

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESTUDIO Y COMPARACIÓN DE MÉTODOS MODERNOS PARA LA DETERMINACIÓN DEL COMPORTAMIENTO EN TIEMPO- FRECUENCIA DE UNA SEÑAL

**TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO
REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN
TELECOMUNICACIONES**

BRYAN LEANDRO CELORIO GUAMÁN

bryan.celorio@epn.edu.ec

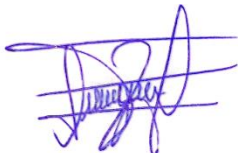
DIRECTOR: Ph.D. ROBIN GERARDO ÁLVAREZ RUEDA

robin.alvarez@epn.edu.ec

DMQ, abril 2023

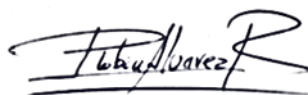
CERTIFICACIONES

Yo, Bryan Leandro Celorio Guamán declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.



Bryan Leandro Celorio Guamán

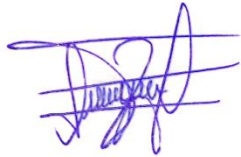
Certifico que el presente trabajo de integración curricular fue desarrollado por Bryan Leandro Celorio Guamán, bajo mi supervisión.



Ph.D. Robin Álvarez Rueda
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.



LEANDRO CELORIO



ROBIN ÁLVAREZ

DEDICATORIA

A mis madres Sonia y Carmelina, pues sin ellas esto jamás hubiese sido posible.

A mí, porque logré superar cada reto, porque tuve la valentía de levantarme en cada caída y porque ahora soy mi mejor versión.

AGRADECIMIENTO

Al terminar esta gran etapa de mi vida quiero expresar mi agradecimiento a quienes hicieron posible este reto, aquellos que caminaron junto a mí y fueron apoyo, inspiración y fortaleza. Esta mención en especial es para mis padres Sonia y Leandro, a mis hermanos Mary y Mateo, a mi sobrino Leandro P, a mi prima Estefanía G. Y sobre todo a mis abuelitos Carmelina y Andrés quienes desde el cielo sé que están muy orgullosos de mí.

A mis amigos, Elizabeth Y, Jorge T, Jannis C, Melanny D, Daysi G, Vanessa V, Dayanara T y Sebastian V, quienes a más de ser mis amigos son la familia que escogí. Gracias por compartir momentos de euforia a mi lado y ser soporte en momentos difíciles.

A mi fiel amigo Bigotes quien llegó a mi vida para ayudarme a iniciar mi día con alegría y está junto a mí en todo momento.

Finalmente, mi gratitud con el Dr. Robin Álvarez quien brindo su apoyo a lo largo de este proyecto y orientó cada idea con mucho profesionalismo.

ÍNDICE DE CONTENIDO

CERTIFICACIONES.....	I
DECLARACIÓN DE AUTORÍA.....	II
DEDICATORIA.....	III
AGRADECIMIENTO.....	IV
ÍNDICE DE CONTENIDO.....	V
RESUMEN	VII
ABSTRACT	VIII
1 INTRODUCCIÓN.....	1
1.1 OBJETIVO GENERAL	2
1.2 OBJETIVOS ESPECÍFICOS	2
1.3 ALCANCE	2
1.4 MARCO TEÓRICO.....	3
1.4.1 ANÁLISIS DE SEÑALES EN EL DOMINIO TIEMPO-FRECUENCIA.....	3
1.4.2 MÉTODOS TIEMPO-FRECUENCIA.....	4
1.4.1.1 Transformada de Gabor	5
1.4.1.2 Reasignada de Fourier	6
1.4.1.3 Sincronizada de Fourier.....	8
1.4.1.4 Superlets	10
2 METODOLOGÍA.....	13
2.1 GENERACIÓN DE SEÑAL MULTICOMPONENTE	15
2.2 IMPLEMENTACIÓN DE LA TRANSFORMADA DE GABOR EN MATLAB	20
2.3 IMPLEMENTACIÓN DE LA REASIGNADA DE FOURIER EN MATLAB	23
2.4 IMPLEMENTACIÓN DE LA SINCRONIZADA DE FOURIER EN MATLAB	27
2.5 IMPLEMENTACIÓN DE LOS SUPERLETS EN MATLAB	30
3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES.....	33
3.1 RESULTADOS.....	33
3.1.1 COMPARACIÓN DE MÉTODOS - PRUEBA 1	33
3.1.2 COMPARACIÓN DE MÉTODOS - PRUEBA 2	35
3.1.3 COMPARACIÓN MÉTODOS - PRUEBA 3.....	36
3.1.4 RESULTADOS OBTENIDOS CON EL MÉTODO TRANSFORMADA DE GABOR	38

3.1.5	RESULTADOS OBTENIDOS CON EL MÉTODO REASIGNADA DE FOURIER	39
3.1.6	RESULTADOS OBTENIDOS CON EL MÉTODO SINCRONIZADA DE FOURIER	41
3.1.7	RESULTADOS OBTENIDOS CON EL MÉTODO SUPERLETS	42
3.1.8	RESUMEN GENERAL.....	43
3.2	CONCLUSIONES.....	45
3.3	RECOMENDACIONES	46
4	REFERENCIAS BIBLIOGRÁFICAS	46
5	ANEXOS.....	50
	ANEXO I Función ASLT	51
	ANEXO II Función FASLT	56
	ANEXO III Código Completo	62
	ANEXO IV Manual de Usuario	68

RESUMEN

En el presente estudio se muestra los resultados más importantes para la comparación de cuatro métodos modernos en el análisis de señales en el dominio tiempo-frecuencia: Transformada de Gabor, Reasignada de Fourier, Sincronizada de Fourier y Superlets. Para esto se hará uso de una señal artificial multicomponente cuya variación de frecuencias y amplitudes dará como resultado ventajas y desventajas de cada uno de los métodos antes mencionados, tanto en términos de resolución en tiempo, resolución en frecuencia, carga computacional y detección de componentes muy pequeñas.

Esta comparación tomará como punto de referencia el diagrama de tiempo - frecuencia ideal, esta representación será utilizada para las 3 pruebas establecidas cuyo fin es mostrar los límites y características de cada método. La primera prueba refleja la carga computacional requerida por cada método con el uso de frecuencias relativamente altas, la segunda se centra en introducir requerimientos de resolución en tiempo y en frecuencia y, por último, la tercera prueba introduce requerimientos de detección de las componentes con amplitudes muy pequeñas. En conjunto estas tres pruebas permiten realizar un análisis general del uso de dichos métodos empleando para ello objetos o funciones disponibles en Matlab.

PALABRAS CLAVE: Comparación de métodos tiempo- frecuencia, Transformada de Gabor, Reasignada de Fourier, Sincronizada de Fourier, Superlets.

ABSTRACT

The present study shows the most important results for the comparison of four modern methods in the analysis of signals in the time-frequency domain: Gabor Transform, Reassigned Fourier Transform, Synchronized Fourier Transform and Superlets. For this purpose, an artificial multicomponent signal will be used, whose variation of frequencies and amplitudes will result in advantages and disadvantages of each of the above-mentioned methods, both in terms of time resolution, frequency resolution, computational load and detection of very small components.

This comparison will take as a reference point the ideal time-frequency diagram, this representation will be used for the 3 tests established to show the limits and characteristics of each method. The first test reflects the computational load required by each method with the use of relatively high frequencies, the second test focuses on introducing time and frequency resolution requirements and, finally, the third test introduces requirements for the detection of components with very small amplitudes. Together these three tests allow a general analysis of the use of these methods using objects or functions available in Matlab.

KEYWORDS: Time-frequency method comparison, Gabor Transform, Reassigned Fourier Transform, Synchronized Fourier Transform, Superlets, results.

1 INTRODUCCIÓN

Los fenómenos existentes en la naturaleza como la voz humana, eventos sísmicos, bioseñales (cerebro, corazón, respiración, etc.), etc., están representados por señales las cuales poseen características específicas como amplitud, frecuencia y duración. Su análisis es de amplia discusión en diversas áreas correspondientes: neurociencias, astrofísica, geotectónica, etc., donde se realizan investigaciones para tratar de determinar sus características tanto en el dominio del tiempo, en el dominio de la frecuencia o en el dominio tiempo-frecuencia.

Históricamente han aparecido varios métodos para determinar el comportamiento de una señal en su dominio temporal y frecuencial; sin embargo, debido a la gran cantidad de métodos y su complejidad, la decisión sobre qué método utilizar se convierte en una tarea problemática para los investigadores. La falta de conocimiento en la utilización de los diferentes métodos, así como en sus limitaciones, puede desencadenar graves consecuencias al momento de interpretar los resultados obtenidos.

