallolkar y Salunke, 2015;).

El desarrollo matemático de estas transformadas es abundante en la literatura, la intención en el presente trabajo no es repetir dicho desarrollo sino abordar un aspecto particular de estas transformaciones: la variable compleja que las hace funcionar. Por ejemplo, en Laplace se tiene a la variable s y en Fourier a $j\omega$, pero ¿Cómo operan estás variables complejas? o mejor dicho ¿Cómo realizan la transformación que les da nombre?

Considerando que la transformada de Laplace es la más general de las cuatro y que de hecho, las otras tres son casos particulares de ésta, el desarrollo que se presenta en este texto se enfoca en la variable s. A continuación se presentan dos apartados, uno para el desarrollo matemático y otro para la interpretación física, en unidades, de la variable mencionada.

2.5.1 Construcción matemática

Este desarrollo se basa en el planteamiento de Mattuck (2003). Algunos de los conceptos fueron ampliados ya que Mattuck da por sentado su previo conocimiento.

Recordando las series de potencias que en general tienen la forma:

$$\sum_{n=0}^{\infty} c_n(x-a)^n \qquad (2.5.1)$$

Al expandir esta serie se tiene:

$$c_0(x-a)^0 + c_1(x-a)^1 + c_2(x-a)^2 + \cdots \qquad (2.5.2)$$

Donde el primer término se reduce a:

$$c_0(x-a)^0 = c_0 \qquad (2.5.3)$$

Esto aplica aun cuando:

$$x = a \qquad (2.5.4)$$

Es decir:

$$c_0 * 0^0 = c_0 \qquad (2.5.5)$$

Por lo cual, se tiene una convención que aplica sólo a las series:

$$0^0 = 1$$

Pero en general:

$$0^0 = indeterminado$$

Por lo anterior, la expansión de la serie queda:

$$c_0 + c_1(x-a)^1 + c_2(x-a)^2 + \cdots \qquad (2.5.6)$$

Para el siguiente ejemplo:

$$\sum_{n=0}^{\infty} \frac{(x-1)^n}{n!} \qquad (2.5.7)$$

Al expandir la serie se tiene:

$$1 + \frac{(x-1)^1}{1!} + \frac{(x-1)^2}{2!} + \frac{(x-1)^3}{3!} + \cdots \qquad (2.5.8)$$

Donde al comparar la ecuación (2.4.7) con la fórmula general de la ecuación (2.4.1) es claro que $a = 1$, pero ¿A qué es igual c_n?

Reescribiendo la expansión de la ecuación (2.4.8) para que sea más claro:

$$1 + \frac{1}{1!}(x-1)^1 + \frac{1}{2!}(x-1)^2 + \frac{1}{3!}(x-1)^3 + \cdots \qquad (2.5.9)$$

De esta forma se puede notar que c_n es igual a $\frac{1}{n!}$

Lo que se ha mostrado hasta ahora sirve para plantear los siguientes puntos:

1) Las series son infinitas

2) Tienen dos constantes (c_n, a) y una variable x

3) El valor de c_n cambia porque depende de n

Las series de potencias tienen otros aspectos a considerar antes de llegar a Laplace. Algunas cosas se muestran de forma muy resumida ya que la intención de este texto no es profundizar en estas series.

La primer simplificación a realizar es quitar la constante a, esto es posible ya que la forma $(x-a)$ sirve para determinar dónde está centrada la serie. Pero en muchos casos, entre ellos el que interesa en este trabajo, las series están centradas en cero por lo que la forma general se reduce a:

$$\sum_{n=0}^{\infty} c_n x^n \qquad (2.5.10)$$

Luego es necesario considerar que las series de potencias son útiles cuando convergen porque pueden aproximar funciones. Por ejemplo, la serie:

$$\sum_{n=0}^{\infty} x^n = 1 + x + x^2 + x^3 + \cdots \qquad (2.5.11)$$

Tiene un intervalo de convergencia:

$$-1 < x < 1$$

Es decir, cuando x es mayor a 1 o menor a -1 la serie crece y diverge. Si se respeta el intervalo de convergencia, la serie se aproxima a la función:

$$\frac{1}{1-x}$$

Por lo tanto se puede expresar que:

$$\sum_{n=0}^{\infty} x^n \approx \frac{1}{1-x} \ldots \qquad (2.5.12)$$

En general, se dice que una serie de potencias cuando converge se aproxima a una función:

$$\sum_{n=0}^{\infty} c_n x^n \approx C(x) \qquad (2.5.13)$$

Donde la letra C que denota función suele ponerse en mayúscula Cabe mencionar que las series de Taylor y Maclaurin son casos particulares de las series de potencias, se menciona aquí para reforzar la idea de funciones expresadas como series, por ejemplo:

$$\sum_{n=0}^{\infty} \frac{x^n}{n!} \approx e^x \qquad (2.5.14)$$

También es posible expresar las funciones trigonométricas y logarítmicas de esta misma forma. Es decir, expresar funciones trascendentes (donde la variable x es argumento de otra función o exponente de alguna constante), como suma de polinomios. Lo anterior, hace posible que una calculadora pueda dar valores como:

$$\text{sen}(24)$$

$$\cos(\pi)$$

$$\ln(4.269)$$

Se mencionó que el valor de c_n cambia porque depende de n. Por ello, se puede reescribir la ecuación (2.4.13) como:

$$\sum_{n=0}^{\infty} c(n)x^n \approx C(x) \qquad (2.5.15)$$

Donde el término c_n se ha reescrito como $c(n)$ dada la dependencia de c con n ¿Es posible realizar este cambio? Sí, porque c es una constante que toma distintos valores dependiendo de n, donde n sólo adquiere valores enteros.

Es importante hacer notar que en la ecuación (2.4.15) se tiene del lado izquierdo una suma discreta que depende de n pero del lado derecho se obtiene una función continua que depende de x, como ocurre con la ecuación (2.4.14).

¿Qué ocurre si se lleva esta idea a una forma continua en ambos lados de la ecuación?

Primero, es necesario cambiar la sumatoria por una integral:

$$\int_{n=0}^{\infty} c(n)x^n dn \approx C(x) \qquad (2.5.16)$$

Luego cambiar la variable n que se usa generalmente como variable discreta, por una continua que se generalmente se denomina t:

$$\int_{t=0}^{\infty} c(t)x^t dt \approx C(x) \qquad (2.5.17)$$

Después se cambia la letra c por una f por la costumbre de poner así las funciones y se quita $t = 0$ del extremo de integración porque generalmente no se escribe, dejando únicamente el cero:

$$\int_{0}^{\infty} f(t)x^t dt \approx F(x) \qquad (2.5.18)$$

Ahora el término $x(t)$ complica el proceso ya que tiene a la variable de integración t como exponente de una base x. Para simplifcar el cálculo se busca que la base sea el número de Euler en lugar de x y que de esta forma sea más fácil el procedimiento. Sin embargo, no se puede incluir de forma arbitraria. Para realizar el cambio se plantea que:

$$x = e^{\ln(x)} \qquad (2.5.19)$$

La ecuación anterior funciona ya que e y ln se anulan. Luego x^t, por leyes de los exponentes:

$$x^t = \left(e^{\ln(x)}\right)^t \qquad (2.5.20)$$

Es necesario observar que en la ecuación (2.4.18) la variable de integración t puede ir hasta infinito, así lo indican los extremos de la integral. Por lo anterior, el término x^t nunca va a converger si x es mayor que 1. Por ejemplo, si $x = 1$ y $t = 100$:

$$1^{100} = 1$$

Pero, si $x = 1.1$:

$$1.1^{100} = 13780.6123$$

Diverge.

Mientras que si $x = 0.9$:

$$0.9^{100} = 0.00002656$$

Converge, es decir, se busca cuidar que x^t no crezca indefinidamente ya que se partió de un planteamiento para series de potencias donde son útiles cuando convergen. Hasta el momento se tiene que:

$$x < 1 \qquad (2.5.21)$$

Por otro lado, se busca evitar que la base x sea un número negativo porque se podría presentar el caso donde $x = -1$ y $t = 0.5$, que generaría:

$$(-1)^{0.5} = \sqrt{-1}$$

Por esta razón, no existen logaritmos cuya base sea un número negativo ya que pueden llevar a números imaginarios. Asimismo, se elude usar cero como base, en este caso podría darse que $x = 0$ y $t = 0$. Como se mencionó previamente en las series existe una solución para 0^0pero no aplica para la integral. Por lo anterior, se tiene que x debe ser menor a 1 y mayor a cero:

$$0 < x < 1 \qquad (2.5.22)$$

Aplicando esta condición al cambio en la base que se está planteando, el logaritmo natural de 1 es cero, pero el logaritmo natural de 0 no existe. Por lo tanto, todos los posibles valores de $\ln(x)$ serán negativos:

$$\ln(1) = 0$$

$$\ln(0.5) = -0.6931 \ldots$$

$$\ln(0.1) = -2.3025 \ldots$$

$$\ln(0.00001) = -11.5129 \ldots$$

Es decir, que aplicando la condición $0 < x < 1$ a $\ln(x)$ el resultado será menor a cero y mayor a menos infinito:

$$-\infty < \ln(x) < 0$$

Ahora, para evitar tener como variable a $\ln(x)$ se propone un cambio de nombre y se denomina como s:

$$s = \ln(x) \qquad (2.5.23)$$

Se mencionó que $\ln(x)$ siempre va a ser negativo. Por lo tanto, para tener como resultado un valor positivo es necesario hacer que:

$$s = -\ln(x) \qquad (2.5.24)$$

Por último, se multiplica por -1 a ambos lados para dejar $\ln(x)$ sin alteración:

$$-s = \ln(x) \qquad (2.5.25)$$

Retomando la ecuación (2.5.18):

$$\int_0^\infty f(t)x^t dt \approx F(x) \qquad (2.5.18)$$

Aplicando los cambios mostrados a la ecuación (2.5.18). Se comienza sustituyendo con $x = e^{\ln(x)}$:

$$\int_0^\infty f(t)\left(e^{\ln(x)}\right)^t dt \approx F(e^{\ln(x)}) \qquad (2.5.26)$$

Pero recién se mencionó que $\ln(x) = -s$:

$$\int_0^\infty f(t)(e^{-s})^t dt \approx F(e^{-s}) \qquad (2.5.27)$$

Notar que también se cambio $F(x)$ por $F(e^{-s})$ dado que la variable de la cual depende la función de salida ahora cambió de nombre. Sin embargo, es común denotar $F(e^{-s})$ como $F(s)$ ya que ambas significan los mismo: una función que depende de una variable llamada s, recordar que e es sólo un número, la dependencia es con la variable. Lo último que falta es aplicar las leyes de los exponentes en:

$$(e^{-s})^t = e^{-st}$$

Quedando:

$$\int_0^\infty f(t)e^{-st} dt \approx F(s) \qquad (2.5.28)$$

Esta expresión es la transformada unilateral de Laplace. Donde generalmente se cambia el símbolo de *aproximadamente* por *igual a*, se asume que al ser continua se aproxima lo suficiente como para establecer una igualdad:

$$\boxed{\int_0^\infty f(t)e^{-st} dt = F(s)} \qquad (2.5.29)$$

Se puede establecer que la transformada de Laplace sirve para tomar una función que depende de una cierta variable (generalmente x o t) y transformarla en otra función que depende de otra variable llamada s. Este cambio de variable implica también un cambio de dominio y por ende de unidades.

Los puntos más relevantes del desarrollo mostrado son:

1) Las series de potencias tienen variables discretas que pueden expresar funciones continuas.

2) Debido a lo anterior, se pueden expresar funciones trascendentes (exponenciales, trigonométricas, etc.) como polinomios (funciones algebraicas).

3) Al llevar una serie de potencias a una forma continua se puede obtener la transformada de Laplace.

4) La esencia de las series de potencias se mantiene: hay cambio de variable y construcción de polinomios.

5) Los polinomios que resultan siempre contienen a la nueva variable s. Por ejemplo, la función trascendente coseno:

$$\cos(at) \qquad (2.5.30)$$

En la transformada de Laplace resulta en el polinomio:

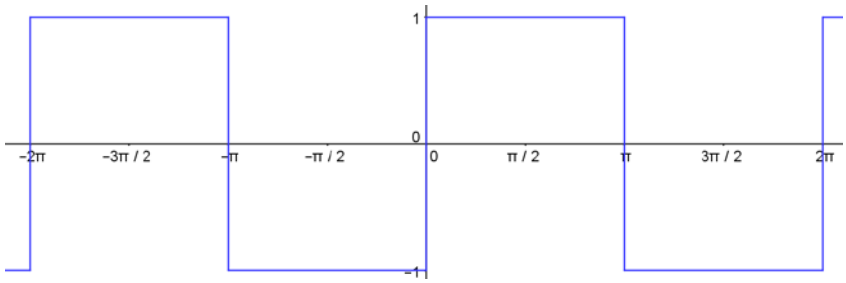
$$\frac{s}{s^2 + a^2} \qquad (2.5.31)$$

2.6 Series trigonométricas de Fourier

En el tema referente a las transformadas se optó por mostrar un aspecto teórico particular de dichas transformaciones, la variable compleja que las hace funcionar abordando su deducción matemática y su interpretación física. En el tema de series trigonométricas de Fourier se decidió hacer lo opuesto y avocarse por un planteamiento completamente práctico. En este sentido la aplicación **Asise** contiene un apartado dedicado a este tema que cuenta con siete ejercicios. En todos los casos, el desarrollo es muy detallado y no se omiten pasos. A continuación se muestra uno de los siete ejercicios. La base para la resolución de problemas se obtuvo principalmente de Adkins y Davidson (2010), Theraja y Theraja (2005) y Tretter (2013).

En este caso se respeta el lenguaje como viene en **Asise**, es decir, hablando en segunda persona ya que al tratarse de la resolución de un problema, este tipo de lenguaje es más cercano al lector.

Determina la serie trigonométrica de Fourier de la siguiente señal:



Paso 1: Escribe la señal como una función a trozos:

$$f(t)\begin{cases} 1 \text{ para } 0 \leq t < \pi \\ -1 \text{ para } \pi \leq t > 2\pi \end{cases} \qquad (2.6.1)$$

Paso 2: Escribe la ecuación general para las series de Fourier:

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(n\omega_0 t) + b_n \operatorname{sen}(n\omega_0 t) \qquad (2.6.2)$$

Paso 3: Determina el periodo (T) y la frecuencia angular (ω_0):

$$T = 2\pi \qquad (2.6.3)$$

$$\omega_0 = \frac{2\pi}{T} = 1 \qquad (2.6.4)$$

Paso 4: Determina el coeficiente a_0:

La fórmula general para a_0 es:

$$a_0 = \frac{2}{T}\int_T f(t)dt \qquad (2.6.5)$$

Recuerda que:

$$\int_T \qquad (2.6.6)$$

Significa integrar en todo un periodo.

Colocando el dato del periodo y considerando que es una función a trozos:

$$a_0 = \frac{2}{2\pi}\left[\int_0^{\pi} 1\, dt + \int_{\pi}^{2\pi} -1\, dt\right] \qquad (2.6.7)$$

Evaluando las integrales:

$$a_0 = \frac{1}{\pi}\left[t\Big|_0^{\pi} - t\Big|_{\pi}^{2\pi}\right] \qquad (2.6.8)$$

Simplificando:

$$a_0 = \frac{1}{\pi}[(\pi - 0) - (2\pi - \pi)] \qquad (2.6.9)$$

Por lo tanto:

$$a_0 = 0$$

Paso 5: Determina el coeficiente a_n:

Por observación, se puede determinar que a_n es cero ya que la función que se pide tiene simetría non. Sin embargo, se muestra el desarrollo con fines didácticos:

La fórmula general de a_n:

$$a_n = \frac{2}{T}\int_T f(t)\cos(n\omega_0 t)\,dt \qquad (2.6.10)$$

Colocando los datos que tenemos para periodo y frecuencia radial

$$a_n = \frac{2}{2\pi}\int_0^{2\pi} f(t)\cos(n1t)\,dt \qquad (2.6.11)$$

Haciendo las simplificaciones obvias en:

$$\frac{2}{2\pi} \text{ y } (n1t)$$

Queda:

$$a_n = \frac{1}{\pi}\int_0^{2\pi} f(t)\cos(nt)\,dt \qquad (2.6.12)$$

Luego, recuerda que es una función a trozos, por lo tanto, se tiene la constante = 1 en un intervalo y -1 en otro intervalo. Por ello, es necesario separar la integral en dos:

$$a_n = \frac{1}{\pi}\left[\int_0^{\pi} 1\cos(nt)\,dt + \int_{\pi}^{2\pi} -1\cos(nt)\,dt\right] \qquad (2.6.13)$$

Simplificando los unos y los signos:

$$a_n = \frac{1}{\pi}\left[\int_0^{\pi} \cos(nt)\,dt - \int_{\pi}^{2\pi} \cos(nt)\,dt\right] \qquad (2.6.14)$$

Resolviendo las integrales:

$$a_n = \frac{1}{\pi}\left[\frac{\text{sen}(nt)}{n}\Big|_0^{\pi} - \frac{\text{sen}(nt)}{n}\Big|_{\pi}^{2\pi}\right] \qquad (2.6.15)$$

Evaluando cada integral:

$$\frac{\text{sen}(nt)}{n}\Big|_0^{\pi} = \frac{\text{sen}(n\pi)}{n} - \frac{\text{sen}(0)}{n} \qquad (2.6.16)$$

$$-\frac{\text{sen}(nt)}{n}\Big|_{\pi}^{2\pi} = -\frac{\text{sen}(n2\pi)}{n} + \frac{\text{sen}(n\pi)}{n} \qquad (2.6.17)$$

Recuerda que:

$$\text{sen}(0) = \text{sen}(n\pi) = \text{sen}(n2\pi) = 0$$

Para valores enteros de n. Por lo tanto:

$$a_n = \frac{1}{\pi}[0] = 0 \qquad (2.6.18)$$

Paso 6: Determina el coeficiente b_n:

La fórmula general de b_n:

$$b_n = \frac{2}{T}\int_T f(t)\,\text{sen}(n\omega_0 t)\,dt \qquad (2.6.19)$$

Colocando los datos que tenemos para periodo y frecuencia radial

$$b_n = \frac{2}{2\pi}\int_0^{2\pi} f(t)\,\text{sen}(n1t)\,dt \qquad (2.6.20)$$

Simplificando:

$$b = \frac{1}{\pi}\int_0^{2\pi} f(t)\,\text{sen}(nt)\,dt \qquad (2.6.21)$$

Es una función a trozos:

$$b_n = \frac{1}{\pi}\left[\int_0^{\pi} 1\,\text{sen}(nt)\,dt + \int_{\pi}^{2\pi} -1\,\text{sen}(nt)\,dt\right] \qquad (2.6.22)$$

Resolviendo las integrales:

$$b_n = \frac{1}{\pi}\left[-\frac{\cos(nt)}{n}\Big|_0^{\pi} + \frac{\cos(nt)}{n}\Big|_{\pi}^{2\pi}\right] \qquad (2.6.23)$$

Evaluando cada integral:

$$-\frac{\cos(nt)}{n}\Big|_0^{\pi} = -\frac{\cos(n\pi)}{n} + \frac{\cos(0)}{n} \qquad (2.6.24)$$

$$\frac{\cos(nt)}{n}\Big|_{\pi}^{2\pi} = \frac{\cos(n2\pi)}{n} - \frac{\cos(n\pi)}{n} \qquad (2.6.25)$$

Recuerda que:

$$\cos(0) = 1$$

Además, en la sumatoria de la serie de Fourier sólo nos importan los enteros:

$$\cos(n2\pi) = \cos(0)$$

Para cualquier valor entero de n. Por lo tanto, la evaluación de las integrales se simplifica:

$$-\frac{\cos(nt)}{n}\Big|_{0}^{\pi} = -\frac{\cos(n\pi)}{n} + \frac{1}{n} \qquad (2.6.26)$$

$$\frac{\cos(nt)}{n}\Big|_{\pi}^{2\pi} = \frac{1}{n} - \frac{\cos(n\pi)}{n} \qquad (2.6.27)$$

Quedando:

$$b_n = \frac{1}{\pi}\left[-\frac{\cos(n\pi)}{n} + \frac{1}{n} + \frac{1}{n} - \frac{\cos(n\pi)}{n}\right] \qquad (2.6.28)$$

Sumando términos semejantes:

$$b_n = \frac{1}{\pi}\left[-\frac{2\cos(n\pi)}{n} + \frac{2}{n}\right] \qquad (2.6.29)$$

Factorizando $\frac{2}{n}$:

$$b_n = \frac{1}{\pi}\left[\frac{2}{n}(-\cos(n\pi) + 1)\right] \qquad (2.6.30)$$

Simplificando:

$$b_n = \frac{2}{\pi n}[-\cos(n\pi) + 1] \qquad (2.6.31)$$

Se ha llegado a un resultado correcto para b_n, sin embargo, es común realizar más simplificaciones para dejar la serie de una forma más compacta y para que al programarla el tiempo de ejecución se reduzca. La simplificación comienza observando que:

$$\cos(n\pi)$$

Alterna entre 1 y -1 dependiendo de los valores enteros de n:

$$\cos(\pi) = -1$$

$$\cos(2\pi) = 1$$

$$\cos(3\pi) = -1$$

$$\cos(4\pi) = 1$$

Es decir, vale 1 cuando *n* es par y -1 cuando es non. Recuerda que los valores 1 y -1 alternados se pueden expresar como:

$$(-1)^n$$

Efectivamente:

$$(-1)^1 = -1$$

$$(-1)^2 = 1$$

$$(-1)^3 = -1$$

$$(-1)^4 = 1$$

Sustituyendo con este término en el coseno del resultado de b_n:

$$b_n = \frac{2}{\pi n}[-(-1)^n + 1] \qquad (2.6.32)$$

Notemos que para valores pares:

$$[-(-1)^{2,4,6\cdots} + 1] = [-(1) + 1] = 0$$

Para valores nones:

$$[-(-1)^{1,3,5\cdots} + 1] = [-(-1) + 1] = 2$$

Podemos decir que el valor de b_n cambia dependiendo del valor de n. Cuando es par:

$$b_n = \frac{2}{\pi n}[-(-1)^n + 1] = 0 \qquad (2.6.33)$$

Cuando es non:

$$b_n = \frac{2}{\pi n}[-(-1)^n + 1] = \frac{2}{\pi n}[2] = \frac{4}{\pi n} \qquad (2.6.34)$$

Escribiendo esta condición de manera formal se tiene:

$$b_n = \begin{cases} 0 \text{ para } n = \text{par} \\ \dfrac{4}{\pi n} \text{ para } n = \text{non} \end{cases} \qquad (2.6.35)$$

Piénsalo de esta forma, si escribes una serie bajo la condición que se ha encontrado tendrías:

$$término\ de\ n = 1 + 0 + término\ de\ n = 3 + 0 + \cdots$$

¿Para qué sirve escribir los términos que son cero? Sólo nos interesa cuando *n* es non.

Paso 7: Ingresa los coeficientes para a_0 , a_n y b_n en la fórmula general (ecuación 2.5.2):

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(n\omega_0 t) + b_n \operatorname{sen}(n\omega_0 t)$$

Luego, considerando que a_0 y a_n son cero:

$$\sum_{n=1}^{\infty} b_n \operatorname{sen}(n\omega_0 t) \qquad (2.6.36)$$

Pero también se encontró que b_n sólo importa cuando n es non. Es decir, buscamos:

$$\sum_{n=1}^{\infty} \frac{4}{\pi n} \operatorname{sen}(n\omega_0 t) \qquad (2.6.37)$$

Recordando que los nones se pueden obtener con:

$$2n - 1 \qquad (2.6.38)$$

Sustituimos con (2.5.38) en los términos donde está n:

$$\sum_{n=1}^{\infty} \frac{4}{\pi(2n-1)} \operatorname{sen}((2n-1)\omega_0 t) \qquad (2.6.39)$$

Observa que podemos sacar $\frac{4}{\pi}$ de la sumatoria, para dejar en ella sólo los términos que contienen a n:

$$\frac{4}{\pi} \sum_{n=1}^{\infty} \frac{\operatorname{sen}((2n-1)\omega_0 t)}{(2n-1)} \qquad (2.6.40)$$

Por último, dijimos que $\omega_0 = 1$:

$$\boxed{\frac{4}{\pi} \sum_{n=1}^{\infty} \frac{\operatorname{sen}((2n-1)t)}{(2n-1)}} \qquad (2.6.41)$$

Esta expresión es la serie ya terminada.

3 Aspectos Técnicos

3.1 Android

Asise está desarrollada en el sistema operativo Android, dicho sistema tiene un Kernel de Linux que es desarrollado por la compañía Google a través AOSP (Android Open Source Project). Por lo tanto, Android es un proyecto de código abierto, lo anterior genera la existencia de diversas distribuciones para una misma versión de Android, por ejemplo, para la versión 7 (Nougat) Google tiene su distribución que instala en teléfonos Nexus, pero otras compañías como Samsung, Verizone, Amazon, Motorola, entre muchas otras, también crean sus distribuciones. Independientemente de la distribución la arquitectura del sistema operativo Android es general , en la Figura 3.1 se muestra dicha arquitectura donde se observan cuatro capas:

1. Aplicaciones: es la capa más próxima al usuario, son la aplicaciones con las que se interactúa de forma directa.
2. Armazón de aplicaciones (Application Framework): Son API's que permiten la interacción de alto nivel con el sistema operativo. Esta capa generalmente no es usada por el usuario promedio sino por los desarrolladores.
3. Librerías y Runtime: esta capa contiene las librerías que permiten modificar el armazón de aplicaciones para realizar funciones como almacenamiento de datos, render de gráficos, etc. Asimismo, incluye el Android Runtime y las librería del núcleo de java para poder ejecutar aplicaciones. Esta capa es usada por desarrolladores avanzados para generar librerías o modificar las existentes.
4. Kernel de Linux: es la capa que se encarga de la comunicación con el hardware (pantalla, cámara, audio, wifi, etc.). Esta capa no es usada por desarrolladores de aplicaciones sino por compañías que modifican el Kernel para obtener su propia distribución.

Android tiene una historia breve pero sus desarrollo en términos de versiones ha sido destacado, en la Tabla 3.1 se muestran las diferentes versiones de este sistema operativo. Puede observarse que en el año 2009 se lanzaron cuatro versiones distintas, a partir de 2012 con Jelly Bean, Google ha sacado una versión principal al año con algunas versiones derivadas. Cabe también señalar que las versiones se deprecian rápidamente, en la actualidad sólo las versiones de KitKat y posteriores se mantienen soportadas. No se incluye en la lista Android 8.0 (API 26) porque su lanzamiento oficial no ha ocurrido aunque para los desarrolladores la plataforma de desarrollo comenzó a llegar desde junio del presente año.

Cuando surgió Android la IDE (Integrated Development Environment) que se usaba era Eclipse con un plugin para Android, esta forma de desarrollo resultaba un tanto engorrosa ya que era necesario seguir varios pasos: instalar Java, instalar Eclipse, dar de alta la dirección web del plugin para Android (se ubicaba en la página de desarrolladores de Google para Android). Para buscar actualizaciones era necesario realizarlo de forma

manual conectándose vía el plugin. Posterior a esta etapa, Google creó una versión modificada de Eclipse llamada Eclipse ADT (Android Development Tools), la cual se descargaba desde el sitio de desarrolladores. Este paso fue de gran ayuda, ya no era necesario instalar Java por separado y las actualizaciones llegaban de forma automática. Sin embargo, la IDE tenía constantes *bugs* que generaban retrasos en la programación de aplicaciones, sobre todo cuando se lanzaba una nueva versión del sistema operativo.

Fig. 3.1. Estructura general de Android (imagen tomada, androidstudio, 2017)

Google encargó a la empresa IntelliJ IDEA una IDE que fue lanzada al mercado como Android Studio en mayo de 2013. Durante algún tiempo Google mantuvo soporte para ambas IDE's (Eclipse ADT y Android Studio) mientras los desarrolladores se acostumbraban al nuevo entorno, pero desde 2015 Eclipse ADT está depreciada. Actualmente Android Studio está en la versión 2.3.3 (junio de 2017) y es la plataforma de desarrollo más usada.

Al ser código abierto existen muchas otras IDE's para Android como Xamarin, AIDE, Cordova, PhoneGap, Cocos2d, Unity, Processing, etc. Algunas de ellas no son específicas

para Android sino IDE's multiplataforma que permiten la creación de aplicaciones para distintos sistemas operativos.

Tabla 3.1. Versiones de Android (elaborada con datos de wikipedia.org y android.com)

Nombre	Versión	Nivel de API	Lanzamiento	Soporte
Nougat	7.0 – 7.1.2	24 -25	22/agosto/2016	Con soporte
Marshmallow	6.0-6.01	23	5/octubre/2015	Con soporte
Lollipop	5.0-5.1.1	21-22	12/noviembre/2014	Con soporte
KitKat	4.4-4.4.4	19-20	31/octubre/2013	Con soporte
Jelly Bean	4.1-4.3.1	16-18	9/julio/2012	Sin soporte
Ice Cream Sandwich	4.0-4.0.4	14-15	18/octubre/2011	Sin soporte
Honeycomb	3.0-3.2.6	11-13	22/febrero/2011	Sin soporte
Gingerbread	2.3-2.3.7	9-10	6/diciembre/2010	Sin soporte
Froyo	2.2-2.2.3	8	20/mayo/2010	Sin soporte
Eclair	2.0-2.1	5-7	26/octubre/2009	Sin soporte
Donut	1.6	4	15/septiembre/2009	Sin soporte
Cupcake	1.5	3	27/abril/2009	Sin soporte
Sin nombre clave	1.1	2	9/febrero/2009	Sin soporte
Sin nombre clave	1.0	1	23/septiembre/2008	Sin soporte

El problema con IDE's externas consiste en que algunas son muy específicas para un fin en particular, como el desarrollo de juegos (Unity) o se desarrollan en un lenguaje muy distinto a Java, el cual ha sido la base de Android desde su aparición y que se comparte entre los desarrolladores, por ejemplo Xamarin se programa en C#. Otras están orientadas la creación de aplicaciones móviles a partir de lenguaje web (html, CSS y JavaScript) como Cordova y PhoneGap, a este tipo de desarrollo se le denomina programación híbrida. En este caso el lenguaje no es la principal barrera ya que el desarrollo web es común entre los programadores, el problema radica en que sólo se puede usar dicho lenguaje, lo cual funciona para cierto tipo de aplicaciones muy específicas. Pero el manejo del hardware de los dispositivos o el procesamiento de datos no es posible, en este sentido algunos desarrolladores han sugerido incluir un intérprete de PHP que funcione en Android y así

poder ejecutar tareas más complejas, han existido algunos intentos como el fallido PFA (PHP For Android). Actualmente, Cordova y PhoneGap ofrecen la ejecución de PHP pero dicha ejecución se realiza en servidor no al interior del dispositivo.

Un último caso de interés es Processing, este lenguaje se puede usar de dos formas: como una librería en integrarla en Android Studio o descargando su IDE para Android. Para un desarrollo básico y observar las capacidades de Processing la segunda opción es adecuada, pero en un proyecto más demandante es necesario usar la versión librería.

En el desarrollo de **Asise** se usó Android Studio más la inclusión de distintas librerías. Se implementaron otros lenguajes como html, pero la hibridación se realizó aledaña a la programación principal, es decir, se trata de una implementación pero no es el núcleo principal de desarrollo.

3.2 Lenguajes en **Asise**

Java y Xml

Como puede deducirse del apartado anterior en Android es posible usar lenguajes distintos. De manera estándar, el desarrollo se realiza usando Java y Xml, donde Java lleva toda la carga dinámica de los widgets (botones, barras deslizantes, layouts, vistas, campos de texto, etc.), el procesamiento de datos, la ejecución y compilación del código, etc., mientras que Xml sirve para todas las declaraciones estáticas de los mencionados widgets. Dicho de otra forma, en Xml se crean los widgets junto con sus atributos estáticos, por ejemplo el nombre del widget, en Java se manejan los métodos que controlan el comportamiento de cada widget. Por lo tanto, se dice que Android usa Programación Orientada a Objetos, dada la dependencia con Java. Lo anterior, trae consigo el uso de herencia simple, interfaces, clases abstractas, manejo de excepciones, etc., todas las características de este paradigma de programación. En el caso de **Asise** el manejo de widgets se realizó de manera estándar con declaraciones estáticas en Xml para los atributos y manejo dinámico en Java con métodos.

Html, CSS y JavaScript

La inclusión de lenguaje web fue necesaria dados dos requerimientos de **Asise**:

1. Textos largos con formato justificado, distintos tipos de letra (cursiva, negrita o normal) y colores en algunas palabras.
2. Despliegue de ecuaciones (uso de Latex).

El primer punto ha sido una carencia histórica de Android, los widgets para mostrar texto como EditText o TextView no cuentan con un atributo o método que permita justificar el texto que despliegan, existen librerías creadas por desarrolladores para solucionar este aspecto pero tienden a presentar incompatibilidades entre distintas versiones de Android, además dichas librerías pueden tener problemas para mostrar caracteres con acento. Por

otro lado, el cambio de tipo de letra y colores en algunas palabras tampoco son posibles. La declaración de forma estática sería:

```
android:textColor="#000000"
android:textStyle="italic"
```

Pero esto afecta a todo el widget. De forma dinámica para el color se tiene:

```
nombre_del_widget.setTextColor(Color.parseColor("#000000"));
```

Para el tipo de letra no hay un método, es posible realizar el cambio pero es necesario crear un Xml específico que *infle* (rellene) la vista del widget. Pero no tiene caso realizar este proceso porque la forma dinámica sirve para cambiar el color de todo el contenido del widget ante un evento, por ejemplo, al presionar un botón todo el texto cambia a rojo con letra cursiva, pero no es posible cambiar sólo un fragmento del texto.

Latex

Con respecto al despliegue de ecuaciones Android tampoco posee atributos o métodos. Para lograr este objetivo es necesario usar un lenguaje especializado como Latex. La inclusión de este lenguaje trae consigo diversos problemas ya que debe realizarse a través de una librería externa.

Para el desarrollo de **Asise** se exploraron tres opciones distintas:

1. Hacer el render de Latex directamente en el TextView
2. Una librería para Java que pueda renderear Latex
3. Una librería en JavaScript que hiciera el render de Latex para Html

Antes de describir con detalle los tres puntos mencionados, es necesario mencionar que el render de Latex depende de un motor. Dicho motor también es una librería y determina qué símbolos se pueden mostrar, pero también define el tiempo que tarda el render en ser completado. En este sentido se probaron los siguientes motores:

1. jqMath
2. KaTeX
3. MathJax

Retomando la cuestión de las librerías. Para conseguir el punto 1 y lograr el render directamente sobre el TextView se usó MathView la cual puede usar como motores KaTeX y MathJax. La implementación resultó exitosa con ambos motores pero el tiempo de renderado se incrementaba notablemente al incluir más ecuaciones. Además, persistía el problema del justificado y tipos de letra, para solucionar este problema se buscó alternar widgets WebView con MathView para mostrar en los primeros el texto y en los segundo

las ecuaciones, pero se tornaba impráctico en el desarrollo de ejercicios dada la cantidad de ecuaciones que llevan éstos.

Otra librería que se exploró fue Android Katex, como su nombre lo indica está diseñada para usar KaTeX como motor. Esta librería tiene algunos atributos gráficos destacados, pero tiene un defecto heredado de KaTeX que consiste en comandos de Latex no soportados.

Para el punto 2, se buscaron librerías de Java como jLaTexMath y JMathTex, el problema de esta solución consiste en que se ubica a mitad del camino, es decir, pueden ser implementadas en Java, pero hacerlas funcionar con el ambiente gráfico de Android implicaba emular el trabajo ya resuelto en la librerías de punto 1. Esta tarea representaría un desarrollo en sí mismo que va más allá de lo objetivos del presente trabajo.

Por último, la implementación del punto 3 usando JavaScript fue la solución adecuada dado que el Html ya resolvía los problemas mencionados previamente (justificación, etc.), además, el desarrollo para el render de Latex en páginas web está muy avanzado. El principal problema en este caso fue el tamaño de la librería. Por un lado, el motor jqMath es excelente en términos de velocidad y tamaño pero el despliegue de ecuaciones largas o con ciertos operadores matemáticos, como la integral, no es adecuado en Android. KaTeX, funciona bien y sigue teniendo un tamaño aceptable pero los términos que llevan subíndices como a_n, no se muestran de forma correcta. La más completa y que no presenta problemas en el despliegue es MathJax, pero la librería tiene un tamaño superior a los 100 Mb, lo cual la hacía inviable para una aplicación móvil. El tamaño de esta librería se debe a que está diseñada para instalarse en un servidor y contiene muchas características útiles en ese ámbito. Por lo tanto, se modificó la librería de forma manual hasta dejar en ella sólo los componentes mínimos necesarios que se necesitaban en Android. Esta tarea fue laboriosa y en muchas ocasiones se realizó por prueba y error, pero se logró dejar la librería en tan sólo 2Mb.

3.3 Programación híbrida

Por los expuesto en el apartado anterior puede notarse que **Asise** tiene un componente de programación híbrida que se usó para desplegar textos largos justificados que requerían el uso de ecuaciones. En este sentido, **Asise** utiliza dos motores jqMath para ecuaciones sencillas y MathJax para aquellas que son más demandantes de recursos, pero nunca se utilizan ambos motores en un sólo documento. En la Figura, 3.2 se muestra la implementación de Latex dentro de la **Asise**, puede observarse que la capa más interna contiene Latex que es rendereado por una librería-motor jqMath o MathJax las cuales se implementan en un documento con html, CSS y JavaScript, donde este documento se despliega en el widget WebWiew que es un objeto de la clase homónima, la cual es manejada por la librería WebKit de Android (aparece también en la Figura 3.1) escrita en

c/c++ que se conecta con Java por medio de JNI (Java Native Interface). En todos los casos se creó un documento Html guardado el directorio *assets* de del proyecto.

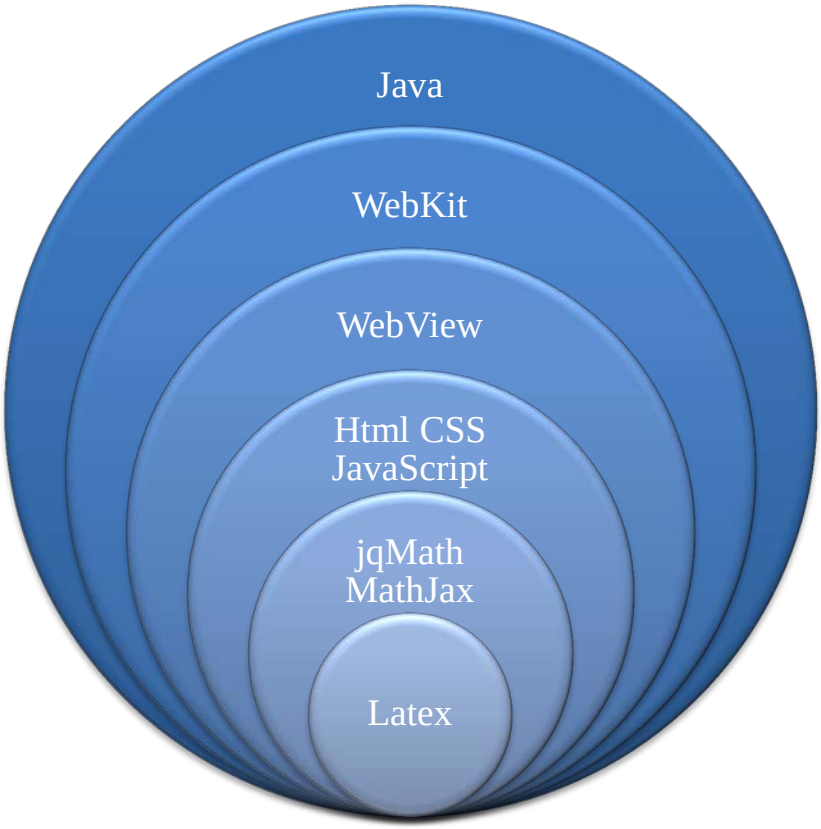


Fig. 3.2. Implementación de Latex en **Asise**.