Como es conocido, todas las técnicas de estimación espectral tienen el problema de que no contienen información temporal, por lo cual no se conoce en qué momento sucede cada una de ellas. Para mitigar este problema se han desarrollado diferentes métodos o algoritmos para el estudio y análisis en el dominio tiempo-frecuencia, los cuales permiten visualizar la evolución del contenido espectral en función del tiempo.

Se han encontrado varios estudios comparativos entre diferentes métodos clásicos de análisis de señales en tiempo-frecuencia [1], [2], [3], [4], [5]; sin embargo, no ha sido posible encontrar un documento que realice la comparación de métodos modernos y novedosos con respecto a las características de resolución de frecuencia, resolución de tiempo, detección y carga computacional.

En este estudio se va a realizar la comparación de cuatro métodos modernos: Transformada de Gabor, Reasignada de Fourier, Sincronizada de Fourier y Superlets, para ello se hará uso de señales artificiales conocidas que están conformadas por algunas componentes con distintas amplitudes, duración y frecuencia. De esta manera, conociendo el comportamiento de dichas señales, se podrá determinar cuál de ellos es el que mejor predice aquel comportamiento y qué método posee las mejores características tanto en resolución en tiempo, en resolución de frecuencia y en detección de componentes muy pequeñas.

1.1 OBJETIVO GENERAL

El objetivo principal es realizar una comparativa de los cuatro métodos modernos de análisis de señales tiempo-frecuencia: Transformada de Gabor, Reasignada de Fourier, Sincronizada de Fourier y Superlets, tomando en cuenta su resolución en tiempo, resolución en frecuencia, detección de componente muy pequeñas y su carga computacional, para determinar qué método muestra los mejores resultados.

1.2 OBJETIVOS ESPECÍFICOS

1. Estudiar los métodos de análisis de las señales en su dominio tiempo-frecuencia: Transformada de Gabor, Reasignada de Fourier, Sincronizada de Fourier y Superlets.
2. Implementar scripts en Matlab con ayuda de funciones y librerías propias este software que permitan observar los resultados producidos por cada uno de dichos métodos.
3. Efectuar distintas pruebas para evaluar cada uno de estos métodos y realizar comparaciones en base a los resultados obtenidos.
4. Establecer, en base a lo anterior, ventajas y desventajas de cada método propuesto y concluir cuál de ellos muestra los mejores resultados en resolución temporal, resolución frecuencia, detección de componentes muy pequeñas y carga computacional.

1.3 ALCANCE

Empleando el software de Matlab en su versión R2021a, se implementará señales artificiales constituidas por varias componentes con distintas frecuencias, amplitudes y duraciones. Esto permitirá realizar pruebas respecto de resolución temporal, resolución frecuencia y capacidad de detección de cada uno de los métodos propuestos. Así también se implementará el diagrama de tiempo-frecuencia ideal con el objetivo de tener una referencia en la comparación de dichos métodos.

Para evaluar a cada uno de los métodos, se empleará tanto librerías como funciones propias o externas a Matlab.

En base a las pruebas, se llevará a cabo una comparación de los resultados obtenidos por los cuatro métodos planteados, dando mayor importancia a la resolución en tiempo, resolución en frecuencia, detección de componentes muy pequeñas y su carga

computacional. Como resultado de esta comparación se determinará que método posee las mejores prestaciones y características.

1.4 MARCO TEÓRICO

1.4.1 ANÁLISIS DE SEÑALES EN EL DOMINIO TIEMPO-FRECUENCIA

Respecto del análisis de señales, usualmente se representa una señal en función del tiempo $s(t)$, la cual proporciona información temporal como inicio, fin y su evolución a lo largo del tiempo, pero es muy complicado o imposible de conocer sus componentes de frecuencia. Por otro lado, se puede analizar dicha señal en su dominio de la frecuencia, donde se puede localizar las componentes de frecuencias más relevantes contenidas en la señal [6]; sin embargo, es imposible conocer la duración de dichas componentes. Como se puede apreciar, ambas representaciones contienen distinta información y se requiere conocer toda esta información en base a algún método conocido como dominio tiempo-frecuencia.

Un primer acercamiento este dominio fue la Transformada de Fourier (TF) la cual permite convertir una función en el dominio del tiempo al dominio de la frecuencia o viceversa como se muestra en la Figura 1, este proceso se realiza con la ayuda de la suma de senos, cosenos y/o exponenciales complejas [7].

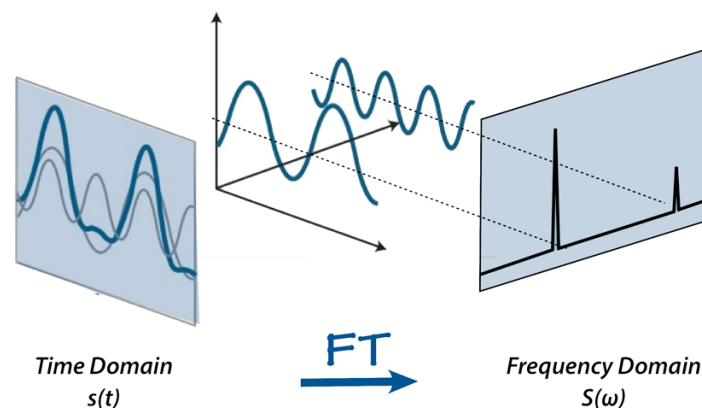


Figura 1. Representación de la Transformada de Fourier. Basado en [8].

No obstante, la Transformada de Fourier posee una gran limitante para las señales no estacionarias, donde, al no tener información temporal, la información en frecuencia no llegará a ser suficiente para lograr determinar diferencias entre una señal u otra [9]. Es decir, si se toma una señal y se le aplica la Transformada de Fourier, lo que se visualiza

es la distribución de las componentes espectrales; sin embargo, no se podrá obtener la información temporal de cada una de estas componentes [6].

El análisis en el dominio tiempo-frecuencia plantea la combinación de ambos dominios con el fin de obtener la mayor información de las características de dicha señal temporal.

1.4.2 MÉTODOS TIEMPO-FRECUENCIA

Los métodos de análisis tiempo-frecuencia se han desarrollado desde finales del siglo XX y han tomado relevancia y mayor interés en los últimos años. La base fundamental y la necesidad yace en la importancia de buscar un método matemático óptimo el cual permita representar de forma simultánea la variación espectral a lo largo del tiempo.

En 1946 Dennis Gabor introduce uno de los métodos pioneros en el estudio de las señales en el dominio de tiempo-frecuencia, la Transformada de Fourier de tiempo corto (Short-time Fourier transform STFT) la cual hace uso del término *ventana temporal* $g(t)$. La ventana enmarca una porción de la señal a la cual se le aplica la TF, de este modo se tiene información de la frecuencia localizada temporalmente en el dominio efectivo de la ventana. Al desplazar dicha ventana a lo largo de la duración de la señal se determina el comportamiento espectral en cada una de las zonas de tiempo [10]. La STFT está definida por la siguiente ecuación:

$$S_t(\omega) = \frac{1}{\sqrt{2\pi}} \int e^{-j\omega\tau} s_t(\tau) g(\tau - t) d\tau \quad (1)$$

Donde $s_t(\tau)$ es la señal temporal y $g(\tau - t)$ es la ventana deslizante. La función de ventana se elige de tal forma que la señal no sea alterada abruptamente alrededor del tiempo t . Dicha ventana deslizante ocupa distintos tiempos tal como se observa en la Figura 2 que corresponde al módulo de la STFT, denominado **espectrograma**.

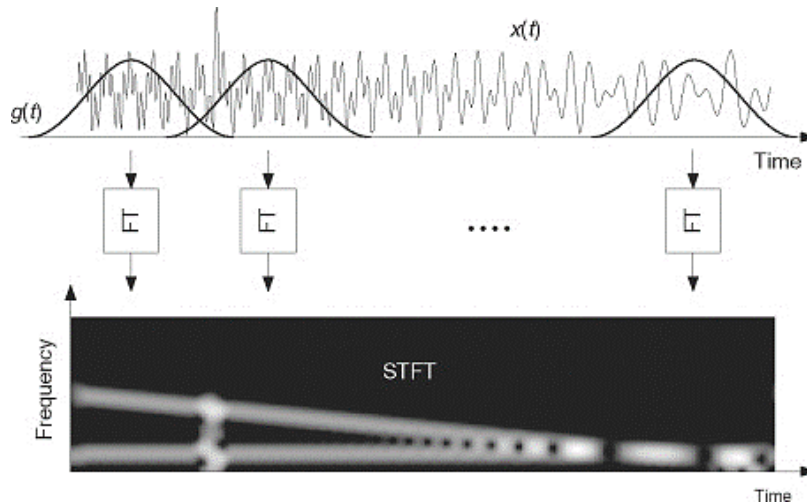


Figura 2. Representación de la Transformada de Fourier de tiempo corto (STFT). Basado en [11].