La hibridación también se usa en aplicaciones sin texto o con texto escaso que requieren mostrar ecuaciones ya sea porque son resultado de algún cálculo o porque se necesita desplegar una fórmula en particular. En este caso, la implementación se realizó de forma similar a la expuesta previamente, la diferencia radica en la ausencia del documento Html, en su lugar se declaró una variable tipo *String* con todo el contenido, incluyendo las librerías, posteriormente se paso dicha variable como argumento del método *loadDataWithBaseURL* de la clase WebWiew. A continuación se muestra un segmento de código con esta implementación.

```
WebView webView = (WebView)vista.findViewById(R.id.fourier_WB);
WebSettings webSettings = webView.getSettings();
webSettings.setDefaultFontSize(letraWeb);
webSettings.setJavaScriptEnabled(true);
formula="\\frac{4}{\\pi}\\sum_{n=1}^{\\infty}\\frac{\\textrm{sen}(\\,(2n-1)\\omega_{0}
t)}{2n-1}";
String funcion="f(t)=\\left\\{\\begin{matrix}\n" +
        " 1\\textrm{ }\\textrm{ }\\textrm{ } 0\\leq t< \\pi, & \\\\ \n" +
        "-1\\textrm{ }\\textrm{ } \\textrm{ }\\pi\\leq t< 2\\pi, & \n" +
```

```
        "\\end{matrix}\\right.";
formulaHTML = "<html><head>"
        + "<meta charset='utf-8'>"
        + "<script type='text/javascript' src =
'file:///android_asset/MathJax.js?config=TeX-AMS_SVG.js'></script>"
        + "<script src='file:///android_asset/jquery-3.1.1.min.js'
type='text/javascript'></script>"
        + "</head><body>"
        +"<p style='color: #0F056F;text-align:center;'><b>Señal cuadrada</b><br>"
        +"Como función: $$"+funcion+"$$"
        +"Periodo: \\(T_{0}=2\\pi\\) <br>Frecuencia: \\(\\omega_{0}=\\frac{2\\pi}{T_0}=1\\)"
        + "</p>"
        +"<p style='color: #0F056F;text-align:center;'>"
        +"Serie trigonométrica: $$"+formula+"$$"
        +"</p>"
        +"</body>"
        +"</html>";
webView.loadDataWithBaseURL("", formulaHTML, "text/html", "UTF-8", "");
```

En este código puede observarse en primer lugar la declaración del WebView, cabe hacer notar que la palabra **vista** se debe a que se está trabajando en un *fragmento* no en un *actividad* este aspecto es abordado con detalle más adelante en el texto. En las siguientes tres líneas se declaran los preparativos (settings) de la vista web lo cual incluye un tamaño para la letra y el permiso para ejecutar JavaScript.

Posteriormente, se tienen tres variables tipo *String* una local *String funcion* y dos globales *formula* y *formulaHTML*, la variable local se definió así porque sólo es necesaria en el ámbito de este WebView, en cambio las dos globales se usan en otros métodos, por ello tienen ese tipo de asignación.

Puede notarse que las cadenas *funcion* y *formula* están escritas en Latex, donde fue necesario incluir doble el caracter de escape '\', es un requerimiento de Java. La variable de mayor interés es *formulaHTML*, dicha cadena contiene toda la información de un documento Html incluyendo la ubicación de las librerías, asimismo, esta variable se concatena a las otras dos variables delimitando la concatenación con los operadores '$$', que indican donde opera el motor de Latex, en este caso MathJax.

Por último, *loadDataWithBaseURL* es un método vació (void) que tiene los siguientes parámetros: loadDataWithBaseURL(String baseUrl, String data, String mimeType, String encoding, String historyUrl). En este caso los parámetros inicial y final no son necesarios porque se accede de forma interna al contenido, es decir, no se tiene una URL de la cual dependa la información mostrada. Los otros tres parámetros se definen como:

- String data: formulaHTML
- String mimeType: text/html
- String encoding: UTF-8

El resultado de esta hibridación se muestra en la Figura 3.3.

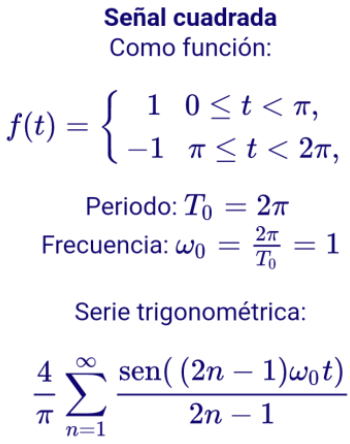


Fig. 3.3. Resultado del código híbrido

3.4 Diálogos de alerta

El uso de diálogos de alerta en **Asise** es importante, ya sea para ingresar datos o para mostrar información relevante, En general, un diálogo de alerta tiene la apariencia mostrada en la Figura 3.4, en **Asise**, se usan de esta forma tradicional para los casos simples (ingreso de datos, Figura 3.5). Para contenido más extenso, producto de botones de *Ayuda* en los cuales es necesario mostrar instrucciones de uso, se modificaron los diálogos de alerta para conseguir el resultado que se observa en la Figura 3.6, puede notarse un texto largo con instrucciones y un botón para cerrar el diálogo.

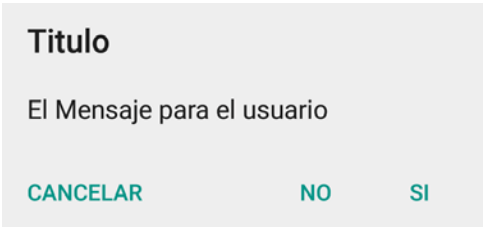


Fig. 3.4. Diálogo de alerta genérico en Android

Fig. 3.5. Diálogo de alerta para ingreso de datos

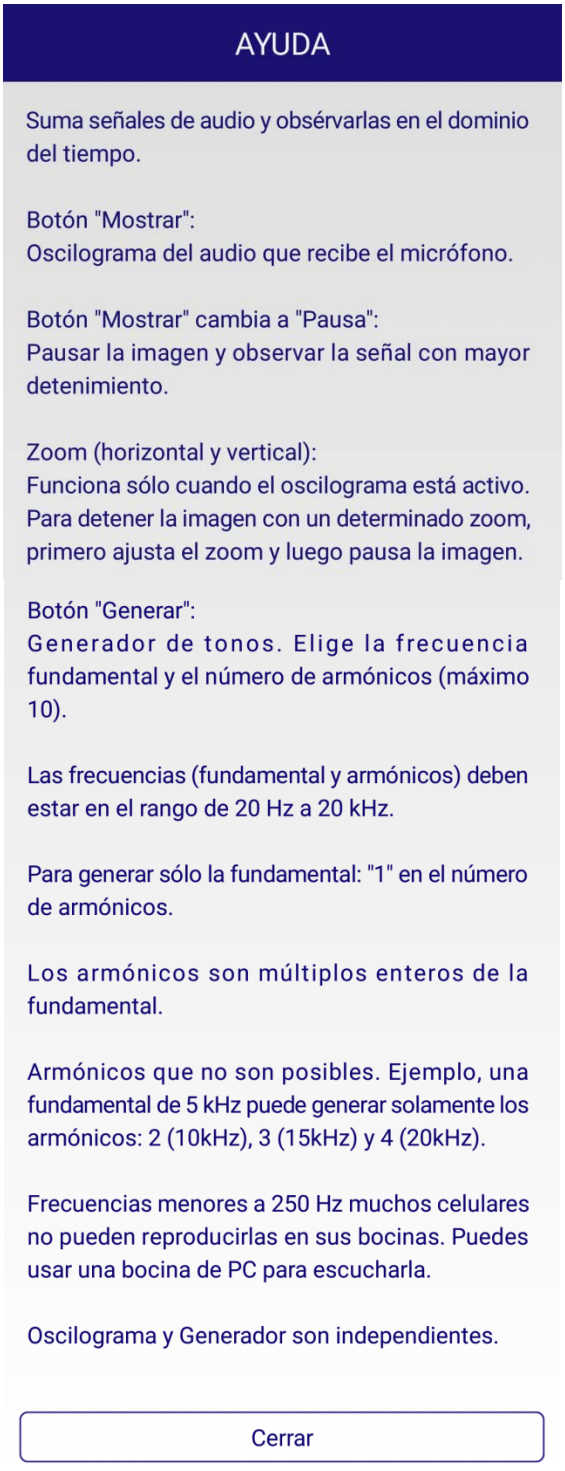


Fig.3.6. Diálogo de alerta modificado

Para lograr esta implementación fue necesario crear un Xml con un *layout* que contiene tres widgets: un TextView normal, un TextView modificado y un RadioButton, posteriormente declarar el *layout* sólo en el contexto del diálogo de alerta e *inflar* la vista de dicho diálogo. El TextView modificado es resultado de la librería externa Android-JustifiedTextView, esta

librería fue descartada para su uso en el caso del texto largo con ecuaciones donde se usó programación híbrida. Sin embargo, para los requerimientos de los diálogos de alerta fue de gran utilidad. En el siguiente segmento de código puede verse un ejemplo de su implementación:

```
ayudaSintetizadorRB.setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View v) {
     ayuda2Segmented.clearCheck();
     AlertDialog.Builder builder4 = new
AlertDialog.Builder(getActivity(),android.R.style.Theme_Light_NoTitleBar_Fullscreen);
     View vistaDialogo4 =
getActividad().getLayoutInflater().inflate(R.layout.dialogo_ayuda_sintetizador,null);
     builder4.setView(vistaDialogo4);
     AlertDialog dialogoAlerta4 = builder4.create();
dialogoAlerta4.getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWA
YS_VISIBLE);
     dialogoAlerta4.show();
     TextView ayudaDialogoSintetizadorET =
(TextView)vistaDialogo4.findViewById(R.id.ayudaDialogoSintetizadorTV);
     RadioButton cerrarAyudaDialogoSintetizadorRB =
(RadioButton)vistaDialogo4.findViewById(R.id.cerrarAyudaDialogoSintetizadorRB);
     ayudaDialogoSintetizadorTV.setTextSize(letraTextView);
     cerrarAyudaDialogoSintetizadorRB.setTextSize(letraTextView);
     ayudaDialogoSintetizadorTV.setText(archivoAyuda());
     cerrarAyudaDialogoSintetizadorRB.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
             dialogoAlerta4.dismiss();
         }
     });
 }
});
```

En este código lo primero que se nota es el método asociado al botón que abre el diálogo de alerta, después se limpia el *check* de dicho botón ya que se trata de un RadioButton. Posteriormente, se crea una instancia del diálogo de alerta donde el método *getactivity()* es necesario pues dicho diálogo se ejecuta dentro de un *fragmento* y donde el estilo se definió para ocupar la pantalla completa. Después, se crea una *vista* en la cual el diálogo se *infla* con el contenido de un Xml llamado *dialogo_ayuda_sintetizador*. Luego, se crea el diálogo con el contexto, estilo y vista señalados.

Todos los diálogos de alerta activan el teclado y lo traen a la vista, pero en este caso no se necesita, por ello se manda esconder con *SOFT_INPUT_STATE_ALWAYS_VISIBLE*. En seguida, se manda mostrar el diálogo y se declaran los widgets con los que va a interactuar:

- TextView modificado (*ayudaDialogoSintetizadorTV*)
- RadioButton (*cerrarAyudaDialogoSintetizadorRB*).

El TextView normal no se declara aquí porque sólo tiene contenido estático, es el encargado de mostrar la palabra *Ayuda* en el encabezado del diálogo. A los widgets declarados se les asigna un tamaño de letra y para cada uno se ejecuta un método diferente.

En el caso de *ayudaDialogoSintetizadorTV* se declara su contenido con *setText(archivoAyuda())*, donde *archivoAyuda()* es otro método que abre un archivo con formato de texto plano (*.txt*). El botón solamente tiene un *OnClickListener* que cierra el diálogo de alerta.

3.5 Librerías

Por las necesidades de **Asise** el uso de librerías externas fue relevante. En las secciones anteriores ya se mencionaron algunas de ellas, así como su uso (jqMath, MathJax y Android-JustifiedTextView). Otras librerías empleadas fueron:

- AndroidPlot
- Processing
- DDF Minim
- Waveform
- MultiSlider
- ACheckBox
- Android-Segmented-Control

AndroidPlot

Esta librería se usó para las gráficas procedentes de funciones trigonométricas y cuyo cálculo se realiza empleando los métodos de la clase Math de Java. En muchos casos era necesario tener un eje horizontal con valores tipo texto como 2π, en AndroidPlot es posible establecer intervalos para el dominio usando valores a partir de pi:

```
plot.setDomainStep(StepMode.INCREMENT_BY_VAL, Math.PI/2);
```

Pero estos valores se muestran como números, por ejemplo, 1.5707, 3.1415, etc. Por otro lado, es posible mostrar cadenas en el dominio pero es necesario declarar dos arreglos estáticos, uno con los valores del rango, que puede estar vacío, y otro con los del dominio, donde estos últimos se someten a un *parse* que permite mostrar cadenas. Esta solución no era funcional en **Asise** porque no se tiene un arreglo de valores estáticos, sino arreglos dinámicos creados a partir de series de puntos, donde el número de puntos varía dependiendo de una ecuación en particular y de las opciones que elija el usuario.

Para solucionar este problema se mandaron graficar dos series, una a partir del arreglo dinámico generado por la ecuación, para la cual sólo se graficó el eje del rango y otra que contiene dos arreglos, uno con las cadenas del dominio y otro con los valores del rango, donde dicho arreglo está vacío. Dicho de otra forma, se tiene una serie que oculta los valores del dominio y otra que no tiene valores en el rango. Lo anterior funciona porque el intervalo del dominio es fijo, como se muestra en la línea de código mostrada, por lo tanto, las cadenas de la segunda serie se insertan en dichos incrementos. Por último, fue necesario

establecer el mínimo y máximo de valores que puede mostrar el dominio y asignar cadenas en dicho intervalo, lo anterior es sumamente importante, si se grafican valores que no tienen asignada una cadena en el dominio la gráfica no se muestra, este aspecto se controló con la siguiente línea de código:

```
plot.getOuterLimits().set(0, 22, -5, 5);
```

Donde los dos primeros números (0, 22) definen el dominio y los siguientes (-5, 5) el rango. Por lo tanto, el valor máximo en la cadena sería 7π:

$$7\pi = 21.99$$

El resultado de esta implementación puede verse en la Figura 3.7, puede notarse que el dominio muestra valores desde -2π hasta 5π en lugar de 0 a 7π, lo anterior se debe a que se decidió mostrar por lo menos un ciclo negativo de la señal, por ello las cadenas comienzan en -2π, pero es sólo una cuestión visual el cálculo de la función comienza desde cero, se aprovecho el período de la función (2π) para mover el origen de lugar usando:

```
plot.centerOnDomainOrigin(2*Math.PI);
    plot.centerOnRangeOrigin(0);
```

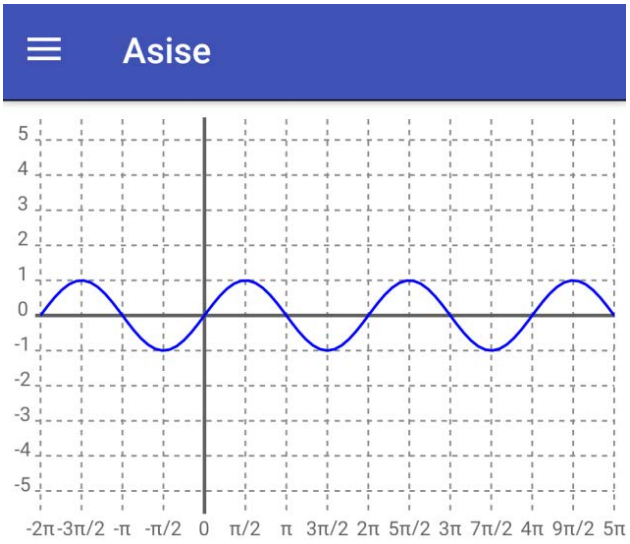


Fig. 3.7. Implementación de la librería AndroidPlot

Por último, cabe señalar que las cadenas del dominio deben estar escritas en UNICODE si se desea mostrar símbolos, por ejemplo:

```
"-3\u03c0/2"
  "\u03c0"
```

Corresponden a -3 π/2 y π respectivamente.

Processing y DDF Minim

Estas dos librerías se usaron en conjunto, de hecho, DDF Minim es una librería desarrollada para Processing. Con ellas, se elaboró un analizador de espectro en tiempo real donde la captura de datos se realiza con Java, el procesamiento con DDF Minim y el despliegue de los mismos en Processing. Considerando que DDF Minim es para Processing no fue posible enviar los datos directamente desde Java, en lugar de ello la interacción se realizó usando Processing como intermediario. Asimismo, el analizador cuenta con diversas funciones que afectan el despliegue de datos, por ejemplo, el rango de frecuencias que se puede mostrar. Para poder implementar estas características fue necesario tener una comunicación constante entre Java y Processing a través de un *hilo* (*thread*), donde dicho *hilo* maneja un canal de audio que controla el acceso al micrófono del dispositivo en modo de grabación, pero sin guardar el audio entrante en la memoria interna. Es decir, sólo se abre el canal y se permite la grabación, el sonido que ingresa se envía a Processing que lo manipula con DDF Minim y se despliega el resultado, pero el audio sólo pasa por un buffer de la RAM. Este proceso puede verse en la Figura 3.8, en dicha imagen se indica que el *hilo* también realiza el muestreo, donde la toma de muestras se realiza con los siguientes parámetros:

```
int TASA_MUESTREO = 44100;
int MODULACION_PCM = AudioFormat.ENCODING_PCM_16BIT;
int CANALES = AudioFormat.CHANNEL_IN_MONO;
```

Para poder declarar el canal de audio sólo falta calcular el tamaño del buffer, dicho tamaño se calculó con el método *getMinBufferSize()*:

```
int tamanioBuffer = AudioRecord.getMinBufferSize(TASA_MUESTREO,
                                                 CANALES,
                                                 MODULACION_PCM);
```

Entonces se tiene una tasa de muestreo de 44100 [muestras/segundo] con un PCM (Modulación por Pulsos Codificados) de 16bits, en monoaural y un buffer en bytes que cambia dependiendo del dispositivo. La declaración del canal de audio para grabación queda:

```
audioRecord = new AudioRecord(MediaRecorder.AudioSource.MIC,
                              TASA_MUESTREO,
                                 CANALES,
                              MODULACION_PCM,
                              tamanioBuffer);
```

Cabe mencionar que al modificar alguna característica desde Java es necesario detener el *hilo* y volverlo a iniciar con los cambios pertinentes, por ejemplo, el analizador cuenta con la posibilidad de mostrar u ocultar el valor de las frecuencias pico. Por lo tanto, al modificar esta opción ocurrirá el proceso ya descrito.

Por último, es posible pausar el análisis en un determinado momento, cuando esto ocurre sólo el despliegue del analizador se detiene el *hilo* continua enviando datos y se siguen

analizando pero éstos no se despliegan. Lo anterior se debe a que la pausa sobre el analizador es una propiedad de Processing y esta librería sólo recibe pero no envía nada a Java.

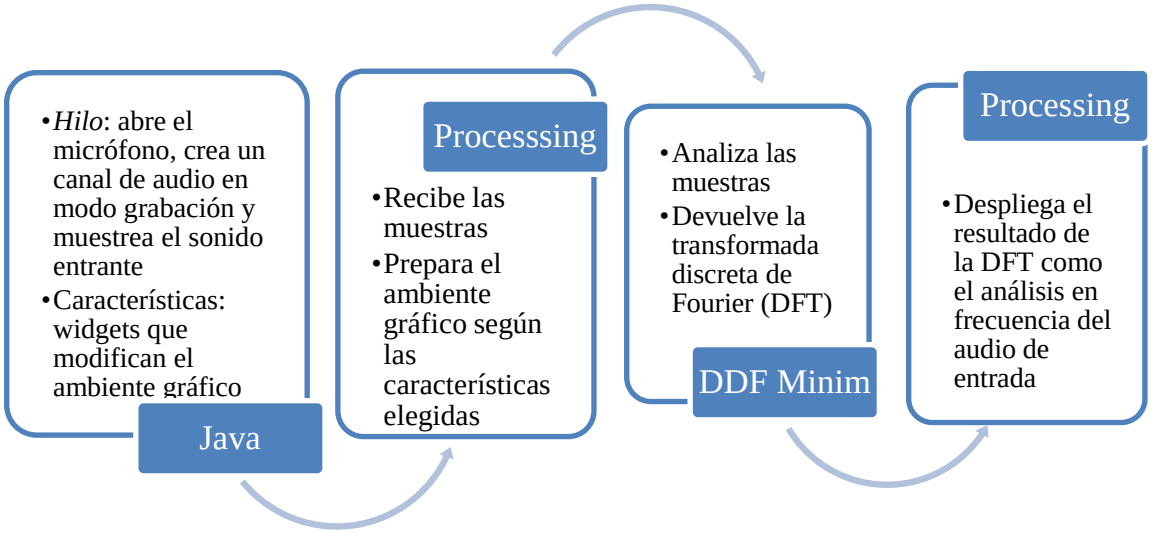


Fig. 3.8. Diagrama del funcionamiento del analizador de espectro.

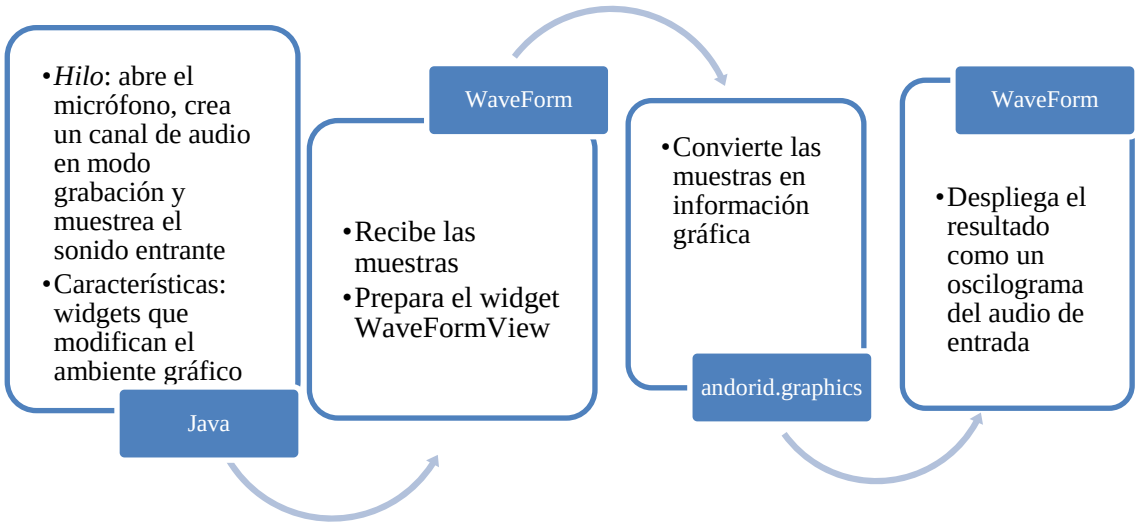
Waveform

Esta librería se usó para desarrollar un oscilograma en tiempo real. Waveform crea un widget llamado WaveFormView, al tratarse de un componente específico para Android su implementación es más sencilla que el caso anterior con Processing, el WaveFormView tiene sus atributos estáticos de Xml y métodos para Java que facilitan su uso. La única complicación con su implementación consistió en que se trata de una librería en desarrollo, según su propio autor, por lo tanto, presentaba algunos problemas de estilo. Por ejemplo, las líneas del oscilograma eran muy gruesas, este aspecto se corrigió modificando directamente la librería.

La aplicación donde se usó Waveform lleva un *hilo* similar al descrito para el analizador de espectro, este *hilo* es necesario porque el oscilograma responde a la entrada del micrófono, por lo tanto, es necesario abrirlo y controlarlo mientras se despliega la señal de audio. Asimismo, incluye características que modifican el despliegue gráfico, en este caso se trata del zoom horizontal y vertical del oscilograma.

De forma interna Waveform usa la librería android.graphics, esta librería contiene las clases Canvas, Bitmap, Picture, Paint, etc., es decir, emplea Java para convertir los bytes de las

muestras del AudioRecord en un despliegue gráfico. La implementación de esta librería se muestra en la Figura 3.9



3.9. Diagrama del funcionamiento del oscilograma.

En el caso del oscilograma no es necesario detener el *hilo* cuando se modifica una característica. Lo anterior se debe a que el zoom horizontal o vertical se maneja con android.graphics y de forma interna se puede realizar el cambio sin afectar el *hilo*. Por último, esta aplicación también se puede pausar, pero a diferencia del analizador de espectro, la pausa se realiza con un botón, es decir, un widget que control a otro widget, lo cual es común en Android.

MultiSlider, ACheckBox, Android-Segmented-Control

Estas tres librerías se usaron únicamente con fines estéticos, son tres widgets: barras deslizantes, casillas de verificación y botones, tienen una mejor presentación que los widgets propios de Android.

3.6 *Actividades* y *fragmentos*

Asise consiste en una colección de aplicaciones que pueden diferir radicalmente unas de otras, por ejemplo, procesamiento de audio y render de Latex. Por lo anterior, se decidió usar una plantilla tipo Navigation Drawer, en la Figura 3.10 se muestra esta plantilla como aparece por default al ser creada en un proyecto de Android. El Navigation Drawer tiene varias ventajas en términos de la interfaz de usuario (UI), contiene un menú listo para ser usado, pero su mayor virtud consiste en estar diseñado para usar *fragmentos* en lugar de *actividades*.

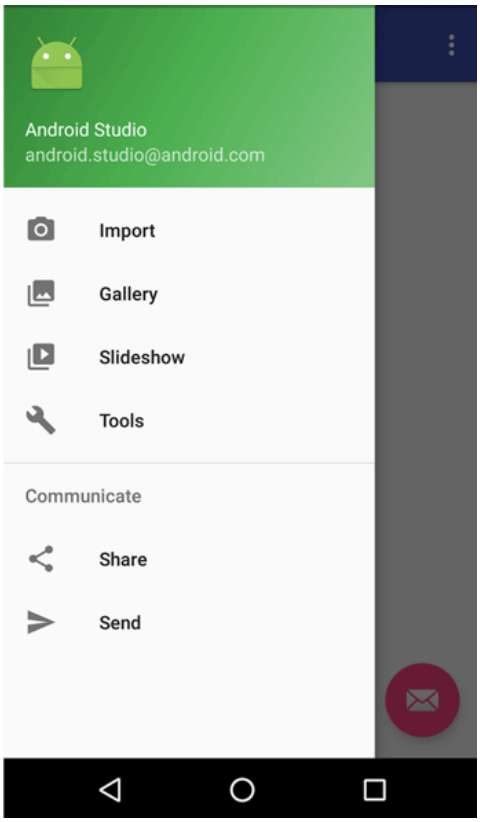


Fig. 3.10. Navigation Drawer genérico

La discusión acerca de *actividades* vs. *fragmentos* sigue siendo un tema a más de seis años de la aparición de los segundos, en general, para un programador que recién comienza le es difícil determinar cuándo usar uno y cuándo usar otro. A continuación se aborda de manera breve esta temática para explicar por qué en **Asise** se usaron *fragmentos*.

Para entender el surgimiento de los *fragmentos* es necesario recordar que existió un mundo sin tablets, cuando éstas aparecieron hubo un impacto sumamente relevante en las interfaces de usuario por un pequeño detalle:

- Los teléfonos generalmente se toman en posición vertical (*portrait*) las tablets generalmente de toman en posición horizontal (*lanscape*)

Para solucionar este problema Google introdujo los *fragmentos* en la versión Android 3.0 (API 11) en febrero de 2011. En la Figura 3.11 se muestra el esquema de la página de desarrolladores de Android que ha estado ahí desde el surgimiento de los *fragmentos*. En dicha imagen se puede observar el problema planteado, mientras en una tablet se puede tener en una interfaz de usuario dos contenidos distintos, donde el de la derecha cambia según alguna elección en la izquierda, en el teléfono son necesarias dos interfaces de usuario distintas aunque también se comuniquen. Pero, lo más importante en la Figura 3.11 es:

- Para la tablet se tienen una *actividad* y dos *fragmentos* mientras que para el teléfono son dos *actividades* y dos *fragmentos*.

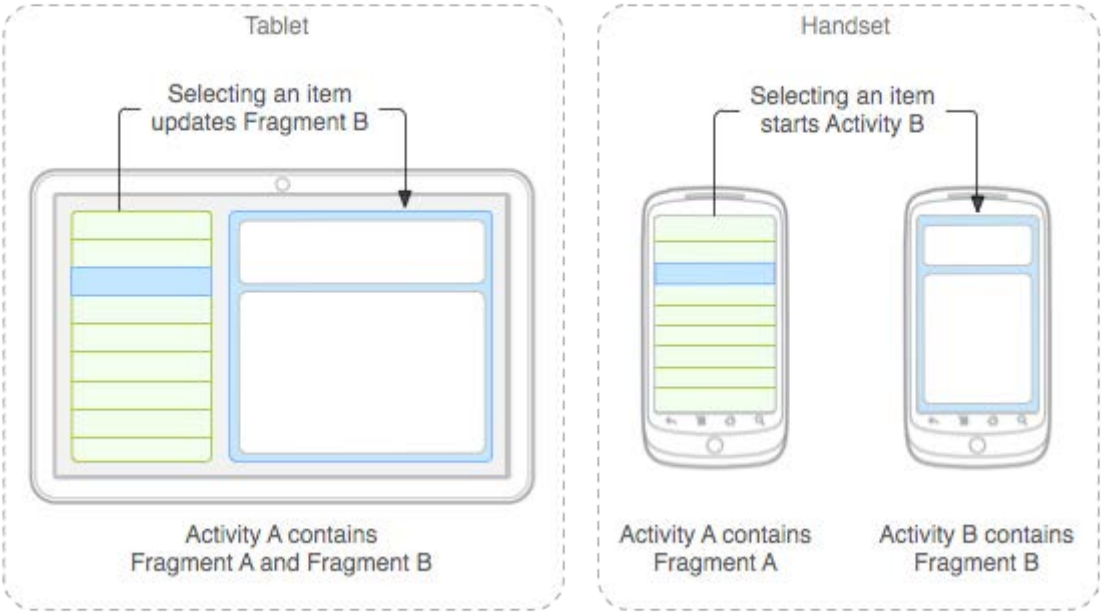


Fig. 3.11. Interfaces de usuario tablet vs. teléfono (imagen tomada de developer.android.com)

Entonces ¿Qué es un *fragmento*? Para contestar esta pregunta primero es necesario entender lo que es una *actividad*:

"Una Activity es un componente de la aplicación que contiene una pantalla con la que los usuarios pueden interactuar para realizar una acción, como marcar un número telefónico, tomar una foto, enviar un correo electrónico o ver un mapa. A cada actividad se le asigna una ventana en la que se puede dibujar su interfaz de usuario. La ventana generalmente abarca toda la pantalla, pero en ocasiones puede ser más pequeña que esta y quedar "flotando" encima de otras ventanas." (Página de Android Developers)

La definición se muestra tal y como aparece en la página de desarrolladores de Android, da una explicación intuitiva pero hay una parte técnica que no aparece en la definición. Las *actividades* son demandantes en términos de recursos, cada *actividad* tiene que estar dada de alta en el archivo *manifiesto* (*AndroidManifest.xml*) de la aplicación, en este archivo se declaran los permisos, versiones de la aplicación, orientaciones de pantalla, nombre de la aplicación, etc., es decir, cada *actividad* queda declarada a un nivel muy alto en la jerarquía del proyecto. Lo anterior se debe a que cada *actividad* va a necesitar una pantalla. Por otro lado, para lanzar una nueva *actividad* es necesario usar la clase *Intent*, es decir, emplear una descripción abstracta de una operación para *intentar* ejecutar un determinado proceso, en

este caso lanzar una *actividad*, donde cada actividad tiene una ventana establecida. Por otro lado, para los *fragmentos* de tiene:

"Un Fragment representa un comportamiento o una parte de la interfaz de usuario en una Activity. Puedes combinar múltiples fragmentos en una sola actividad para crear una IU multipanel y volver a usar un fragmento en múltiples actividades. Puedes pensar en un fragmento como una sección modular de una actividad que tiene su ciclo de vida propio, recibe sus propios eventos de entrada y que puedes agregar o quitar mientras la actividad se esté ejecutando (algo así como una "subactividad" que puedes volver a usar en diferentes actividades)." (Página de Android Developers)

En este caso la definición de Android Developers es más fácil de entender, un *fragmento* es una parte de la pantalla que puede ser reutilizada, por ejemplo, si se tiene una serie de botones que son comunes en dos o más *actividades*, en lugar de declarar dichos botones en cada *actividad*, se crea un *fragmento* con ellos y dicho *fragmento* se inserta en las *actividades* que lo requieran, donde dicho *fragmento* puede parecer en una actividad en la parte inferior y en otra a un costado.

Los fragmentos no se declaran en el *manifiesto*, tampoco es necesario llamarlos usando un *Intent*, en su lugar se usa la clase *FragmentManger* cuya operación es más sencilla. Como se mencionó previamente con el *Intent* se está buscando ejecutar un proceso nuevo, con el *FragmentManger* se realiza una transacción, donde dicha transacción generalmente reemplaza el contenido de la interfaz de usuario.

En el caso de **Asise** se usó un Navigation Drawer y *fragmentos*, de hecho sólo se tiene la *actividad* principal (*MainActivity*). La interfaz de usuario de esta actividad se reemplaza con *fragments* dependiendo del contenido que se quiera desplegar. Lo anterior genera que la transición entre pantallas sea lo más eficiente posible, por supuesto, existen interfaces de usuario que tardan más en desplegar su contenido que otras, este aspecto está estrechamente ligado al render de Latex, los *fragmentos* con mayor contenido web que requieren mostrar ecuaciones tardan más en mostrarse completamente que aquellos donde sólo hay contenido gráfico. Sin embargo, una vez en ejecución éstos últimos son mas demandantes de recursos dado que son dinámicos e interactivos.

3.7 Permisos

A pesar de que **Asise** contiene distintas aplicaciones, los permisos necesarios para su funcionamiento sólo son dos: Internet y grabación de audio. El primero se necesita porque **Asise** incluye un buscador de libros que se conecta con la página web de la Facultad de Ingeniería de la UNAM. La grabación de audio se solicita porque es necesaria para el funcionamiento del analizador de espectro y el oscilograma, como se mencionó previamente en este capítulo. Los permisos se declararon en el *manifiesto* del proyecto:

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
  <uses-permission android:name="android.permission.INTERNET" />
```

Por motivos de seguridad, a partir de Android 6.0 (Marshmallow) Google cambió la forma de manejar los permisos en las aplicaciones y los categorizó como normales o riesgosos. Los primeros se siguen manejando sólo con el manifiesto, pero con los riesgosos es necesario declarar en la *actividad* principal estos permisos y manejar su aprobación. De los dos permisos necesarios en **Asise** la grabación de audio está catalogada como riesgosa, por ello, se declaró este permiso siguiendo la normativa de Android Developers:

```
ActivityCompat.requestPermissions(this, permissions,
REQUEST_RECORD_AUDIO_PERMISSION);
```

Primero se solicita el permiso a través de *ActivityCompat*, donde el método *requestPermissions* llama el siguiente código:

```
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    switch (requestCode){
        case REQUEST_RECORD_AUDIO_PERMISSION:
            permissionToRecordAccepted  = grantResults[0] ==
PackageManager.PERMISSION_GRANTED;
            break;
    }
    if (!permissionToRecordAccepted ){
        AlertDialog.Builder builder1 = new AlertDialog.Builder(MainActivity.this);
        builder1.setMessage("La grabación de audio es necesaria para el funcionamiento de
Asise. La aplicación se cerrará");
        builder1.setCancelable(true);
        builder1.setPositiveButton(
                "OK",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        dialog.cancel();
                        finish();
                    }
                });
        AlertDialog alert11 = builder1.create();
        alert11.show();
    }
}
```

El código original de Android Developers se modificó para incluir un *AlertDialog* que le avisa al usuario sobre la necesidad del permiso para usar el micrófono y el cierre de la aplicación.

3.8 Requerimientos de los dispositivos

La versión mínima de Android que acepta **Asise** es JellyBean 4.1 (API 16), no existen limitaciones para el tamaño de pantalla y puede instalarse en tablets, teléfonos y phablets,

en este sentido se optimizó el tamaño de letra de los botones, campos de texto y demás widgets implementando en los *fragmentos* el siguiente método:

```
private void pantalla() {
    int pantallaTamanio = getResources().getConfiguration().screenLayout
&Configuration.SCREENLAYOUT_SIZE_MASK;
    switch(pantallaTamanio) {
        case Configuration.SCREENLAYOUT_SIZE_XLARGE:
            letraGrafica = 16;
            letraTextView=22;
            letraWeb=22;
            break;
        case Configuration.SCREENLAYOUT_SIZE_LARGE:
            letraGrafica = 15;
            letraTextView=20;
            letraWeb=20;
            break;
        case Configuration.SCREENLAYOUT_SIZE_NORMAL:
            letraGrafica = 11;
            letraTextView=15;
            letraWeb=15;
            break;
        case Configuration.SCREENLAYOUT_SIZE_SMALL:
            letraGrafica = 9;
            letraTextView=14;
            letraWeb=14;
            break;
        case Configuration.SCREENLAYOUT_SIZE_UNDEFINED:
            letraGrafica = 11;
            letraTextView=15;
            letraWeb=15;
            break;
        default:
            break;
    }
}
```

En el código mostrado se crea una variable a partir del tamaño de pantalla y con dicha variable se asignan distintos tamaños de letra para los elementos de la interfaz de usuario.

4 Manual de usuario de Asise

4.1 Estructura de Asise

En el presente capítulo se describe con detalle las características de **Asise** y se muestra el uso de cada uno de sus componentes. La aplicación está dividia en seis categorías principales:

1. Aplicaciones
2. Conceptos
3. Ejercicios (Series Fourier)
4. Tablas
5. Bibliografía
6. Recursos web

Estás categorías se presentan como un menú al iniciar la aplicación (Figura 4.1).

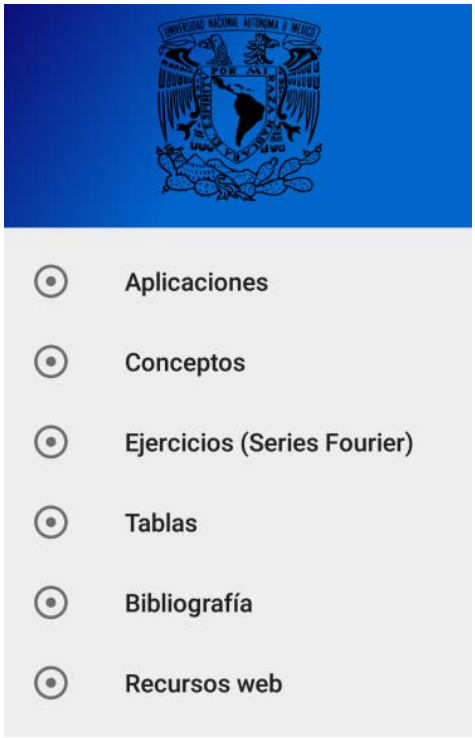


Fig.4.1 Pantalla de inicio

Las categorías *Aplicaciones*, *Conceptos*, *Ejercicios (Series Fourier)* y *Tablas* dan paso, cada una, a una lista seleccionable que contiene distintas subcategorías. Donde cada subcategoría abre una nueva pantalla donde se despliega el contenido correspondiente.

En *Aplicaciones* se tiene: *Partes de una señal*, *Series de Fourier*, *Analizador de espectro*, *Muestreo* y *Sumar señales de audio*.

Conceptos da pie a: *Señales en Ingeniería Eléctrica Electrónica*, *SLIT*, *Tiempo discreto y teorema del muestreo*, *Número de Euler*, *Transformadas de forma intuitiva*, *Variables complejas en las transformadas 1 (construcción matemática)* y *Variables complejas en las transformadas 2 (interpretación física).*

Para *Ejercicios (Series Fourier)* están: *Cuadrada*, *Cuadrada rectificada*, *Triangular*, *Diente de sierra*, *Onda completa rectificada*, *Media onda rectificada* y *Pulso rectangular*.

Por último, la categoría *Tablas* contiene: *Identidades trigonométricas*, *Laplace (transformada)*, *z (transformada)*, *Fourier (transformada)* y *Fourier (series).*

Cabe mencionar que la sub categoría *Series de Fourier* de la categoría *Aplicaciones* genera una nueva lista seleccionable cuyo contenido es idéntico al de la categoría *Ejercicios (Series Fourier)*, lo anterior se debe a que el contenido de los ejercicios está reforzado con el contenido de la aplicación específica para este tema.

Las otras dos categorías (*Bibliografía* y *Recursos web*) muestran listas del contenido al cual hacen referencia. En el caso de *Bibliografía* se incluye un buscador de libros que se conecta a la base de datos de las bibliotecas de la Facultad de Ingeniería, UNAM.

En la Figura 4.2 se muestra un esquema de la estructura (categorías y subcategorías) de **Asise**.

4.2 Categoría *Aplicaciones*

Al ingresar en la categoría *Aplicaciones* se muestran sus respectivas subcategorías (Figura 4.3). Como su nombre lo indica, *Aplicaciones* hace referencia al contenido dinámico de **Asise** diseñado para mostrar el análisis de sistemas y señales de forma práctica e interactiva.

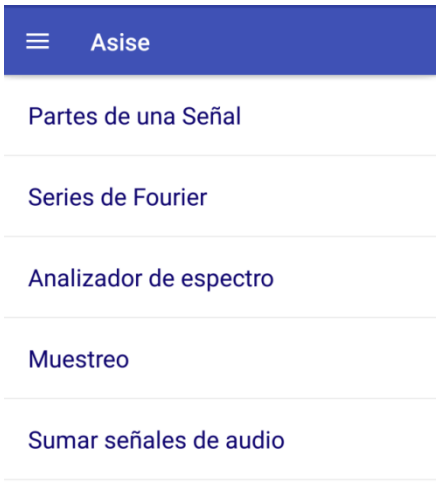


Fig.4.3 Menú de la categoría *Aplicaciones*

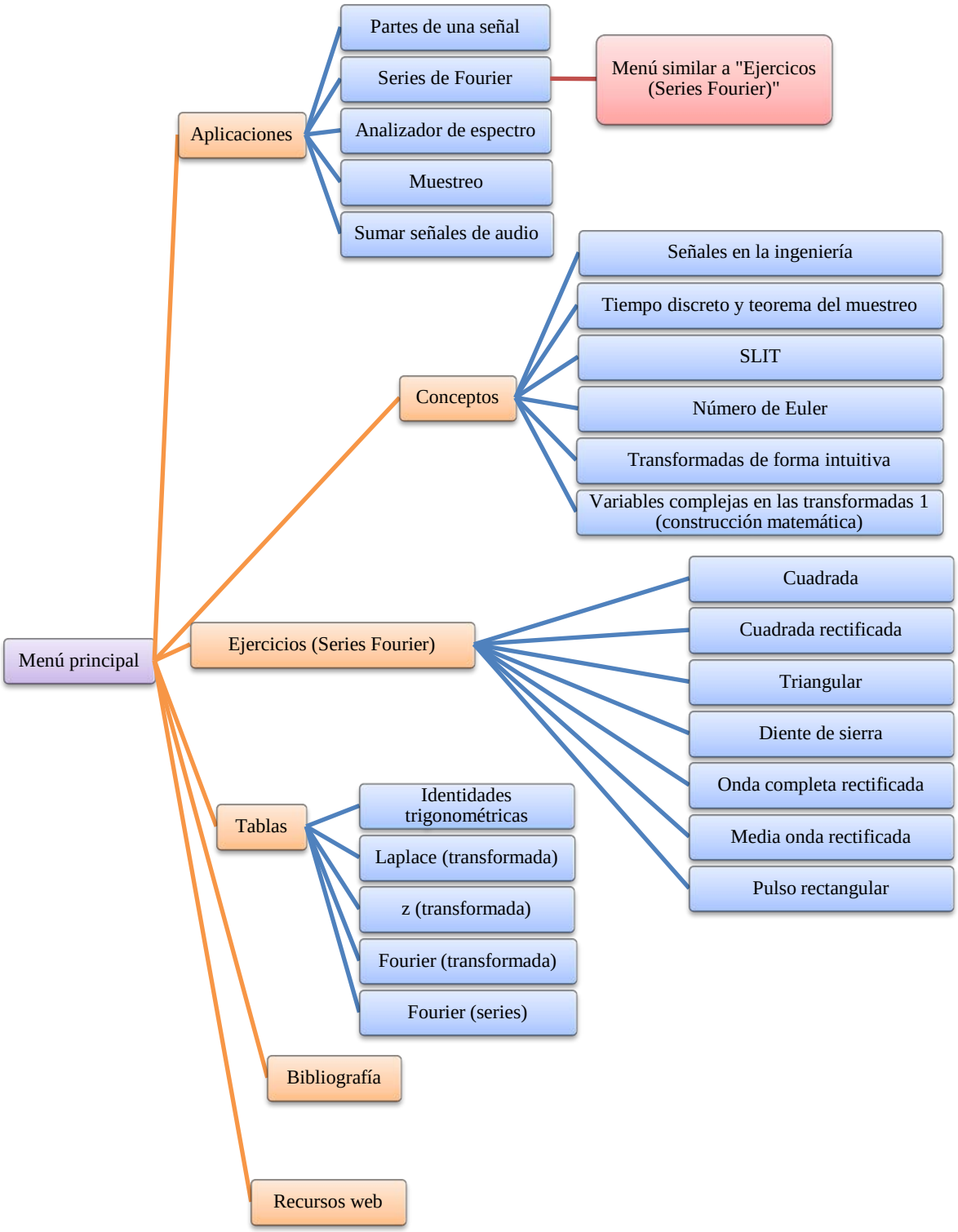


Fig. 4.2 Estructura de **Asise**.

4.2.1 *Partes de una señal*

La primer subcategoría es *Partes de una señal* (Figura 4.4). Al ingresar se observan los siguientes elementos:

1. Despliegue gráfico
2. Amp (Amplitud)
3. Frec (Frecuencia)
4. Fase
5. Comparar con la señal original

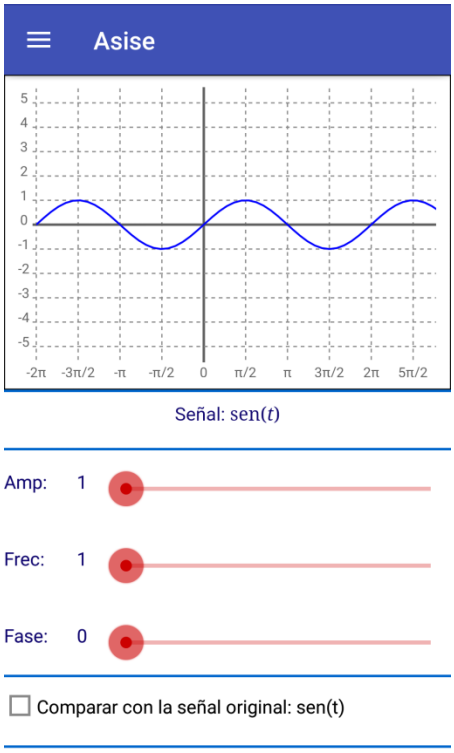


Fig.4.4 Contenido de la subcategoría *Partes de una señal*

Despliegue gráfico

Es un área para graficar señales senoidales. Al iniciar la aplicación se muestra la función sen(*t*) que es la señal por default. El eje horizontal de la gráfica va desde -2π hasta 5π, el eje vertical tiene un rango de -5 a 5. La gráfica es deslizable hacia la izquierda y la derecha y tiene zoom horizontal.

Amplitud

Es una barra deslizable que modifica la amplitud de la señal que se ve en el despliegue gráfico. La escala que se puede manejar es de 1 a 5 unidades, con una resolución de una unidad.

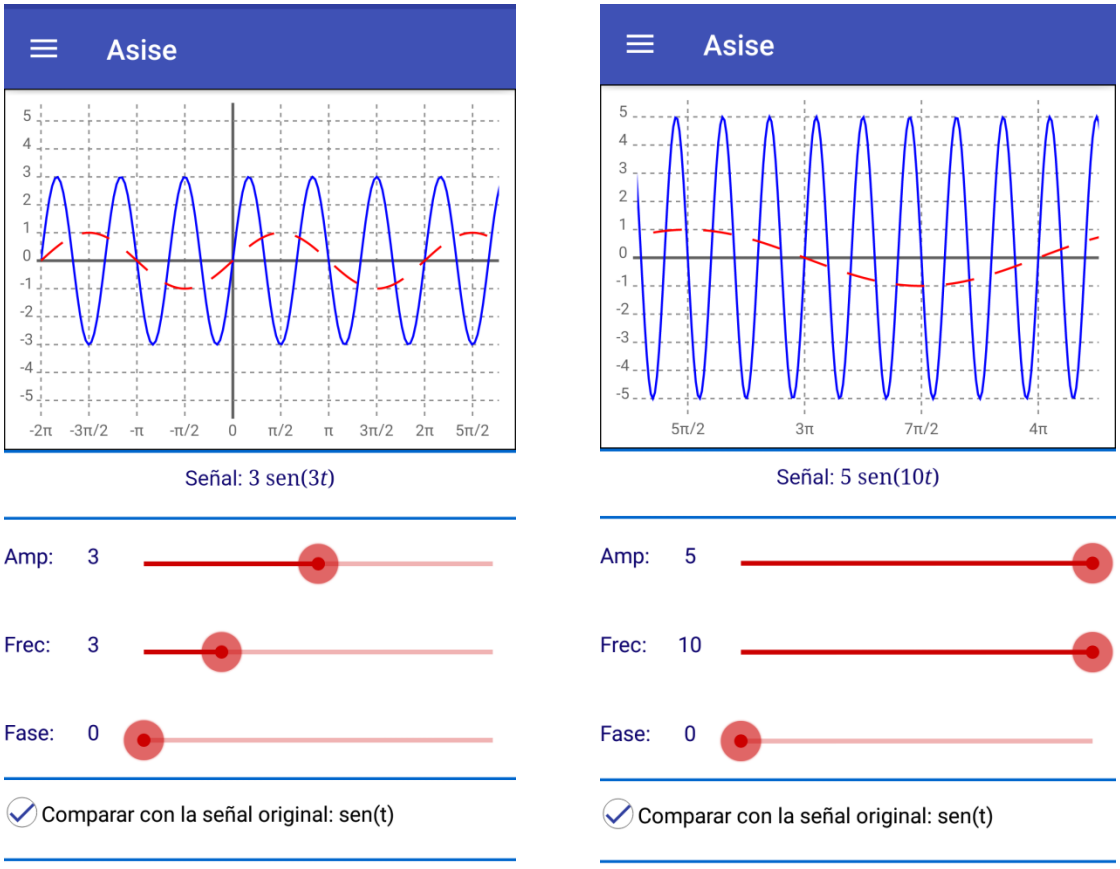
Frecuencia

Es una barra deslizable que modifica la frecuencia de la señal que se ve en el despliegue gráfico. La escala que se puede manejar es de 1 a 10 unidades, con una resolución de una unidad.

Fase

Es una barra deslizable que modifica la fase de la señal que se ve en el despliegue gráfico. La escala que se puede manejar es de 0 a 5π unidades, con una resolución de π/2.

A continuación se muestran dos ejemplos de gráficas generadas con esta aplicación (Figura 4.5).



(a) (b)

Fig.4.5 Ejemplos de la aplicación *Partes de una señal* .Izquierda señal por default y la resultante de las barras deslizantes. Derecha gráfica con zoom horizontal y deslizada hacia la izquierda.

4.2.2 *Series de Fourier* (Aplicaciones)

Esta subcategoría abarca un grupo de aplicaciones con el mismo fin, todas están orientadas a graficar de forma dinámica distintas señales en series trigonométricas de Fourier. En la Figura 4.6 se observa el menú correspondiente a este grupo de aplicaciones.

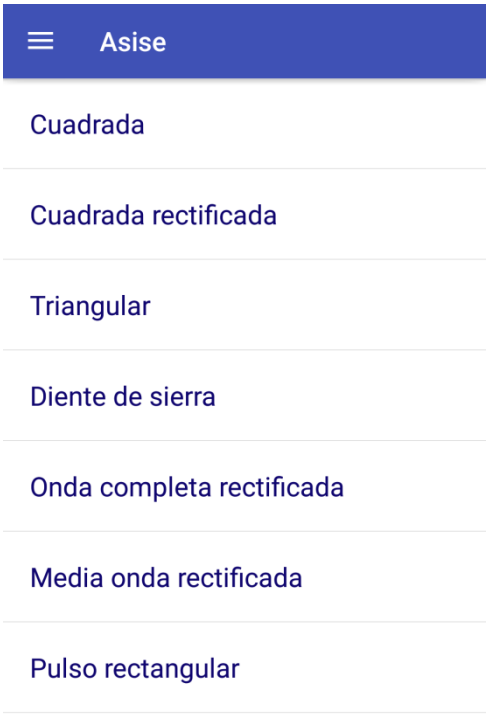


Fig. 4.6 Menú de la subcategoría *Series de Fourier*

Como puede notarse en la Figura 4.6 se incluyen siete señales distintas. A continuación se muestra sólo una de ellas (Diente de Sierra) ya que la operación de todas es similar. Los elementos que tienen este grupo de aplicaciones son (Figura 4.7):

1. Despliegue gráfico
2. Escala
3. Términos
4. Función y serie
5. Expansión de la serie
6. Cálculo de términos
7. Ejercicios y tablas

Despliegue gráfico

Es un área donde se muestra una gráfica resultado de la suma de *n* términos de la serie trigonométrica de Fourier correspondiente. La gráfica tiene una escala desde -2 π hasta 5 π. La gráfica es deslizable hacia la izquierda y la derecha y tiene zoom horizontal.

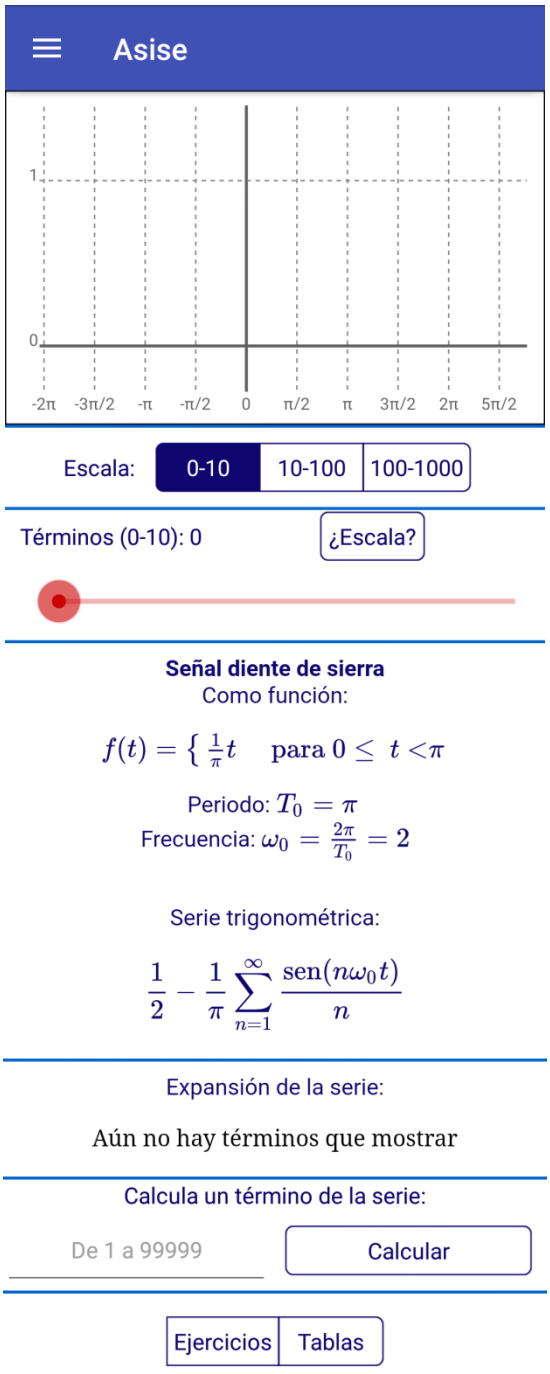


Fig. 4.7 Contenido de *Series de Fourier* (Aplicaciones)

Escala

Son tres botones que permiten elegir entre graficar de 0 a 10, de 10 a 100 o de 100 a 1000 términos de la serie trigonométrica de Fourier correspondiente. Se incluye un botón de ayuda donde se explica esta funcionalidad.

Términos

Es una barra deslizable con la que es posible realizar un barrido de *n* términos de la serie, donde el valor de *n* está delimitado por los botones de la escala. Cuando la escala es de 0 a 10 el barrido se puede realizar para *n*=1, es decir 0, 1, 2, 3,…, 10,. En la escala 10 a 100 *n*=10, por lo tanto, es posible graficar 10, 20, 30,…,100 términos . Por último, en la escala 100 a 1000 *n*=100, el barrido puede realizarse cada 100, 200, 300,…, 1000 términos.

Función y serie

Es un área de texto donde se define la señal que se va graficar, donde dicha definición se realiza a partir de una función a trozos, se muestra el período y la frecuencia de esta función. También se puede observar la serie trigonométrica de Fourier correspondiente en forma de sumatoria.

Expansión de la Serie

Es un área de texto en la cual se muestra la serie trigonométrica de Fourier correspondiente de forma expandida. Al ingresar por primera a la aplicación vez se ve la leyenda "*Aún no hay términos que mostrar*" esto se debe a que la gráfica no contiene información alguna. Para la escala de 0 a 10 se muestran todos los términos implicados (Figura 4.8a), para las escalas restantes sólo se muestran los primeros dos términos y el último (Figura 4.8 b).

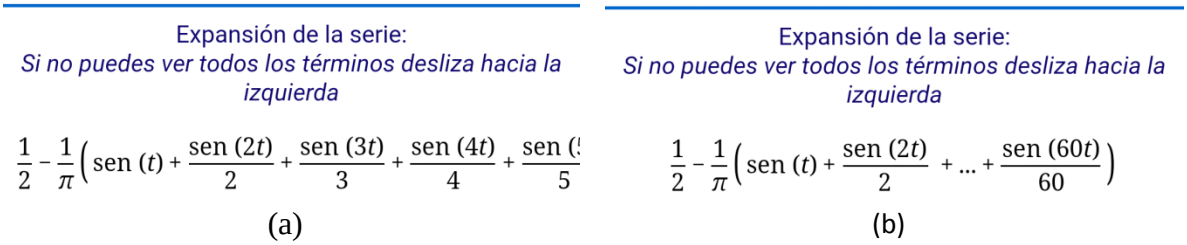


Fig. 4.8 Expansiones de la serie trigonométrica de Fourier de la señal diente de sierra. (a) para n=8 términos. (b) n=60 términos.

En la Figura 4.8 puede notarse que aparece la leyenda "*Si no puedes ver todos los términos desliza hacia la izquierda*". Esta funcionalidad es necesaria dada la diversidad en los tamaños de pantalla presente entre distintos dispositivos Android. De esta manera, al deslizar hacia la izquierda, el contenido mostrado en la Figura 4.8 (a) cambia por el mostrado en la Figura 4.9. Cabe mencionar que el deslizamiento se debe hacer sobre el área de texto destinada para la expansión de la serie, ya que es un espacio de la pantalla que se mueve de forma independiente.

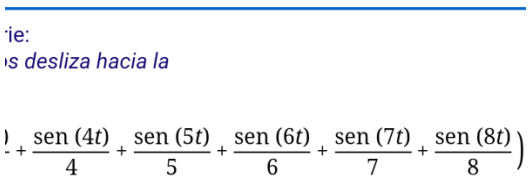


Fig. 4.9 Expansión de la serie trigonométrica de Fourier de la señal diente de sierra para *n*=8 términos deslizado hacia la izquierda.

Cálculo de Términos

En este campo es posible obtener de forma individual el valor de un término en particular de la serie trigonométrica de Fourier. Para las señales cuadrada, cuadrada rectificada, triangular, diente de sierra, y pulso rectangular es posible calcular para *n*=1 y hasta *n*=99999. Para las series onda completa rectificada y media onda rectificada sólo se puede hasta *n*=9999. Lo anterior se debe a que en estas series se tiene dentro de la sumatoria un denominador donde *n* está elevado al cuadrado, por lo tanto, se vuelve impráctico mostrarlo. Para usar esta calculadora primero es necesario ingresar el término que se desea obtener y después presionar el botón *Calcular*, el resultado se mostrará y el campo de texto donde se ingresó el término se limpiará. En la Figura 4.10 se muestra el resultado para *n*=13167 de la serie diente de sierra.