Es conocido que, a mayor longitud de ventana, se obtiene mejor resolución en frecuencia (se aprecia mejor las componentes muy cercanas), pero peor resolución en tiempo (se aprecia mejor las duraciones de las componentes); y lo opuesto, a menor longitud de ventana se obtendrá peor resolución en frecuencia, pero mejor resolución en tiempo, lo cual se constituye en un compromiso entre estas dos resoluciones. Adicionalmente, este método tendrá un límite respecto del tamaño mínimo de las amplitudes de las componentes que puede detectar. Este método es un punto de partida y en base a este se han desarrollado nuevos métodos cuyo objetivo principal es obtener los resultados respecto de dichos parámetros.

En específico, el presente trabajo va dirigido al estudio y comparación de cuatro métodos en el dominio tiempo-frecuencia: Transformada de Gabor, Reasignada de Fourier, Sincronizada de Fourier y Superlets.

1.4.1.1 Transformada de Gabor

La Transformada de Gabor es un caso especial de la STFT en la cual, como ya se comentó, $g(t)$ es una función de ventana que puede ser: rectangular, Hann, Blackman, entre otras. Para el caso de la Transformada de Gabor la función ventana que se emplea es la Gaussiana dado que esta representa una distribución normal, además matemáticamente, al emplear esta ventana, permite que la transformada de Gabor sea invertible [12].

Para el caso de la Transformada de Gabor el objetivo consiste en la obtención de los denominados “átomos” (a_{mk}) cuya representación matemática está definida de la siguiente forma [6]:

$$a_{mk} = \int x(t) \cdot \omega_{mk}(t) dt \quad (2)$$

Donde $\omega_{mk} = \omega(t - mT)e^{jk\Omega t}$ es una función de ventana con m y k valores enteros, $\Omega \leq 2\pi$ se define como la celda de muestreo. Es así como a_{mk} proporciona los valores de amplitud para cada átomo y además indica qué señal puede reconstruirse con la suma de señales provenientes de una señal básica (función ventana). Las limitaciones de este método se originan en el tamaño de la celda de muestreo, si es demasiado grande se tendrá poca información, y, por otro lado, si es demasiado pequeña se genera redundancia en la reconstrucción de la señal [6]. En la Figura 3 se muestra la representación de una señal y el resultado al aplicar la Transformada de Gabor, de esta se observa la duración de la señal y la frecuencia asignada. Esta representación muestra de forma conjunta el comportamiento de una señal.

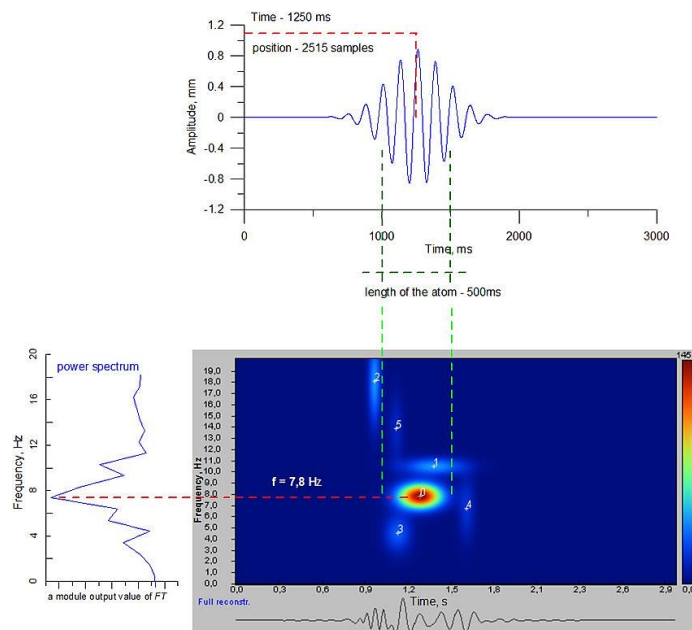


Figura 3. Representación de la Transformada de Gabor. Basado en [13].

1.4.1.2 Reasignada de Fourier

El proceso de reasignación tiene como objetivo el cálculo de las estimaciones espectrales afinadas en las diferentes componentes de frecuencia. Esto se lo realiza partiendo de las derivadas parciales del espectro de fase. En otras palabras, en lugar de localizar las componentes en el centro geométrico de la ventana de análisis (espectrograma común), las componentes se van a reasignar al centro de gravedad de la distribución espectral. Este fundamento propone que la energía de la señal (distribuida en el semiplano T-F), contribuye de manera significativa en las zonas donde la fase tiene una variación lenta

comparada con el entorno. La Figura 4 muestra el principio de funcionamiento del método de reasignación, las elipses representan la ventana de análisis mientras que los círculos representan el punto de asignación de la energía que para este método se refiere al centro geométrico de la ventana, finalmente los triángulos representan el centro de gravedad de la energía y son el punto de reasignación [14].

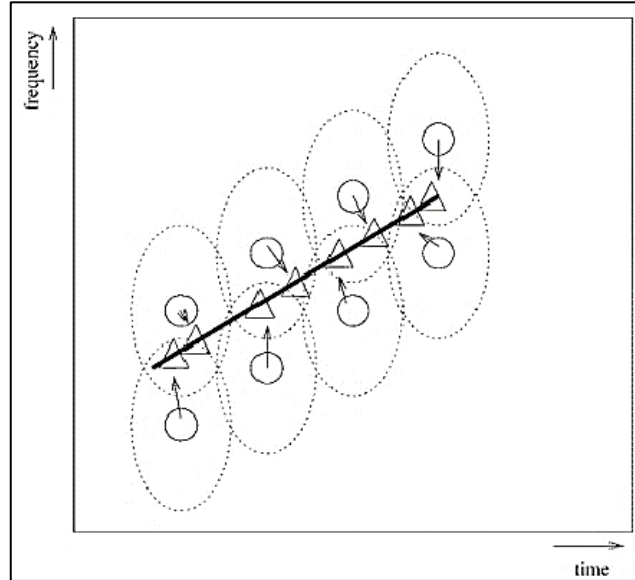


Figura 4. Esquema del funcionamiento de la reasignada de frecuencias. Basado en [14]

La parte matemática del método de reasignación se puede analizar de diversas formas, donde los resultados, si bien divergen ligeramente, estos se asemejan. Una de las posibles maneras de calcular las posiciones tiempo–frecuencias reasignadas (tomado de [15]) es:

$$\hat{t}_{k,n} = n - \Re \left[\frac{X_{t,n} X_n^*(k)}{|X_n(k)|^2} \right] \quad (3)$$

$$\hat{\omega}_{k,n} = k + \Im \left[\frac{X_{f,n} X_n^*(k)}{|X_n(k)|^2} \right] \quad (4)$$

Donde $\hat{t}_{k,n}$ y $\hat{\omega}_{k,n}$ representa las localizaciones en tiempo y frecuencia corregidas, correspondientes a la k –ésima componente espectral centrada en el tiempo n (en muestras, asumiendo ventanas de análisis de longitud impar). $X_{t,n}$ y $X_{f,n}$ hacen referencia a las transformadas localizadas, utilizando una ponderación enventanada temporal y frecuencial (respectivamente) de la señal, mientras que \Re y \Im son la parte real e imaginaria de lo encerrado entre corchetes. Además, $X_n(k)$ representa la Transformada de Fourier de tiempo corto (STFT) centrada en un tiempo n [16].

a. Espectrograma Reasignado

El espectrograma reasignado consiste en realizar un espectrograma en el que las frecuencias más perceptibles en el gráfico son únicamente las más intensas, es decir, solo se consideran los picos espectrales más oscuros y la demás frecuencia desaparecen [17]. Esto se obtiene mediante la reasignación de frecuencias la cual consiste en desplazar las frecuencias con menor intensidad hasta los puntos con mayor intensidad, dando como resultado que las frecuencias más intensas se refuercen con ayuda de las frecuencias vecinas con menor intensidad [18].