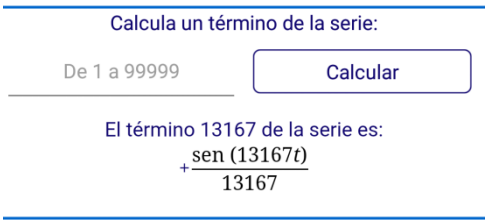


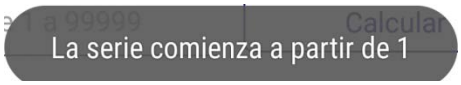
Fig. 4.10 Cálculo del término 13767 de la serie trigonométrica de Fourier para la señal diente de sierra.

Este cálculo cuenta con dos excepciones:

1. Cuando no escribe nada y se presiona *Calcular*
2. Cuando se escribe cero y se presiona *Calcular*

En el primer caso se muestra el siguiente mensaje:

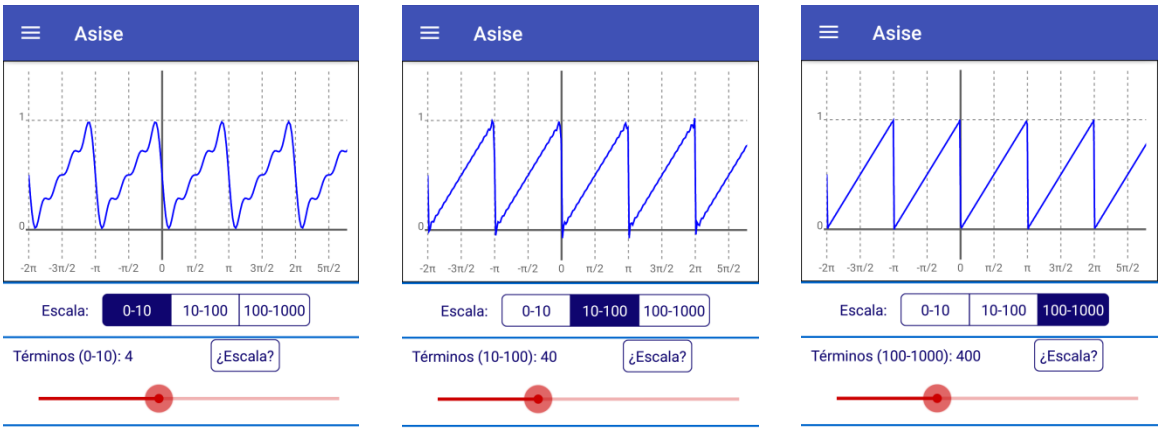
En el segundo caso se muestra:



Ejercicios y Tablas

Consiste en dos botones que llevan directamente a otras categorías de **Asise** vinculadas directamente con éste tema.

Por último, en la Figura 4.11 se muestra un ejemplo funcional de la aplicación, son tres gráficas de la señal de diente de sierra para *n*=4, *n*=40 y *n*=400 términos de la serie.



(a) (b) (c)

Fig. 4.11 Gráficas de la señal diente de sierra. (a) *n*=4. (b) *n*=40. (c) *n*=400

4.2.3 *Analizador de espectro*

Esta subcategoría (Figura 4.12) se compone de los siguientes elementos:

1. Despliegue gráfico
2. Ayuda
3. Rango del espectro
4. Ajuste vertical
5. Picos
6. Generador de tonos

Despliegue Gráfico

Es un área donde se muestra el análisis en frecuencia de una señal de entrada, en su parte inferior se observa el rango de frecuencias que puede observarse en la gráfica. El analizador comienza a funcionar al momento de ingresar a esta aplicación, esto puede verificarse hablando directamente hacia el dispositivo o reproduciendo alguna canción, es posible que el espectro sea incipiente y que sea necesario realizar el ajuste vertical, este aspecto se explica más adelante en el texto. El espectro tiene como entrada el micrófono del dispositivo, de esta forma para un mismo sonido ambiental es posible que la intensidad se muestre de forma distinta si se aproximan dos dispositivos diferentes. Lo anterior se debe a la calidad del hardware que varía notablemente entre distintas marcas.

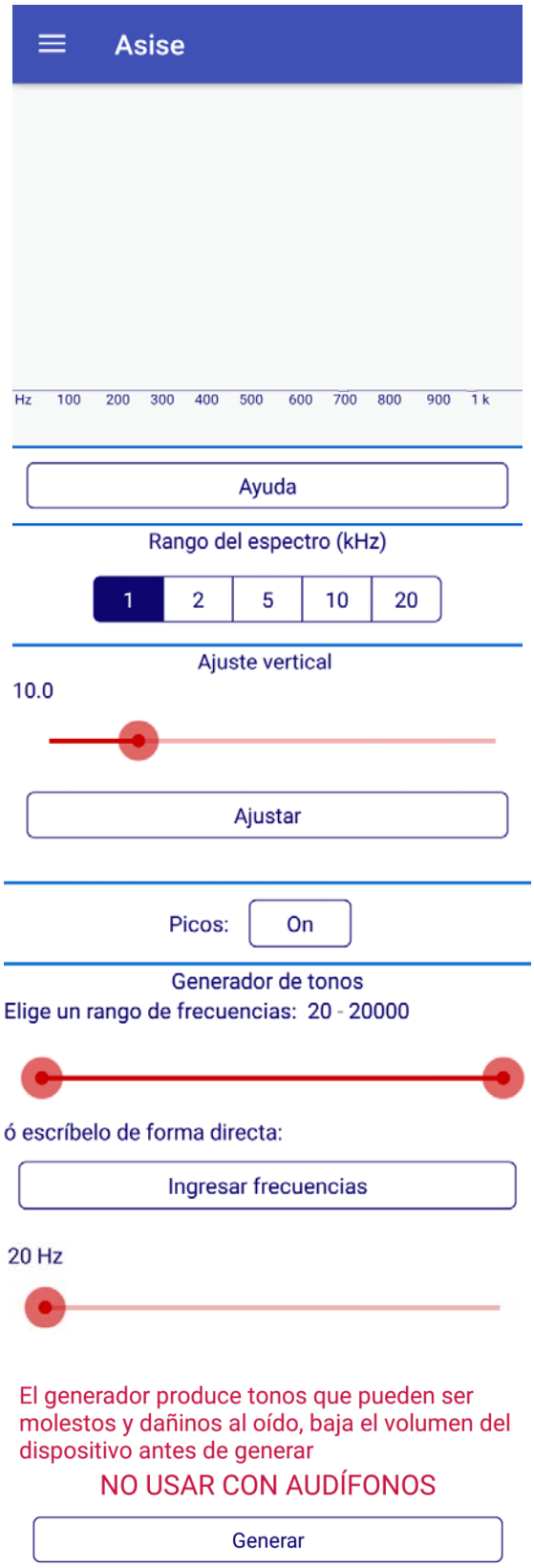


Fig. 4.12 Contenido del *Analizador de espectro*

Ayuda

Es un botón que despliega un diálogo de alerta en el cual se explican las características y el funcionamiento del analizador de espectro, al final de dicho diálogo se incluye un botón para cerrarlo y regresar al analizador. Cabe mencionar, que el diálogo de alerta utilizado no es el estándar que se utiliza en el sistema Android dado que este widget se modificó para usar programación híbrida y lograr incluir en él texto con formato html. Este aspecto se describe con mayor detalle en el Capítulo **Aspectos Técnicos**.

Rango del Espectro

Son cinco botones mutuamente excluyentes que contienen los siguientes números: 1, 2, 5, 10 y 20. Estos números hacen referencia a la escala máxima que se muestra en el despliegue gráfico, es decir, cuando se tiene seleccionado el botón '1' el rango máximo es 1 [kHz], el botón '2' cambia el rango hasta 2 [kHz], este patrón continúa hasta los 20 kHz.

Ajuste Vertical

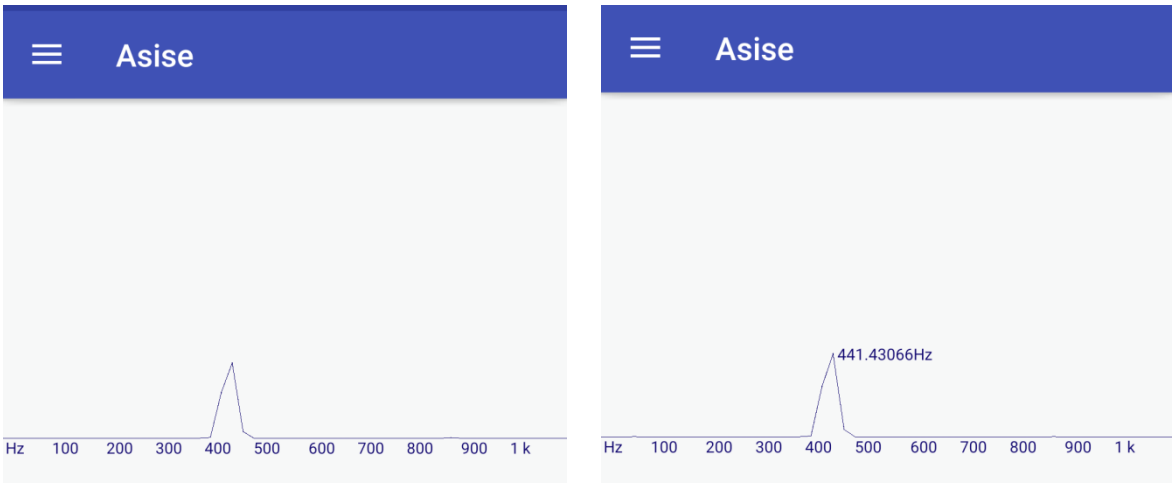
Es una barra deslizable cuya escala va desde 0 hasta 200 en intervalos de 4 unidades. El ajuste vertical magnifica el espectro en el despliegue gráfico. Para realizar el ajuste es necesario seleccionar primero un valor en la barra deslizable y posteriormente presionar el botón *Ajustar*. Esta funcionalidad es necesaria debido a que el sistema Android está presente en dispositivos provenientes de muchas marcas distintas, por lo tanto, la calidad de sus componentes electrónicos cambia considerablemente. Por ello, para una misma señal de entrada el despliegue gráfico puede variar notablemente entre distintos dispositivos.

Picos

Es un botón que activa o desactiva el despliegue de los valores de las frecuencias pico (en Hertz) en el analizador de espectro. En la Figura 4.13 se muestra el despliegue gráfico para una misma señal donde en (a) no se muestra el valor pico y en (b) sí se observa.

Generador de Tonos

Son dos barras deslizables y dos botones. La primer barra deslizable tiene dos punteros, uno en cada extremo, y sirve para determinar las frecuencias mínima y máxima de las cuales depende la segunda barra deslizable, con ésta segunda barra deslizable es posible realizar un barrido de frecuencias y observar dicho barrido en el despliegue gráfico, donde los valores extremos del barrido se adquieren de la primer barra deslizable. El botón *Ingresar frecuencias* tiene la misma función que la barra deslizable uno, pero con el botón se puede determinar de manera más exacta los rangos mínimo y máximo, ya que dicho botón despliega un diálogo de alerta donde las frecuencias se ingresan de forma manual (Figura 4.14). Por último el botón *generar* sirve para reproducir la frecuencia ingresada.



(a) (b)

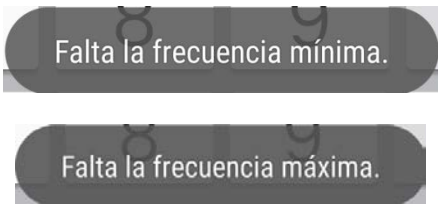
Fig. 4.13 Analizador de espectro. (a) Sin mostrar frecuencias pico, (b) Con frecuencias pico

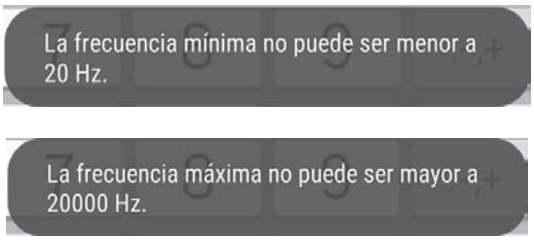
Fig. 4.14 Diálogo de alerta para ingresar las frecuencias del generador de tonos

El diálogo de alerta para ingresar las frecuencias mínima y máxima del generador de tonos tiene las siguientes excepciones:

1. No admite cadenas vacías
2. La frecuencia mínima no puede menor a 20 [Hz]
3. La frecuencia máxima no puede ser mayor a 20000 [Hz]

Las excepciones mostradas lanzan los siguientes mensajes:





Pausar el *Analizador de espectro*

Para poner pausa en el análisis sólo es necesario hacer presionar sobre el despliegue gráfico, para reanudar hay que volver a presionar en dicha área.

INFORMACIÓN IMPORTANTE

El generador produce tonos puros que pueden ser molestos y dañinos al oído humano, se recomienda bajar el volumen del dispositivo antes de usar el generador.

NO USAR CON AUDÍFONOS

4.2.4 *Muestreo*

Muestreo (Figura 4.15) tiene los siguientes elementos:

Fig. 4.15 Contenido de *Muestreo*

1. Señal de entrada
2. Despliegue gráfico de la señal muestreada
3. Señal de salida
4. Despliegue gráfico resultado del muestreo
5. Tasa de muestreo
6. Frecuencia de la señal de salida

Señal de entrada

Es un área de texto donde se muestra cual es la señal de entrada. Dicha señal siempre es cos(*t*), por lo tanto, su frecuencia siempre es de 1 [Hz].

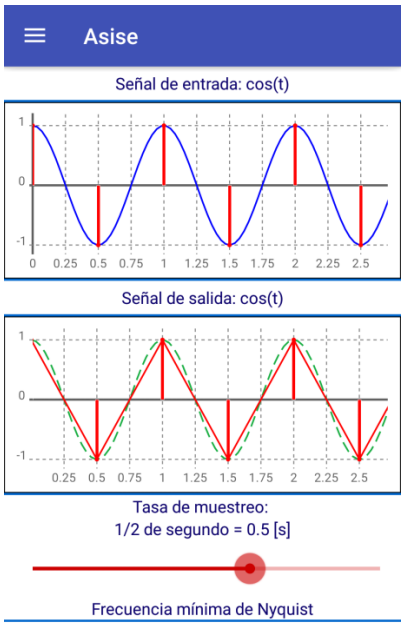


Fig. 4.15 Contenido de *Muestreo*

Despliegue gráfico de la señal de entrada

Es un área donde se muestra la señal de entrada de forma gráfica, se incluyen barras verticales para indicar los puntos donde se están tomando las muestras de dicha señal. La escala de la gráfica va desde 0 hasta 14π. La gráfica es deslizable hacia la derecha e izquierda y tiene zoom horizontal.

Señal de salida

Es un área de texto donde se muestra cual es la señal de salida. Dicha señal puede ser: cos (*t*), cos (t/2), cos (t/3), cos (t/4), cos (t/9) y cos(0t) estos valores dependen de la tasa de muestreo.

Despliegue gráfico de la señal de salida

Es un área donde se ve la señal de salida de forma gráfica. Esta gráfica se muestra de dos formas distintas. Primero, como la unión de los distintos puntos de muestreo, lo que genera una gráfica con curvas rígidas de color rojo. Segundo, como la unión de los distintos puntos de muestreo, pero donde dicha unión es una curva suave discontinua de color verde. Esta gráfica tiene la misma escala y atributos para el desplazamiento y el zoom que la gráfica de la señal de entrada.

Tasa de muestreo

Es una barra deslizable con la cual se modifica la tasa de muestreo que se aplica a la señal de entrada, dicha tasa va desde 1 [s] (1[Hz]) hasta 1/16 [s] (16 [Hz]) pasando por 1/2 [s] (2[Hz]) que corresponde a la frecuencia de muestreo mínima que cumple con el teorema de Nyquist-Shannon. Debajo de la barra se incluye un campo de texto que despliega información concerniente al proceso de muestreo, se muestran tres posibles mensajes: "*Aliasing*", "*Frecuencia mínima de Nyquist*" y "*No hay pérdida de información*". Al comenzar la aplicación se muestran las gráficas y campos de texto correspondientes a una frecuencia de muestreo de 2 [HZ].

Frecuencia de la señal de salida

Es una área de texto donde se muestra el cálculo para obtener la frecuencia de la señal de salida. El contenido de este campo es dinámico, cambia de acuerdo con la tasa de muestreo seleccionada.

4.2.5 *Sumar señales de audio*

Sumar señales de audio (Figura 4.16), sus elementos son:

1. Despliegue gráfico
2. Mostrar

3. Generar
4. Ayuda

Despliegue gráfico

Es un oscilograma en el cual se observa una señal de entrada en función del tiempo. Para que este oscilograma comience a funcionar es necesario presionar el botón mostrar. La señal de entrada para el oscilograma es el micrófono de dispositivo.

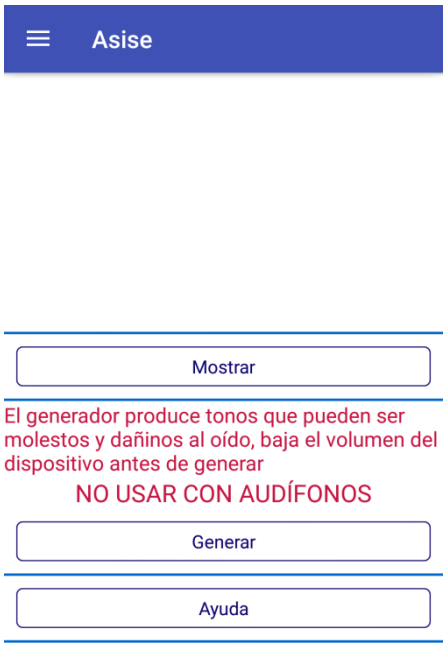


Fig. 4.16 Contenido de *Sumar señales de audio*

Mostrar

En un botón que la ser presionado genera tres cambios: primero, se activa el oscilograma, segundo, el botón *Mostrar* cambia de nombre a *Pausa* y tercero, se despliegan dos barras deslizables, una para realizar zoom horizontal sobre el oscilograma y otra para el zoom vertical (Figura 4.17). Obviamente, al cambiar el nombre a *Pausa* su función se invierte en todo sentido, es decir al presionarlo desaparecen las barras deslizables, se detiene el oscilograma pero conserva el último oscilograma mostrado y cambia nuevamente su nombre *Mostrar*.

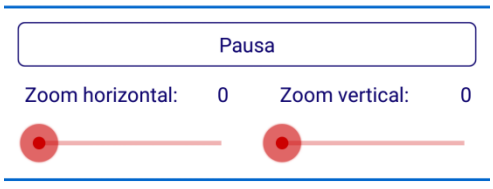


Fig. 4.17 Cambios que genera el botón *Mostrar*

Generar