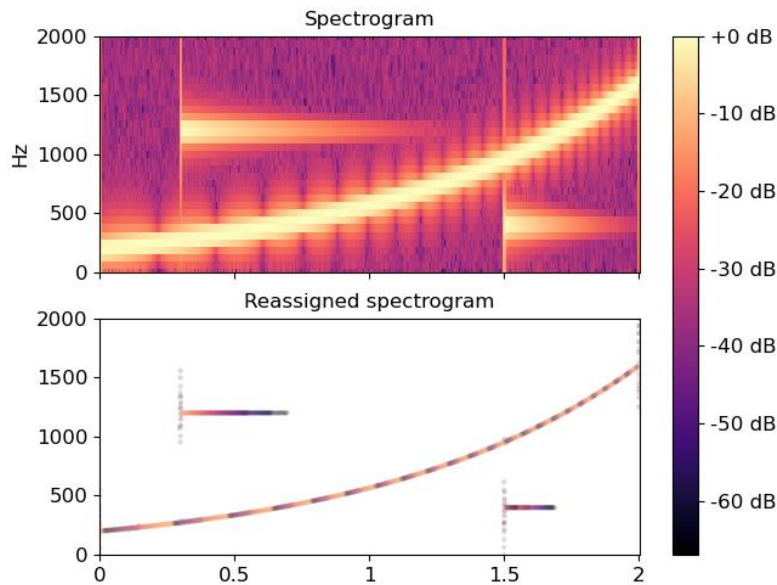


Figura 5. Resultados de Espectrograma Tradicional VS Espectrograma Reasignado. Basado en [19].

En la Figura 5 se evidencia como el método de reasignación agudiza los datos borrosos en comparación del **espectrograma tradicional** o módulo de la STFT. Dicho de otra manera, la reasignación permite un mapeo de coordenadas y su reubicación en el diagrama tiempo-frecuencia con una mayor precisión para las señales que pueden separarse en tiempo y frecuencia en relación a la ventana de análisis [20].

1.4.1.3 Sincronizada de Fourier

La transformada Sincronizada de Fourier (Fourier-based synchrosqueezing Transform o FSST), es un método de reasignación que tiene como objetivo afinar una representación de escala de tiempo y frecuencia, mientras permanece invertible. Las energías de los coeficientes de frecuencia de tiempo se reasignan para lograr una mayor energía alrededor de las trayectorias, lo que da como resultado un seguimiento más preciso de estos [21]. La Figura 6 se muestra el proceso de afinamiento en los resultados al emplear este método,

las líneas resultantes muestran mayor precisión y nitidez al comparar con el módulo de la STFT.

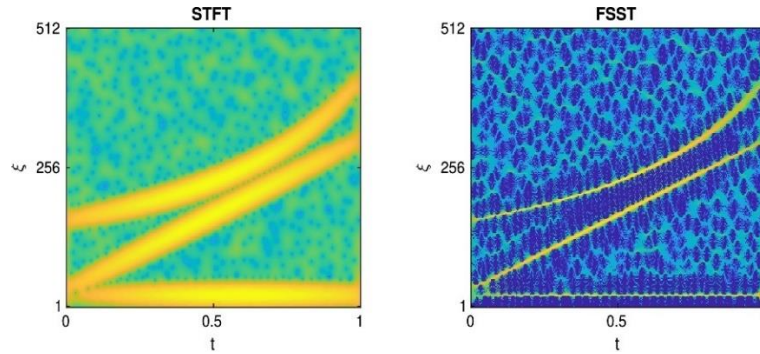


Figura 6. Representación y evolución de la resolución tiempo-frecuencia de una señal empleando STFT y FSST. Basado en [22].

La transformada sincronizada de Fourier parte de la transformada de Fourier en tiempo corto STFT, la Transformada sincronizada se asemeja al espectrograma reasignado dado que genera estimaciones de tiempo-frecuencia con mayor nitidez que la transformada convencional. es muy similar al espectrograma reasignado pues genera estimaciones de tiempo-frecuencia más claras que la transformada convencional [23].

$$V_g f(t, \eta) = \int_{-\infty}^{\infty} f(\tau) g(\tau - t) e^{-j2\pi\eta(\tau-t)} d\tau \quad (5)$$

En la ecuación se observa una gran diferencia con la ecuación convencional de la STFT dado que se agrega un factor extra $e^{j2\pi\eta t}$, donde η y t representan las variables de tiempo y frecuencia respectivamente. Al emplear dicho factor los valores de transformación se reducen para que se concentren alrededor de las curvas de frecuencia instantánea (FI) en el plano de tiempo-frecuencia $T_g f(t, \omega)$, con ello las frecuencias instantáneas se estiman con la transformación de fase, la cual viene descrita por [24]:

$$\omega_f(t, \eta) = \frac{\frac{\partial}{\partial t} V_g f(t, \eta)}{j2\pi V_g f(t, \eta)} \quad (6)$$

La ecuación (5) es que a $\frac{\partial}{\partial t} V_g f(t, \eta)$ es una aproximación a la frecuencia instantánea de f y al realizar la división para $V_g f$ se elimina la influencia de la ventana g y por lo tanto se logra agudizar la representación gráfica de tiempo-frecuencia [25].

$$T_g f(t, \omega) = \int_{-\infty}^{\infty} V_g f(t, \eta) \delta(\omega - \Omega_g f(t, \eta)) d\eta \quad (7)$$

La ecuación (7) representa el operador de sincronización (Synchrosqueezing operator) el cual comprime el contenido de $V_g f(t, \eta)$ a las curvas de la frecuencia instantánea para formar el operador $T_g f(t, \omega)$ [23].

1.4.1.4 Superlets

Un "superlet" (SL) se define como un conjunto de wavelets con una frecuencia central fija, y que abarca un rango de ciclos diferentes (restringiendo progresivamente el ancho de banda). Un SL es un estimador espectral que permite una superresolución en frecuencia y de tiempo que utiliza conjuntos de wavelets con un ancho de banda cada vez más restringido. Estos se combinan geoméricamente para mantener una buena resolución temporal de ondículas individuales y ganar resolución en frecuencia en las bandas superiores [26]. La Figura 7 muestra el principio de funcionamiento de los Superlets donde este se consigue al combinar wavelets cortas y una gran AB con wavelets más largas y de AB estrecho.

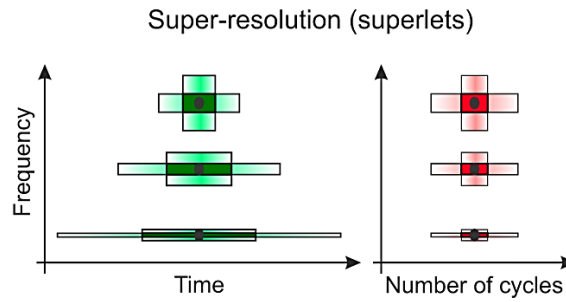


Figura 7. Esquema de funcionamiento de Superlets. Basado en [26].

Para obtener un SL se puede partir de una wavelet base, como, por ejemplo, Morlet con un número fijo de ciclos, lo cual proporciona una resolución temporal relativa constante, pero degrada la resolución en frecuencia (aumento de la redundancia) a medida que aumenta la frecuencia central de la wavelet. Al aumentar el parámetro de dispersión de tiempo de la wavelet, es decir más ciclos, se incrementa la resolución en frecuencia, pero se pierde resolución temporal.

$$SL_{f,o} = \{\psi_{f,c} | c = c_1, c_2, \dots, c_o\} \quad (8)$$

En la ecuación (8), o es el orden de la SL y los valores de c_1, c_2, \dots, c_o son los números de los ciclos para cada wavelet en el conjunto. Además, se establece como un SL es un conjunto finito de o wavelets que abarcan varios AB en la misma frecuencia central f .

Para incrementar la resolución, se combinan ondículas cortas con alta resolución temporal, (pequeño número de ciclos, bajo parámetro de dispersión de tiempo), con ondículas más

largas, con alta resolución de frecuencia (mayor número de ciclos, menor resolución temporal), de la misma manera es posible hacerlo con los espectrogramas [26].

a. Adaptive Superlets (Superlets Adaptativos)

Los Superlets Adaptables (ASL) ajustan su orden a la frecuencia central, esto con el objetivo de compensar el aumento del AB de la wavelet con el aumento de la frecuencia. En la Transformada Adaptativa de Superlets (ASLT), se inicia con un orden bajo para estimar las frecuencias bajas y se aumenta el orden en función de la frecuencia, esto mejora la representación tanto en frecuencia como en tiempo, esto se consigue se la siguiente forma [26]:

$$ASLf = SL_{f,o} | o = a(f) \tag{9}$$

En la ecuación (9) y (10), $a(f)$ representa una función creciente de la frecuencia central, que tiene posee valores enteros. Una opción simple es variar el orden linealmente:

$$a(f) = o_{min} + round \left[(o_{max} - o_{min}) \cdot \frac{f - f_{min}}{f_{max} - f_{min}} \right] \tag{10}$$

donde, o_{min} es el orden correspondiente a la frecuencia central más pequeña: f_{min} , mientras que o_{max} es el orden correspondiente a la frecuencia central más grande: f_{max} , en la representación tiempo-frecuencia. La Figura 8 muestra una señal con diferentes amplitudes y duración, esta señal es evaluada con la transforma de tiempo corto STFT y con Superlets (ASLT), de esta se evidencia que al emplear ASLT existe una mayor concentración de la energía, por lo que no existe dispersión ni alargamiento como se muestra en el caso de STFT.

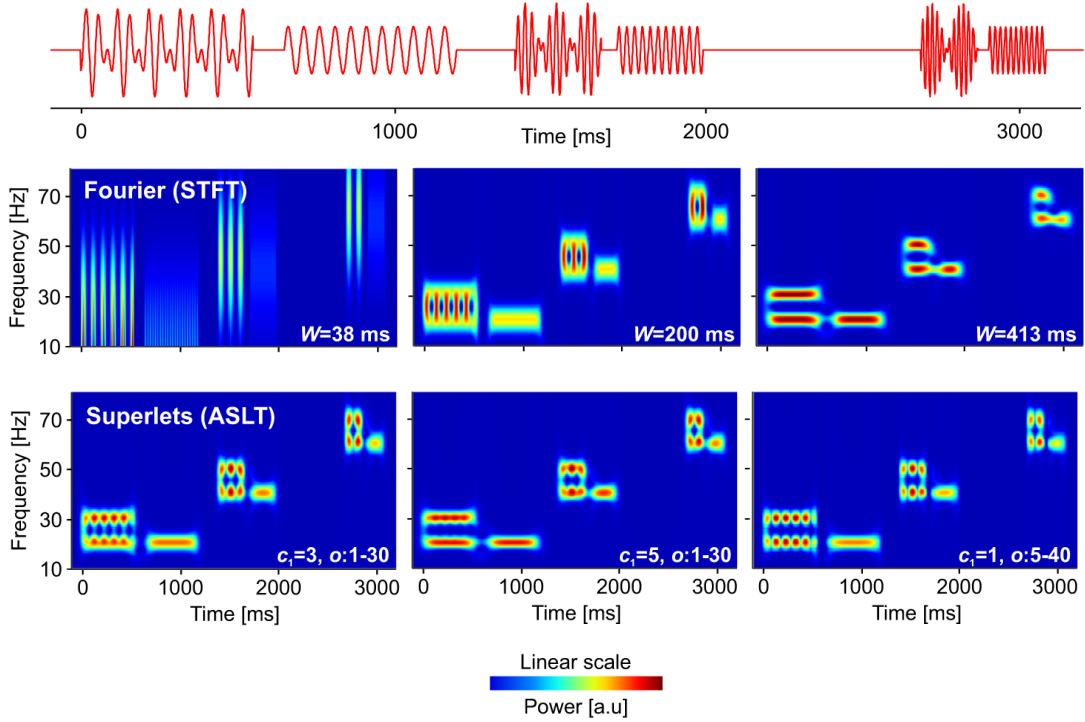


Figura 8. Representación y evolución de la resolución tiempo-frecuencia de una señal empleando STFT y Adaptive Superlets (ASLT). Modificado de [26].

b. Adaptive fractional Superlets (Superlets fraccionarios adaptativos)

En este caso se introduce la idea de "Superlets fraccionarios", que permite tener valores no enteros en la ASLT. Se parte de la definición de la media geométrica ponderada y se considera que cada wavelet del conjunto de la parte entera posee un valor de 1, mientras que la última tiene un valor fraccionario. Esto permite obtener una variación continua y suave proporcionando así una representación más nítida en el dominio de la frecuencia. Se define el "orden fraccionario" como [27]:

$$o_w = o_i + \alpha \quad (11)$$

En la ecuación (11), o_w representa el orden fraccionario, compuesto por un orden entero $o_i \geq 1$, y una parte fraccionaria, $\alpha \in [0,1)$. En definitiva, la ASLT fraccionaria (FASLT) introduce ciclos fraccionarios entre los ciclos integrales en (10), permitiendo así solucionar el problema de bandas discontinuas con el aumento de frecuencia [28].

Utilizando los Superlets fraccionarios, no es necesario restringir el orden de la ASLT a valores enteros. FASLT simplemente utiliza el orden fraccionario verdadero, como [27]:

$$o_w(f) = o_{min} + \left[(o_{max} - o_{min}) \cdot \frac{f - f_{min}}{f_{max} - f_{min}} \right] \quad (12)$$

En ecuación (12) al igual que en la ecuación (10), O_{min} y O_{max} representan el orden correspondiente a la frecuencia central más pequeña f_{min} y más alta f_{max} respectivamente. La única diferentes es que (12) carece del operador *round* mismo que aproxima al inmediato entero, qué para el caso de FASLT no es necesario. La Figura 9 representa la comparación entre los resultados obtenidos al emplear ASLT y FASLT, a simple vista lo que caracteriza a los Superlets fraccionales es la continuidad en la representación gráfica, es decir, existe mayor nitidez y menos cortes en comparación a los Superlets Adaptativos.

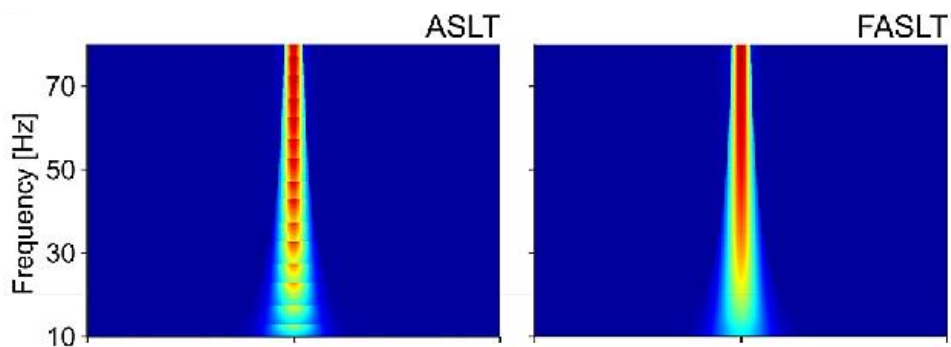


Figura 9. Representación y evolución de la resolución tiempo-frecuencia de una señal empleando ASLT y FASLT. Modificado de [27].

2 METODOLOGÍA

Este capítulo mostrará la comparativa de los cuatro métodos estudiados para el análisis tiempo-frecuencia: Transformada de Gabor, Reasignada de Fourier, Sincronizada de Fourier y Superlets.

Para esta comparación se realizará 3 pruebas donde se evaluará la resolución en tiempo, resolución en frecuencia capacidad de detección de componentes muy pequeñas y la carga computacional de cada uno de estos métodos.

1. Prueba para evaluación de la carga computacional

En esta primera prueba todas las componentes de frecuencia están lo suficientemente alejadas (no hay problema de resolución en frecuencia) y todas tendrán una amplitud igual a 1 (no hay problema de detección de componentes pequeñas). Constará de 5 componentes de frecuencias relativamente altas en un intervalo de [100 - 300] [Hz] que permitirá evaluar el límite computacional de cada método planteado.

2. Prueba para evaluación de resolución tiempo y resolución en frecuencia

Se tendrán 5 componentes en frecuencias bajas en el intervalo [11 - 44] [Hz]. Todas tendrán una amplitud igual a 1. Además, se aprovechará para examinar también la carga computacional de cada método.

La segunda prueba evaluará los métodos con frecuencias bajas en un intervalo [11 - 44] [Hz] y una amplitud igual a 1 en todos los componentes, aquí el objetivo principal es el análisis en resolución en tiempo y frecuencia.

3. Prueba para evaluación de la detección de componentes muy pequeñas

Finalmente, en la tercera prueba se realizará una variación de amplitudes y frecuencias con el objetivo de analizar la detección de las componentes más pequeñas en la señal compuesta y su representación visual con cada método, dichas amplitudes irán en un rango de [0.001 - 1].

En las pruebas se asignará valores de frecuencia cercanos con la finalidad de mostrar el nivel de nitidez y el alcance de la detección de cada componente de frecuencia. Así también la duración de cada componente pondrá a prueba el nivel de distorsión y alargamiento del espectrograma resultante.

La siguiente tabla muestra las frecuencias empleadas, así como la duración y amplitud para la señal multicomponente utilizada en las pruebas.