Es un botón que despliega un diálogo de alerta (Figura 4.18) en el cual se tienen que ingresar dos parámetros: una frecuencia entre 20 Hz y 20 kHz y el número de armónicos que acompañarán a dicha frecuencia, este número puede ir de 1 a 10. Cabe mencionar que en este caso es posible ingresar frecuencias con números decimales de hasta dos posiciones. Al presionar el botón *Ingresar* del diálogo de alerta se regresa a la pantalla principal donde se observarán los siguientes cambios: primero, el botón *Generar* cambia de nombre por *Detener*, segundo, se generan dos áreas de texto por cada armónico, una para mostrar la frecuencia y otra para mostrar la amplitud y tercero, se crea una barra deslizable por cada armónico donde dicha barra modifica la amplitud del armónico correspondiente, la escala de todas las barras es de 0 a 100 unidades (Figura 4.19). Cabe mencionar que el diálogo de alerta es capaz de detectar armónicos que no son posibles de generar, por ejemplo, para una frecuencia de 5 [kHz] sólo es posible generar hasta el cuarto armónico que corresponde a 20 [kHz], el quinto armónico ya se saldría de la escala auditiva del ser humano. Por último, al presionar el botón *Detener* se cancelan todos estos cambios.

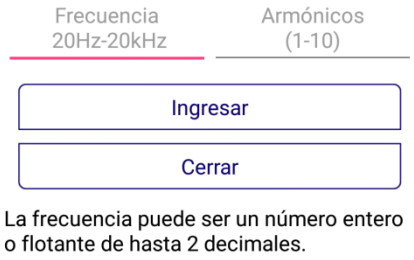


Fig. 4.18 Diálogo de alerta del botón *Generar*

Las amplitudes de la fundamental y los armónicos son independientes. De esta forma, es posible cambiar el resultado de la suma de señales modificando este aspecto, si se quiere anular un armónico sólo es necesario darle una amplitud de cero.

El generador y el oscilograma son independientes, por lo tanto, para observar la suma de las señales es necesario presionar el botón *Mostrar*.

El diálogo de alerta tiene las siguientes excepciones:

1. No se admite la falta de la frecuencia fundamental
2. No se admite la falta del número de armónicos
3. La frecuencia fundamental no puede ser menor a 20 [Hz]
4. La frecuencia fundamental no puede ser mayor a 20000 [Hz]
5. No admite que los armónicos sean mayores a 20000[HZ]

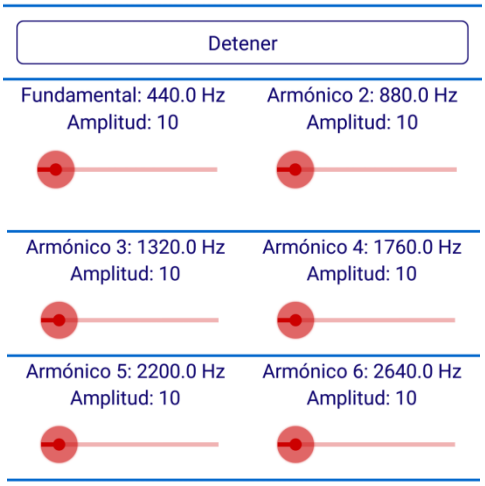
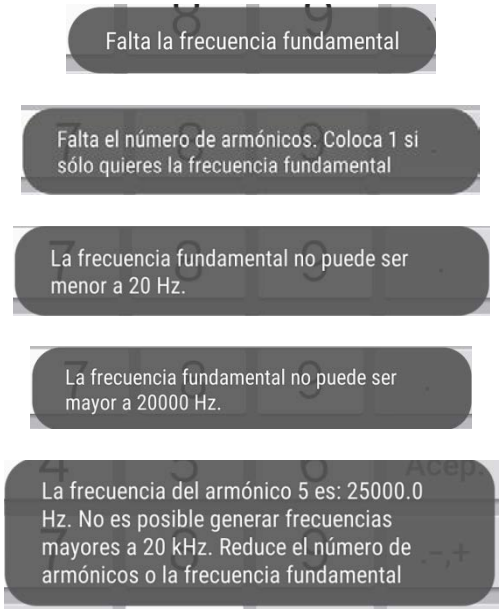


Fig. 4.19 Resultado del botón *Generar* para una frecuencia fundamental de 440 [HZ] y seis armónicos

Estas excepciones generan los siguientes mensajes:



El último mensaje es producto de una fundamental de 5000 [HZ] con 5 armónicos.

Por último, el campo de texto donde se ingresa el número de armónicos tiene varias restricciones:

1. No se puede colocar cero como primer valor
2. Si se escribe uno como primer valor, el segundo valor sólo puede ser cero
3. Si se escribe como primer valor cualquier número distinto de uno no se puede poner un segundo valor

Estas restricciones aseguran que en dicho campo sólo se pueda colocar los número del 1 al 10. No se generan mensajes, sólo no es posible escribir combinaciones de valores fuera de las permitidas.

Ayuda

Tiene la misma función y forma que el botón de *Ayuda* descrito para el analizador de espectro, consiste en un botón que despliega un diálogo de alerta que contiene las características y el funcionamiento de la aplicación *Sumar señales de audio.* Al final del diálogo de alerta está un botón para cerrarlo.

INFORMACIÓN IMPORTANTE

El generador produce tonos puros que pueden ser molestos y dañinos al oído humano, se recomienda bajar el volumen del dispositivo antes de usar el generador

NO USAR CON AUDÍFONOS

4.3 Categoría *Conceptos*

Al ingresar en la categoría *Conceptos* se muestran sus subcategorías (Figura 4.20). Como su nombre lo indica *Conceptos* se refiere al contenido teórico de **Asise**.

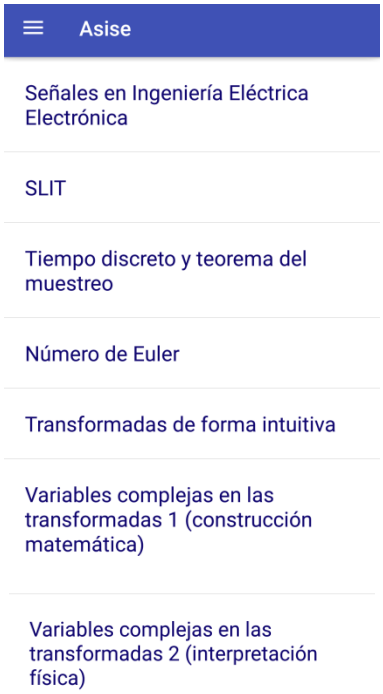


Fig.4.20 Menú de la categoría *Conceptos*

En todos los casos al hacer presionar cualquier elemento de la lista se abre una pantalla donde se observa el contenido en cuestión. En las Figuras 4.21 (a) y (b) se muestra el inicio de los conceptos *SLIT* y *Variables complejas en las transformadas 1*.

Para los conceptos *Señales en ingeniería*, *Tiempo discreto y teorema del muestreo*, se incluyen dos botones la final del texto, estos botones son atajos dentro de **Asise** que llevan a subcategorías de *Aplicaciones* relacionadas con el contenido teórico mostrado. En la Figura 4.22 se muestran los botones de *Señales en ingeniería.*

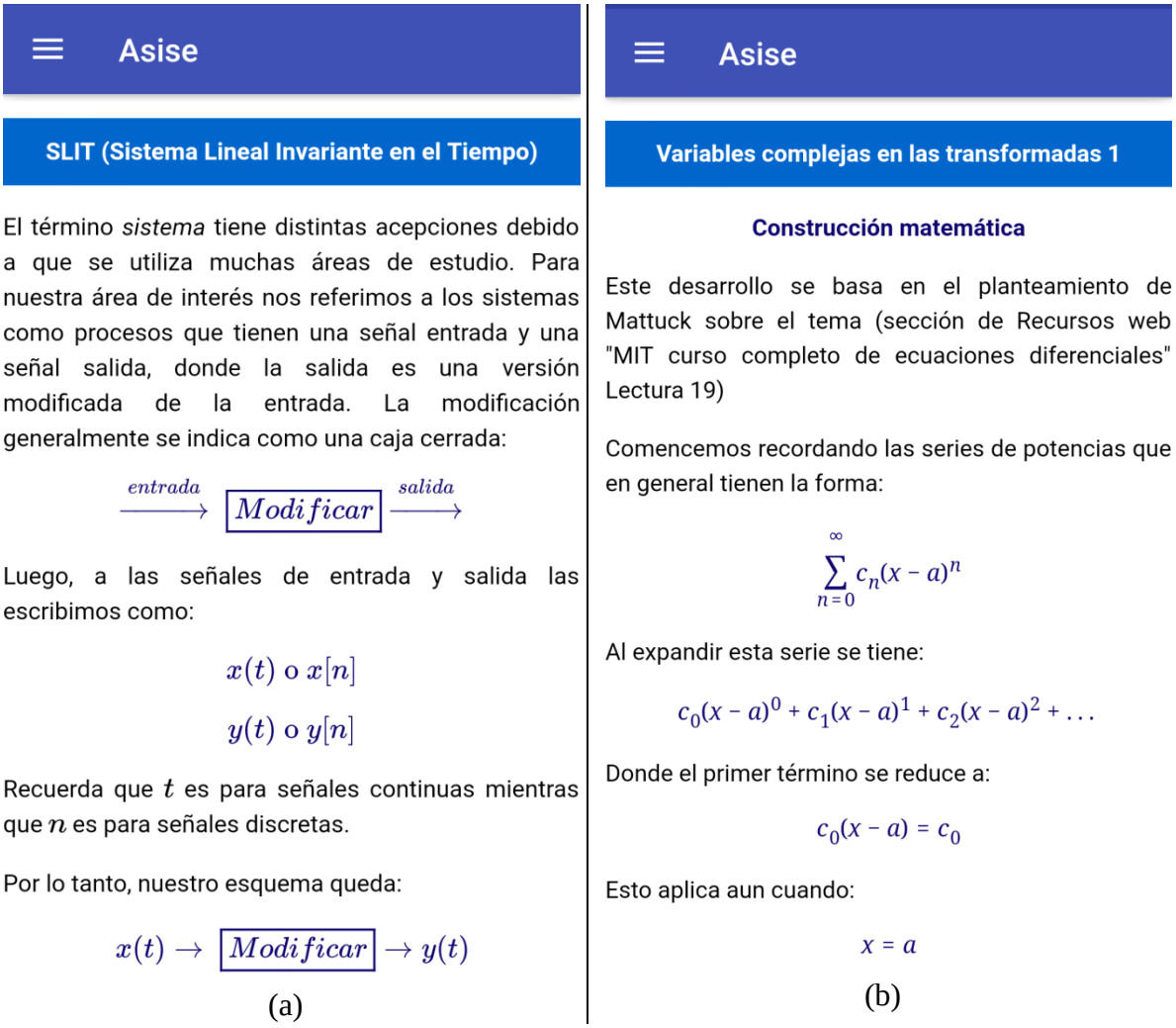


Fig. 4.21 Contenido parcial de (a) concepto *SLIT*, (b) *Variables complejas en las transformadas 1*

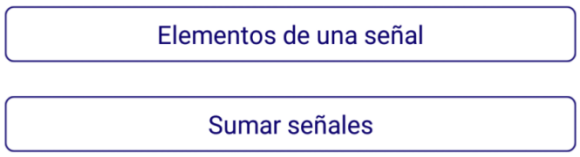


Fig. 4.22 Botones para contenido relacionado de *Señales en ingeniería*

4.4 Categoría *Ejercicios (Series Fourier)*

Al ingresar en la categoría *Conceptos* se muestran sus subcategorías (Figura 4.23). El contenido de esta categoría es práctico como en *Aplicaciones* pero a diferencia de esta última no es interactivo. Se trata de ejercicios sobre la serie trigonométrica de Fourier explicados de forma detallada. No se trata de un solucionario sino de un desarrollo completo de cada caso.

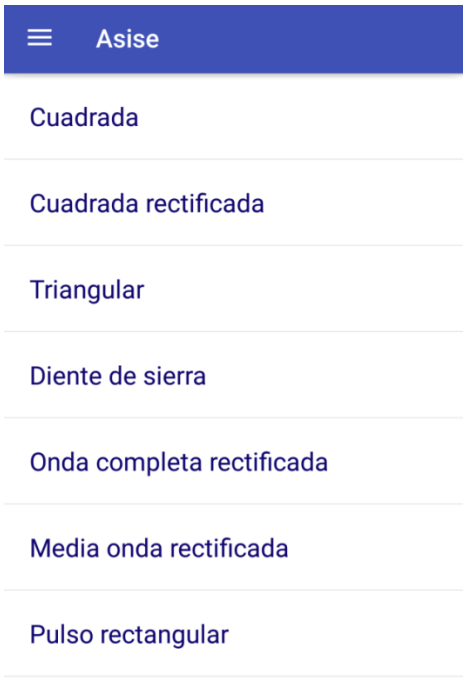


Fig.4.23 Menú de la categoría *Ejercicios (Series Fourier)*

Como se mencionó en el punto 4.1 del presente capítulo el contenido de este menú es idéntico al mostrado en la Figura 4.6 para la subcategoría *Series de Fourier*. Lo anterior, está hecho con la intención de reforzar directamente la relación entre el desarrollo de un ejercicio y la interacción con el mismo de forma aplicada.

En la Figura 4.24 se observa el contenido inicial del ejercicio *Media onda rectificada*. Al final de cada ejercicio hay un botón que lleva directamente a la subcategoría *Series de Fourier* vinculada con cada serie trigonométrica (Figura 4.25).

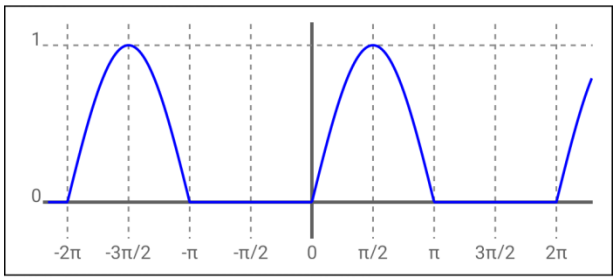
Asise

Ejercicios

Series de Fourier

Señal media onda rectificada

Hallar la serie de Fourier de la siguiente señal:



Paso 1: Escribe la señal como una función a trozos:

$$f(t) = \begin{cases} \text{sen}(t) & \text{para } 0 \leq t < \pi \\ 0 & \text{para } \pi \leq t < 2\pi \end{cases}$$

Paso 2: Escribe la ecuación general para las series de Fourier:

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(n\omega_0 t) + b_n \text{sen}(n\omega_0 t)$$

Fig. 4.24 Contenido parcial del ejercicio *Media onda rectificada*

$$\frac{1}{\pi} + \frac{1}{2}\text{sen}(nt) - \frac{2}{\pi}\sum_{n=1}^{\infty} \frac{\cos(2nt)}{4n^2 - 1}$$

Esta expresión es la serie ya terminada.

Graficar esta serie

Fig. 4.25 Botón para contenido relacionado del ejercicio *Media onda rectificada*

4.5 Categoría *Tablas*

Esta categoría puede considerarse como material de apoyo para el resto del contenido de **Asise**. El menú de esta categoría se muestra en la Figura 4.26. Por otro lado, en la Figura 4.27 se observa, como ejemplo, la tabla correspondiente a la *Transformada z* .

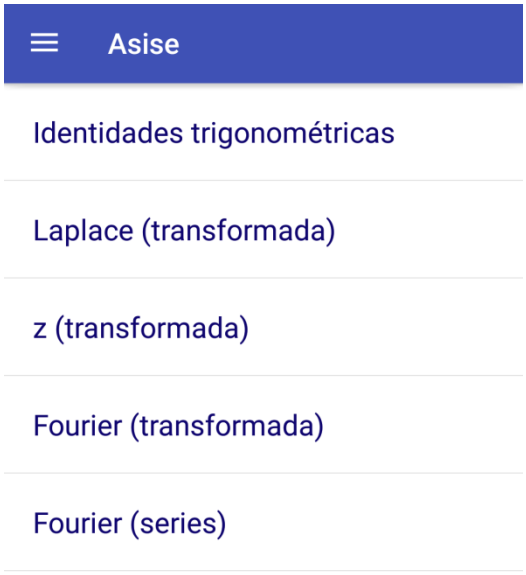


Fig.4.26 Menú de la categoría *Tablas*

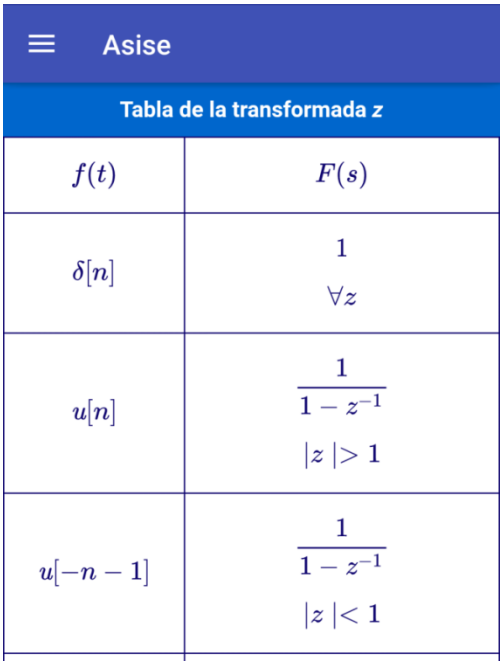


Fig.4.27. Tabla *Transformada z* contenido parcial

4.6 Categoría *Bibliografía*

Esta categoría también es material de apoyo y muestra una lista de libros no es seleccionable (Figura 4.28). Además de incluir parte de la bibliografía que se usó para desarrollar los contenidos de Asise, se incluye un buscador de libros vinculado a la base de datos de las bibliotecas de la Facultad de Ingeniería de la UNAM. La búsqueda puede ser un poco lenta debido al diseño de la página web con la cual se comunica **Asise**. Cuando se presiona el botón *Buscar* se abre una página web (Figura 4.29 (a)) que muestra el progreso de la búsqueda en la base de datos, dicha página se ve muy pequeña dado que no ha sido diseñada para que sea responsiva, por lo tanto, su contenido no se adapta a dispositivos móviles. Lo mismo ocurre cuando se obtienen los resultados (Figura 4.29 (b)). En ambos casos la solución consiste en hacer zoom en las páginas usando la pantalla del dispositivo.

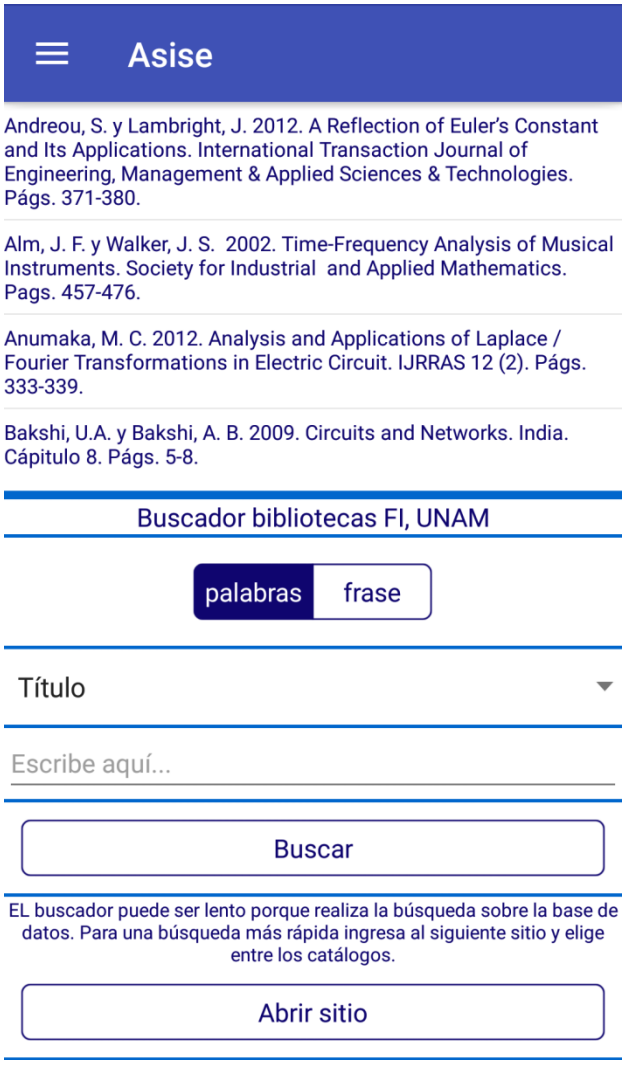
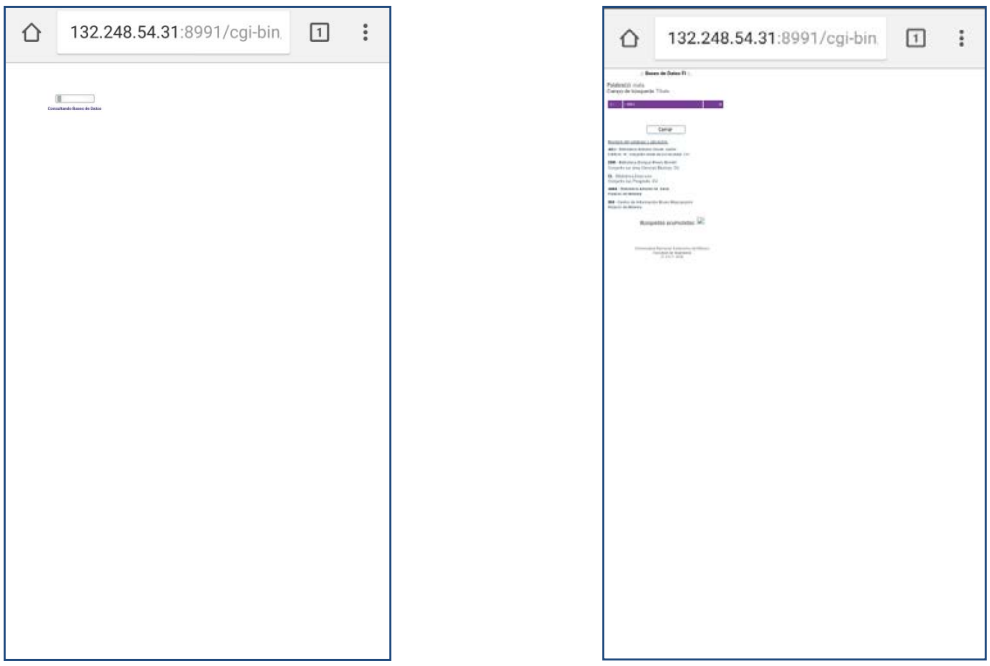


Fig.4.28. Contenido de la categoría *Bibliografía*



(a) (b)

Fig. 4.29. Páginas resultado del botón *Buscar*. (a) Progreso de la búsqueda. (b) Resultados de la búsqueda.

Considerando la tardanza en la búsqueda, se incluye el botón *Abrir sitio*, dicho botón accede directamente a otro página de las bibliotecas mencionadas que permite realizar el proceso de manera más rápida. Dicho sitio tampoco tiene diseño responsivo para dispositivos móviles (Figura 4.30).

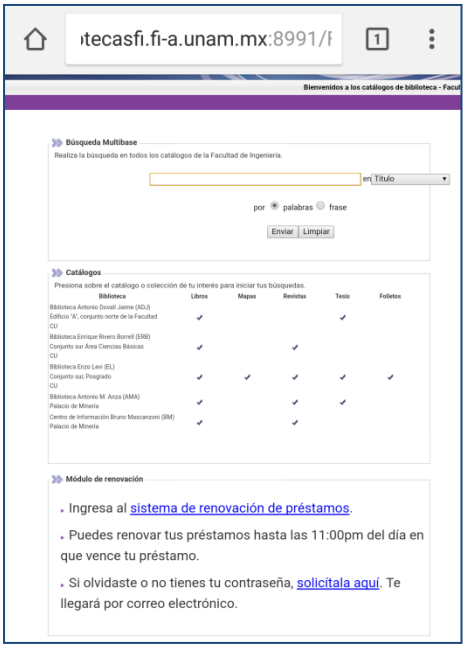


Fig. 4.30. Página web para una búsqueda más rápida de libros.

4.7 Categoría *Recursos web*

Esta categoría se incluyó debido al auge que tienen sitios como YouTube donde es común buscar contenidos relacionados con temas académicos. Consiste en una lista seleccionable donde al presionar sobre un cada elemento se abre un sitio web con el contenido asociado. Cabe mencionar que se colocaron en la lista recursos en español e inglés, así como sitios tipo OCW (Open Course Ware) que son comunes en algunas universidades estadounidenses y que contienen semestres enteros de asignaturas en video lecturas. En la Figura 4.31 se muestra el contenido parcial de esta categoría.

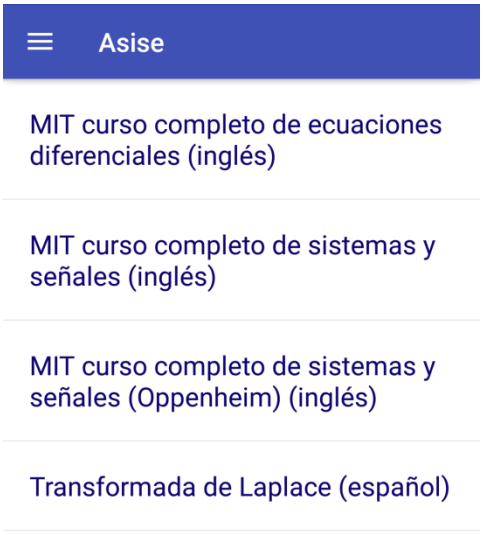


Fig. 4.31 Contenido parcial de la categoría *Recursos web*

5 Conclusiones

Durante en el desarrollo del presente proyecto se lograron implementar algunos temas de la asignatura Análisis de Sistemas y Señales dentro de una aplicación móvil para dispositivos Android. Se abarcaron aspectos teóricos, prácticos (ejercicios resueltos) y se implementaron aplicaciones interactivas y dinámicas.