Tabla 1. Tabla de parámetros empleados en cada prueba

Prueba	Componente	Duración [s]	Amplitud	Frecuencia [Hz]
1 (Evaluación Carga Computacional)	C1	[0 1]	1	100
	C2	[1 4]	1	130
	C3	[2 3]	1	175
	C4	[3 5]	1	250
	C5	[4 5]	1	300
2 (Evaluación de resolución en tiempo y	C1	[0 1]	1	11
	C2	[1 4]	1	20
	C3	[2 3]	1	30
	C4	[3 5]	1	38

resolución en frecuencia)	C5	[4 5]	1	44
3 (Evaluación para la detección de componentes muy pequeñas)	C1	[0 1]	0.001	80
	C2	[1 4]	0.01	130
	C3	[2 3]	0.1	165
	C4	[3 5]	0.1	200
	C5	[4 5]	1	235

Para el análisis y comparación de la carga computacional se hará uso la función *tíc toc* de Matlab, la función *tíc* registra el tiempo inicial mientras que la función *toc* utiliza el valor tomado con el fin de calcular el tiempo transcurrido [29]. Esto se lo va a realizar para cada uno de los métodos con el fin de tener valores comparables entre sí.

El equipo empleado para realizar la experimentación fue un laptop con Matlab en su versión R2021a, con características técnicas: Windows 10 Pro, Procesador Intel(R) Core (TM) i5-4200U CPU @ 1.60GHz, 2301 MHz y 8 Gb de memoria RAM.

Es imprescindible remarcar que, si bien la función *tíc toc* muestra el tiempo de procesamiento de cada código, esto no incluye el tiempo en que se demora en mostrar las gráficas. En ciertos métodos este tiempo fue superior al empleado por la función. Dichos detalles se analizarán en el capítulo 3 (Resultados).

2.1 GENERACIÓN DE SEÑAL MULTICOMPONENTE

Para la implementación de la señal compuesta se hará uso de los parámetros establecidos en la Tabla 1. Adicionalmente, para tener un campo visual correcto se presenta de cada una de las componente y la señal compuesta, así como también el diagrama de tiempo-frecuencia ideal.

La señal final “*yt1*” es el resultado de la suma de 5 componentes, cada una con su amplitud, frecuencia y duración asignada. Los valores de frecuencia se ingresan de forma ascendente teniendo como F_s (frecuencia de muestreo) al valor más exigente en comparación a las demás frecuencias. La frecuencia de muestreo a su vez permite asignar

el tiempo de muestreo “*T_s*” con el que se crea el vector de tiempo “*t*” que se emplea para cada componente de la señal final.

Para la obtención del diagrama de tiempo - frecuencia ideal se lo realiza mediante el uso del objeto “*plot*”, para esto se le asigna el intervalo de tiempo de cada componente y la frecuencia correspondiente en forma de vector, el cual toma cada tiempo y se le destina una frecuencia hasta que la duración de la componente termine.

Finalmente, para que en el diagrama de tiempo - frecuencia se visualicen todas las componentes de la señal, se usa “*hold on*” lo cual permite unir en un mismo gráfico todas las componentes de la señal de forma que cada una conserve su posición establecida.

A continuación, se muestra el segmento de código empleado para la generación de la señal multicomponente, este será empleado para todos los métodos y para todas las pruebas, realizando los cambios necesarios en los parámetros según se mencionan en la Tabla 1.

```
%% 1. Recepción de datos (frecuencia, amplitudes y duraciones)
de todas las componentes:
% Los datos siguientes solo deben ser reemplazados según se desee
introducir problemas de:
% - Resolución en frecuencia: tonos muy juntos
% - Detección: componentes muy pequeñas relativas a la mayor de
1.

% %1.1 Frecuencias de las componentes (ordenadas de menor a
mayor):
% Prueba 1
f1=1000; f2=2000; f3=3000; f4=3500; f5=4000;

% Prueba 2
%f1=11; f2=20; f3=30; f4=38; f5=44;

%Prueba 3
%f1=80; f2=130; f3=165; f4=200; f5=235;

%% Amplitudes: en este caso todas las amplitudes son iguales
%Prueba 1 y 2
a1 = 1; a2 = 1; a3 = 1; a4 = 1; a5 = 1;
```

```

% Prueba 3
%a1 = 0.001; a2 = 0.01; a3 = 0.1; a4 = 0.1; a5 = 1; % 1.3
Duraciones de las componentes:
];
t1=[0 1]; t2=[1 4]; t3=[2 3]; t4=[3 5]; t5=[4 5];
%% 2. Generación de la señal compuesta (suma de todas las
componentes):
Fs=10*f5; % Fs respecto de la f5 que es la más exigente
Ts=1/Fs;
duracion = 5; % es el tiempo máximo que ocupan las componentes
t=0:Ts:duracion;
%% MÉTODO 2: aprovechando las ventajas de programación de
Matlab,
% se realiza el método anterior pero ahora de manera compacta:
% Primeramente se generan cada una de las componentes en todo el
intervalo
comp1=a1*sin(2*pi*f1*t);
comp2=a2*sin(2*pi*f2*t);
comp3=a3*sin(2*pi*f3*t);
comp4=a4*sin(2*pi*f4*t);
comp5=a5*sin(2*pi*f5*t);

% Se llena de ceros las componentes en los intervalos donde no
existen pero se aprovecha
% las posibilidades de Matlab:
comp1=(t<=1).*comp1;
comp2=((t>=1)&(t<=4)).*comp2;
comp3=((t>2) & (t<=3)).*comp3;
comp4=((t>=3)&(t<=5)).*comp4;
comp5=((t>=4)&(t<=5)).*comp5;

% Presentación de las componentes en forma ascendente tal como
es el
% diagrama TF ideal:
figure

```

```

subplot(6,1,1);plot(t,comp5);xlim([0 5]);title('componente 5');
xlabel('t(s)');grid on;
subplot(6,1,2);plot(t,comp4);xlim([0 5]);title('componente 4');
xlabel('t(s)');grid on;
subplot(6,1,3);plot(t,comp3);xlim([0 5]);title('componente 3');
xlabel('t(s)');grid on;
subplot(6,1,4);plot(t,comp2);xlim([0 5]);title('componente 2');
xlabel('t(s)');grid on;
subplot(6,1,5);plot(t,comp1);xlim([0 5]);title('componente 1');
xlabel('t(s)');grid on;

% Una vez generadas las componentes, las sumamos para obtener la
señal compuesta:
yt1 = comp1+comp2+comp3+comp4+comp5;
subplot(6,1,6);plot(t,yt1,'r');title('Señal compuesta');
xlabel('t(s)');grid on;
sgtitle('Componentes individuales y señal compuesta')
xlim([0 5])

% Visualización del TF ideal en conjunto con la señal compuesta
obtenida de
% manera que se vea la existencia de una o varias componentes
según los intervalos
figure
hold on
grid minor
plot (t1,[f1 f1],'linewidth',1.5)
plot (t2,[f2 f2],'linewidth',1.5)
plot (t3,[f3 f3],'linewidth',1.5)
plot (t4,[f4 f4],'linewidth',1.5)
plot (t5,[f5 f5],'linewidth',1.5)
hold off
title('DIAGRAMA TIEMPO-FRECUENCIA IDEAL')
xlabel('t(sec)')
ylabel('F(Hz)')
xlim([0 5])

```

```

ylim([0 600])
legend(['a1 = ' num2str(a1)], ['a2 = ' num2str(a2)], ['a3 = '
num2str(a3)], ['a4 = ' num2str(a4)], ['a5 = ' num2str(a5)])

```

La Figura 10 muestra 5 componentes y la señal compuesta, según los parámetros de la prueba 1. La Figura 11 muestra el diagrama tiempo - frecuencia ideal el cual servirá como base para las comparaciones entre los 4 métodos.

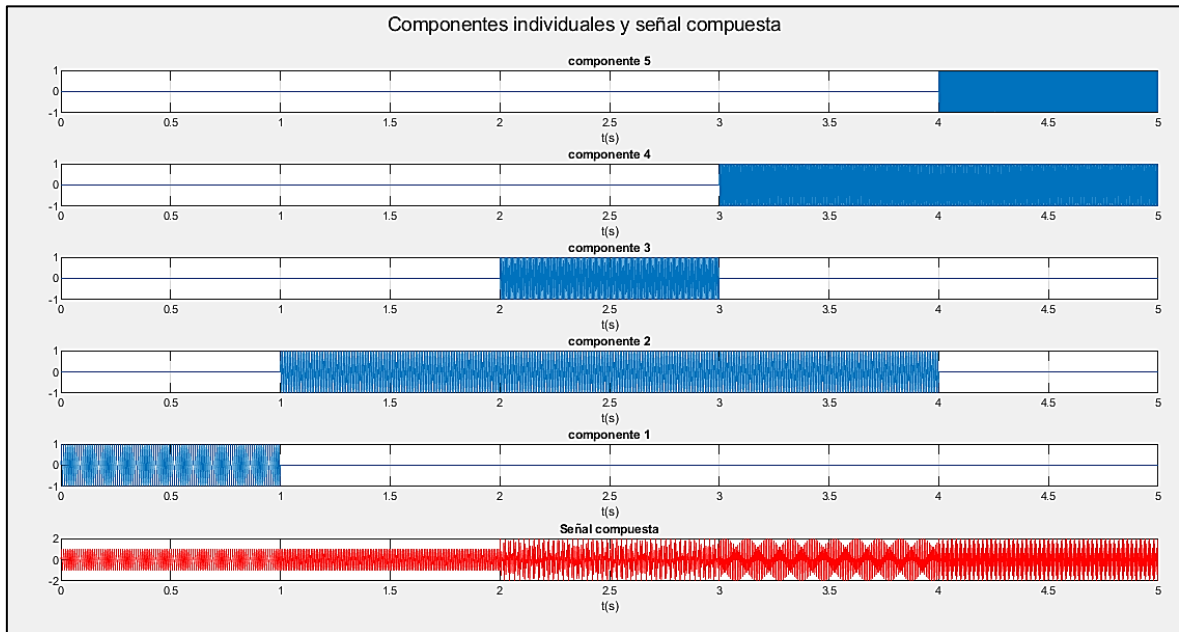


Figura 10. Componentes individuales y señal compuesta para la prueba 1.

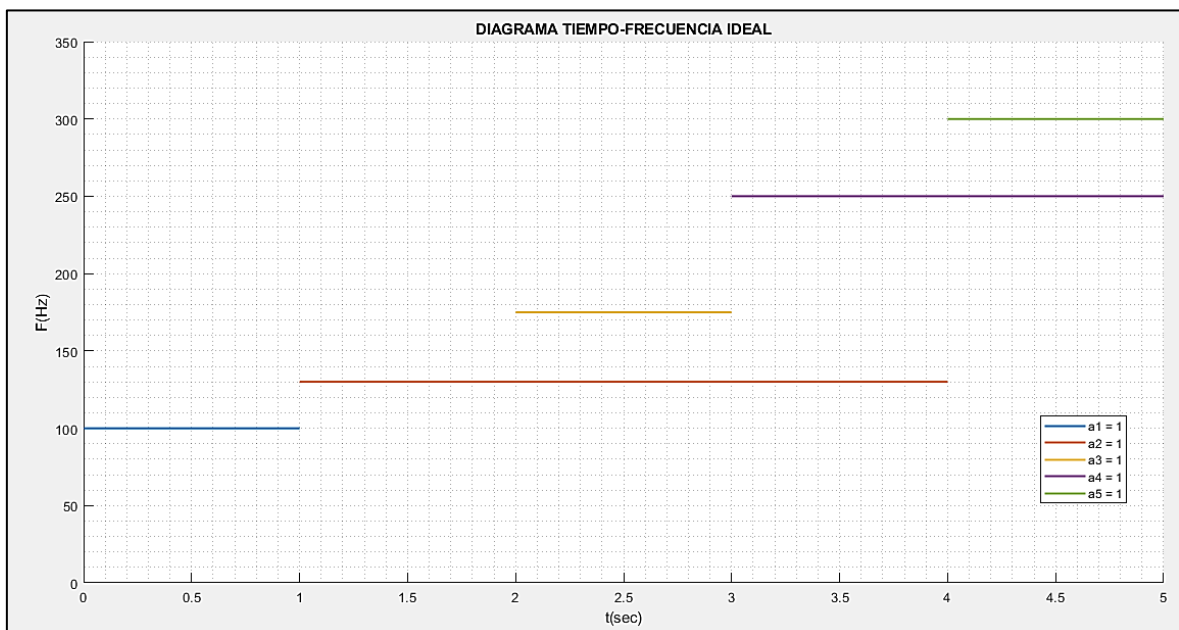


Figura 11. Diagrama tiempo-frecuencia ideal para la prueba 1.

2.2 IMPLEMENTACIÓN DE LA TRANSFORMADA DE GABOR EN MATLAB

“Signal Processing Toolbox” es una herramienta de Matlab la cual ofrece funciones para el análisis procesamiento y extracción de características de señales en su dominio tiempo-frecuencia. Para el caso de la Transformada de Gabor, el Toolbox de Matlab posee el objeto *DGTtool* el cual proporciona la Transformada de Gabor Discreta, y puede ser representado de la siguiente forma:

```
F =
DGTtool('windowShift',5,'windowLength',250,'FFTnum',nFFT_ventana_RT,'windowName','Gauss')
```

En la Tabla 2 se muestran los parámetros de esta función.

Tabla 2. Parámetros DGTtool [30]

PARÁMETRO	DESCRIPCIÓN
WindowsShift	Representa un entero positivo el cual va a definir el desplazamiento de la ventana
WindowLength	Este parámetro va a representar el tamaño de ventana donde es un valor entero positivo
FFTnum	Número de intervalos de frecuencia en el dominio de tiempo-frecuencia, este valor debe ser entero positivo
WindowName	Nombre de la ventana a emplear, puede ser cualquiera del siguiente listado: <ul style="list-style-type: none"> - 'Hann' - 'Blackman', - '3termC1Nuttall' - '3termC3Nuttall' - '4termC1Nuttall' - '4termC3Nuttall' - '4termC5Nuttall' - 'Gauss' - 'Slepian' - 'Chebyshev' El nombre de la ventana se puede acortar (por ejemplo, 'b' es aceptable en lugar de 'Blackman').

Es importante resaltar que algunas de las opciones se pueden omitir dado que el objeto tomará valores predeterminados para su correcto uso. Además, el orden de las opciones no está restringido (cualquier orden es aceptable). Finalmente, se necesitará que la señal a ser analizada sea un vector de columna, además de la frecuencia y el tiempo de muestra cómo se indica en la sintaxis de código.

En base a la Tabla 2 se observa que existen diferentes opciones de ventana (parámetro “*WindowName*”). Para el caso teórico de la Transformada de Gabor, la ventana empleada es la Gaussiana, dicha ventana permite una vista del diagrama tiempo-frecuencia más compacta, es decir se minimiza la incertidumbre en los dominios tiempo y frecuencia.

Los parámetros de “*windowShift*”, “*windowsLength*” y “*FFTnum*” serán completados con los valores que permitan un equilibrio temporal y espectral, para el caso del tamaño de ventana (*windowsLength*) y (*FFTnum*) el valor empleado es: 2048, a medida que este parámetro decrece, habrá una resolución temporal mayor, pero limitará la resolución en frecuencia. Por otro lado, el desplazamiento de ventana (*windowShift*) tendrá un valor de 256, donde un valor mayor afectará la resolución en tiempo.

Para la representación gráfica se hace uso del objeto *plot*, donde los valores obtenidos en la variable de salida *F* por medio del objeto *DGTtool*. El siguiente código permite analizar la señal en el dominio tiempo-frecuencia empleando el método Transformada de Gabor:

```
% Espectrograma haciendo uso de DGTtool que utiliza la
transformada discreta de Gabor
tic
tic
xn = yt1;
inputSignal = yt1'; %señal de entrada

% Se crea el objeto DGTtool
% Algunas de las opciones se pueden omitir (se utilizarán los
valores predeterminados).
% El orden de las opciones no está restringido (cualquier orden
es aceptable).
% La lista de ventanas disponibles se puede obtener mediante
DGTtool.windowList.
```

```

% El nombre de la ventana se puede acortar (por ejemplo, 'b' es
aceptable en lugar de 'Blackman').f
% Si la versión de MATLAB es 2021a o posterior, se puede
utilizar la siguiente sintaxis.
% F =
DGTtool(windowShift=20,windowLength=250,FFTnum=400,windowName='B
lackman')
F =
DGTtool('windowShift',256,'windowLength',2048,'FFTnum',2048,'win
dowName','Gauss');
% Se calcula la Transformada de Gabor
% La señal de entrada debe ser un vector de columna.
% La señal de entrada puede ser multicanal (muestras x canales).
% La frecuencia normalizada f y el índice de muestra t se pueden
obtener de la siguiente manera:
% [X,f,t] = F(x)
X = F(inputSignal); %X contiene los valores complejos de la
transformada generada por DGTtool
%x = F.pinv(X);
toc
% Trazar el espectrograma
% Gráfica del espectrograma. La frecuencia de muestreo se puede
omitir: F.plot(x).
% Nota: Las funciones de trazado se pueden usar directamente
después de definir F.
F.plot(inputSignal,Fs)
sgtitle('Transformada de Gabor ventana de Gauss');
toc

```

A continuación, se presenta un ejemplo del funcionamiento del objeto DGTtool para el análisis de señales en su dominio tiempo - frecuencia. Para esto se emplea la señal multicomponente con frecuencias de 350, 400, 435, 500 y 750 Hz. La Figura 12 muestra los resultados obtenidos.