El aprendizaje de los temas tocados en **Asise** fue relevante ya que la programación de los mismos requirió de su comprensión y entendimiento, este aspecto tuvo mayor énfasis con los ejercicios resueltos y con la aplicaciones, por ejemplo, desarrollar las series trigonométricas de Fourier de forma dinámica demandó muchas pruebas de escritorio para comprobar que el resultado es correcto.

Asise se desarrolló como una alternativa más para los estudiantes, como se mencionó en la introducción, no es sustituto para los libros o programas de escritorio sino una forma distinta de ofrecer opciones de estudio. A nivel global existe un crecimiento constante de aplicaciones móviles orientadas a la educación y **Asise** busca insertase en dicho crecimiento.

A futuro se planea mantener **Asise** actualizada e incrementar su contenido y funcionalidad. Algunas posibles implementaciones son:

1. Agregar más temas de la asignatura Análisis de Sistemas y Señales
2. Desarrollar electrónica externa que pueda interactuar con la aplicación
3. Incluir paquetería del programa Octave y poder ejecutar archivos *.m*
4. Realizar la versión de **Asise** para IOS
5. Actualizar el contenido a través de un servidor
6. Abrir el contenido a colaboradores

Con respecto al primer punto temas como Transformada Z, series exponenciales de Fourier, ecuaciones en diferencias, etc., pueden ser implementados de forma teórica, práctica o como aplicaciones. Por ejemplo, se podrían generar filtros digitales, hacer sonar una canción y observar el resultado de la aplicación de dichos filtros, tanto en un oscilograma como en un analizador de espectro.

Electrónica externa serviría para complementar el contenido de **Asise**. En este sentido las posibilidades son muy extensas, los dispositivos móviles pueden proporcionar la interfaz de usuario para el despliegue de información, por ejemplo, se podría implementar un electrocardiógrafo de forma externa y usar **Asise** para el despliegue gráfico en tiempo real, también se podrían controlar motores, graficar señales de sensores, etc., cualquier información que se adquirida por medio de electrónica externa se puede enviar a un dispositivo móvil por medio de bluetooth. Por supuesto, se requiere que dicha información ya esté digitalizada.

Incluir Octave en **Asise** puede ser un proyecto a largo plazo dados los requerimientos para lograrlo. Actualmente existe una versión de Octave para Android pero esta aplicación no es soportada por los desarrolladores de Octave, se trata de un esfuerzo independiente que ha recibido críticas dado que Octave no viene implementado completamente y para tener acceso a ello es necesario realizar un donativo al autor, si este aspecto es correcto o no escapa del ámbito del presente trabajo, se narra aquí con la intención de mostrar que la vinculación puede ser llevada a cabo. En el caso de **Asise**, no se buscaría hacer una versión móvil de Octave sino usar la capacidad de este programa para ejecutar archivos *.m* y realizar algún tipo de análisis. También se exploraría su capacidad como generador de gráficos.

Un reto particular de esta implementación serían los paquetes de Octave que se incluyan, es necesario considerar que dichos paquetes son archivos con tamaños grandes, por lo tanto, sería necesario emular el trabajo que se realizó con MathJax, para el desarrollo de **Asise**, y reducir su tamaño a lo mínimo posible.

Realizar una versión para iPhone y iPad significaría el complemento ideal de **Asise**, la ventaja sería que el contenido ya está definido y las pruebas de escritorio que confirman los resultados ya se llevaron a cabo, por lo tanto, la implementación en IOS ya no requeriría de este proceso. Lo anterior, no quita toda la carga de programación en un nuevo lenguaje, pero sí simplifica el camino.

Actualizar el contenido a través de un servidor es un proceso muy usado actualmente, se realiza de esta forma para evitar que la publicación de nuevo contenido obligue a generar una nueva versión de la aplicación, este proceso requiere de mayor tiempo de espera, es necesario dar de baja la versión anterior en el Play Store de Google y subir una nueva versión. En este sentido, contenido teórico y ejercicios resueltos se podrían enviar a la aplicación desde un servidor y mejorar la experiencia del usuario, por ejemplo, se podría poner un botón para descargar más ejercicios, el cual al ser presionado buscaría en el servidor si hay contenido nuevo, de existir este contenido se descargaría al dispositivo y el listado de ejercicios se incrementaría dándole al usuario nuevos ejercicios.

Con respecto a la colaboración de terceros, **Asise** se abriría para que los profesores de la asignatura puedan colaborar con contenido teórico o ejercicios, donde se daría crédito al profesor que colabora tanto en el ejercicio como en una categoría llamada colaboradores. En la Figura 5.1 (a) se muestra el menú de **Asise** modificado para incluir la colaboración de terceros, en la Figura 5.1(b) se observa el contenido de dicha categoría. También sería necesario modificar la estructura de **Asise** quitando del menú principal la categoría *Ejercicios (Series Fouirer)* y sustituyéndolo por otro que fuese sólo *Ejercicios*, al interior de éste último colocar el correspondiente a series de Fourier junto con otras opciones como *Transformada Z*. Se mencionó que también existiría crédito en el ejercicio, para ello en

cada ejercicio se colocaría en el encabezado el nombre del profesor que colaboró con dicho ejercicio.

En suma, las posibilidades de crecimiento para **Asise** son muchas, este factor dependerá de la recepción que tenga por parte de los estudiantes y profesores de la asignatura quienes determinarán si la aplicación contribuye o no de manera efectiva al aprendizaje. Si el resultado es positivo podría extenderse el trabajo de **Asise** hacia otras asignaturas.

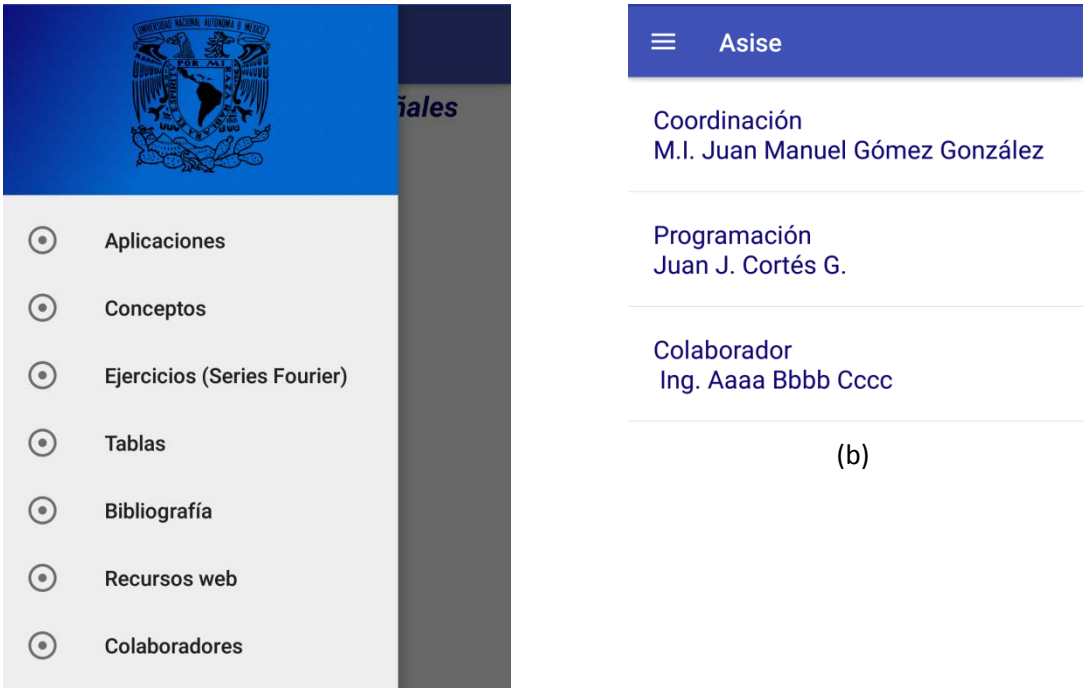


Fig. 5.1. (a)Menú de **Asise** modificado con la categoría *Colaboradores.* (b) Lista de colaboradores

6 Bibliografía y Mesografía

Alm, J. F. y Walker, J. S. 2002. Time-Frequency Analysis of Musical Instruments. *Society for Industrial and Applied Mathematics*. Pags. 457-476.

Andreou, S. y Lambright, J. 2012. A Reflection of Euler's Constant and Its Applications. *International Transaction Journal of Engineering, Management & Applied Sciences & Technologies*. Págs. 371-380.

androidstidio. 2017. Imagen tomada de: http://www.androidestudio.com/2015/08/

Anumaka, M. C. 2012. Analysis and Applications of Laplace /Fourier Transformations in Electric Circuit. IJRRAS 12 (2). Págs. 333-339.

AppBrain(1). 2017, Most Popular Google Play Categories. Información consultada en línea: https://www.appbrain.com/stats/android-market-app-categories

AppBrain(2). 2017, All Time Popular Education Apps. Información consultada en línea: https://www.appbrain.com/apps/popular/education/

Bakshi, U.A. y Bakshi, A. B. 2009. Circuits and Networks. India. Cápitulo 8. Págs. 5-8.

Chang, R. 2017. U.S. Education Apps Market to Grow 20.15% in 2016-2020. *The Journal, Transforming Education Through Technology*. Documento consultado en línea:

https://thejournal.com/Articles/2017/01/26/U.S.-Education-Apps-Market-to-Grow-28.15-Percent-in-2016-to-2020.aspx

Ferry, B. 2009. Using mobile phones to enhance teacher learning in environmental education. Australia, University of Wollongong, Faculty of Education, Faculty of Social Sciences. Capítulo 5. Págs. 45-55.

Ghosh, .S. P. y Chacraborty, A.K. 2010. Network Analysis and Synthesis. New Delhi. Págs. 233-235.

Gok, N. y Khanna, N. 2013. Bulding Hybrid Android Apps with Java and JavaScript. O'Reilly. USA. First Edition.

Greaves, T. W., Hayes, J., Wilson, L., Gielniak, M., Peterson, E. L. 2012. Revolutionizing Education Though Technology. ISTE, USA.

INEGI. 2015. Estadísticas a propósito del día mundial de Internet.

INEGI. 2016. Estadísticas a propósito del día mundial de Internet.

Kallolkar, V. S. y Salunke, S. G. 2015. Relation between Fourier, Laplace and Z-transforms. *International Journal of Scientific & Engineering Research*. Págs. 16-18.

Lüke, H. D. 1999. The Origins of the Sampling Theorem. *IEEE Communications Magazine*. Págs. 106-108.

Mata, H. G., Sánchez, E. V. M., Gómez, G. J. M. 2002. Análisis de Sistemas y Señales con Cómputo Avanzado. Facultad de Ingeniería, UNAM, México.

Mattuck, A. 2003. Lecture 19: Introduction to the Laplace Transform. Material consultado en línea: https://ocw.mit.edu/courses/mathematics/18-03-differential-equations-spring-2010/video-lectures/

McIntosh, A. I. 2017, On Euler's Number e. Boston University. Págs.1-5. Documento consultado en línea: http://people.bu.edu/aimcinto/number.e.pdf

Oppenheim, A. v. y Willsky, A. S. 1998. Sistemas y Señales. Prentice Hall, México 2da edición.

Spiegel, M. R., Ph. D.1965. Theory and Problems of Laplace Transforms. Schaum's.

Sudhakar, A. y Palli, S.S. 2010. Circuits and Networks Analysis and Synthesis. New Delhi. Págs. 694-697.

Tognetti, K. 1998. e the Exponential - the Magic Number of Growth. School of Mathematics and Applied Statistics University of Wollongong NSW 2522 Australia. Págs.1-32.

Tenner, E. 2013. Higher Education's Internet Revolution. *The American. American Enterprice Institute*. Documento consultado en línea:

http://www.aei.org/publication/higher-educations-internet-revolution/

Tan, E. S. Q. y Teo, D. J. L. 2015. Appsolutely smartphones: Usage and perception of apps for educational purposes. *Asian Journal of the Scholarship of Teaching and Learning*, *5*(1), 55-75.

Tretter, S. A. 2013. Discrete-Time Signal Processing. *John Wiley & Sons*, Págs. 227–230.

INFORMACIÓN IMPORTANTE

Las marcas Samsung, Verizone, Amazon, Motorola, IPhone, Matlab, Octave, Nexus, Google y Android pertenecen a sus respectivos dueños y son utilizadas sólo como referencia.