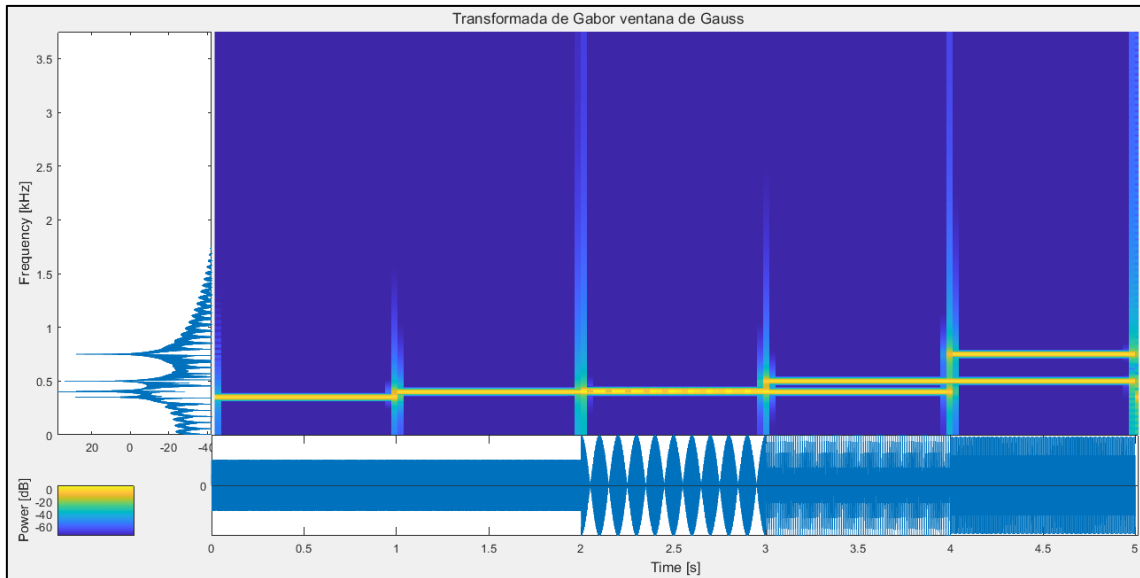


Figura 12. Transformada de Gabor de la señal, mediante *DGTtool*.

2.3 IMPLEMENTACIÓN DE LA REASIGNADA DE FOURIER EN MATLAB

Al igual que para el caso anterior, en la Reasignada de Fourier será necesario emplear el toolbox “*Signal Processing Toolbox*”, donde existe el objeto destinado para este método: ***pspectrum***, este objeto permite realizar el proceso de reasignación de tiempo y frecuencia de una señal, esto lo consigue mediante la división de la señal en segmentos con una longitud igual, estos segmentos tienen que ser suficientemente cortos para que no exista perturbación en la frecuencia de la señal, además se colocan ventanas Kaiser (para eliminar datos redundantes) por separado y se calcula la STFT para finalmente concatenar las transformadas para formar una matriz y promediar los periodogramas de todos los segmentos.

A continuación, se presentan los parámetros empleados para la correcta utilización del objeto [31].

```
[P, F, T]=
pspectrum(signal,Fs,'FrequencyLimits',[0
50],'FrequencyResolution',1.5,'Leakage',0.85,'spectrogram','Reas
sign',true);
```

Donde los parámetros de entrada se los resumen en la Tabla 3.

Tabla 3. Parámetros pspectrum [31]

PARÁMETRO	DESCRIPCIÓN
Signal	Señal de entrada.
Fs	Representa la frecuencia de muestreo, 10^* (frecuencia respecto más exigente).
Spectrogram	Este parámetro calcula el espectrograma de entrada para analizar la variación del contenido de frecuencia en un determinado tiempo.
FrequencyLimits	Este es un vector numérico que consta de 2 valores los cuales representan los límites de frecuencia de visualización. Por ejemplo: [0 50] Hz.
Leakage	La fuga espectral, es un escalar numérico real que va del 0 al 1. 'Leakage', busca el mejoramiento de la resolución y disminuir la fuga, Por ejemplo: ['Leakage',0.85] donde 0.85 representa la Ventana de Hann.
TimeResolution	Este argumento se encarga de controlar la duración de los segmentos empleados para utilizados para calcular los espectros de potencia de tiempo corto que forman estimaciones de espectrograma. 'TimeResolution' se expresa en segundos y es un valor real que posee información de tiempo, o como un número entero de muestras.
FrequencyResolution	Al igual que el argumento anterior, este hace referencia al ancho de banda de resolución de frecuencia, este se expresa en hercios como un valor escalar real.
Reassign	Si se establece esta opción como true, pspectrum agudiza la localización de las estimaciones espectrales al realizar la <i>reasignación de tiempo y frecuencia</i> .

Para el uso del objeto en las pruebas se plantea la siguiente configuración de los parámetros de entrada:

- **yt1:** representa la señal compuesta por los cinco componentes.
- **Fs:** frecuencia de muestreo.

- **FrequencyLimits:** es un parámetro que se asemeja a “axis”, lo que permite es delimitar el eje de las frecuencias para centrar los resultados, esto en forma de vector. Para las pruebas se emplea el valor de la frecuencia más alta “f5” sumado 50 Hz con el fin de permitir observar de forma clara los componentes en el eje espectral.
- **FrequencyResolution/TimeResolution:** Estos parámetros permiten mejorar la resolución ya sea en tiempo o en frecuencia. De la Tabla 3 se define que este parámetro es un número escalar positivo por lo que, para mejor resolución en tiempo y frecuencia el valor óptimo y el cual mostró mejores resultados es 0.1. Es importante mencionar que no se puede emplear ambos parámetros de forma simultánea por lo que se divide tanto los resultados en tiempo como en frecuencia.
- **Leakage (fuga espectral):** Para obtener los mejores resultados en tiempo. se empleó un valor de leakage igual a 1, lo que equivale a aplicar una ventana rectangular. Para el caso de la resolución en frecuencia el leakage adecuado y el que se empleó es 0, esto debido a que este valor reduce al mínimo la dispersión espectral. Adicional el valor que permite tener un equilibrio tanto en resolución tiempo como en frecuencia es 0.85, el cual aplica una ventana Hann a los datos de forma que hay una armonía en la resolución.
- **spectrogram:** Al asignar este tipo se procede a calcular el espectrograma de la señal de entrada.
- **Reassign | true:** Al asignar la opción “true” a este parámetro se enfatiza el método de Reasignación para así tener una localización más precisa de los datos tanto en frecuencia como en tiempo.

Para el caso de los parámetros o argumentos de salida se tiene lo siguiente:

- **P:** contiene una estimación de los espectros de potencia de las distintas ubicaciones de la ventana deslizante
- **F:** es el vector de frecuencias.
- **T:** es el vector de tiempos en los cuales se fue ubicando la ventana deslizante.

Finalmente, para la visualización de los resultados se hace uso de la función “mesh” el cual permite crear la gráfica tridimensional con los valores de P, F y T. Para obtener los

resultados en dos dimensiones se aplica la función “*view(2)*” lo cual permite conseguir la gráfica con el eje “y” (frecuencia) y eje “x” (tiempo).

A continuación, se muestra el segmento de código empleado con el objeto *pspectrum* para las pruebas con la Reasignada de Fourier y la obtención de resultados tanto para resolución en tiempo como en resolución en frecuencia.

```
%% Para Frecuencia
[P, F, T]=pspectrum(yt1,Fs,'FrequencyLimits',[0
50],'FrequencyResolution',1.5,'Leakage',0.85,'spectrogram','Reas
sign',true);
mesh(seconds(T),F,P)
colormap parula
title('ESPECTROGRAMA REASIGNADO EN FRECUENCIA')
xlabel('Tiempo')
ylabel('F(Hz)')
axis tight
view(2)

%% Para Tiempo
figure(4)
[P, F, T]=pspectrum(yt1,Fs,'FrequencyLimits',[0
50],'TimeResolution',1,'Leakage',0.85,'spectrogram','Reassign',t
rue);
mesh(seconds(T),F,P)
title('ESPECTROGRAMA REASIGNADO EN TIEMPO')
xlabel('Tiempo')
ylabel('F(Hz)')
axis tight
view(2)
xlabel('Tiempo')
ylabel('F(Hz)')
axis tight
view(2)
```

La Figura 13 muestra un ejemplo del funcionamiento del objeto *pspectrum* en el análisis de señales en el dominio tiempo - frecuencia. Para esto se hizo uso de la señal

multicomponente con frecuencias: 350, 400, 435, 500 y 750 Hz. A continuación, las gráficas obtenidas.

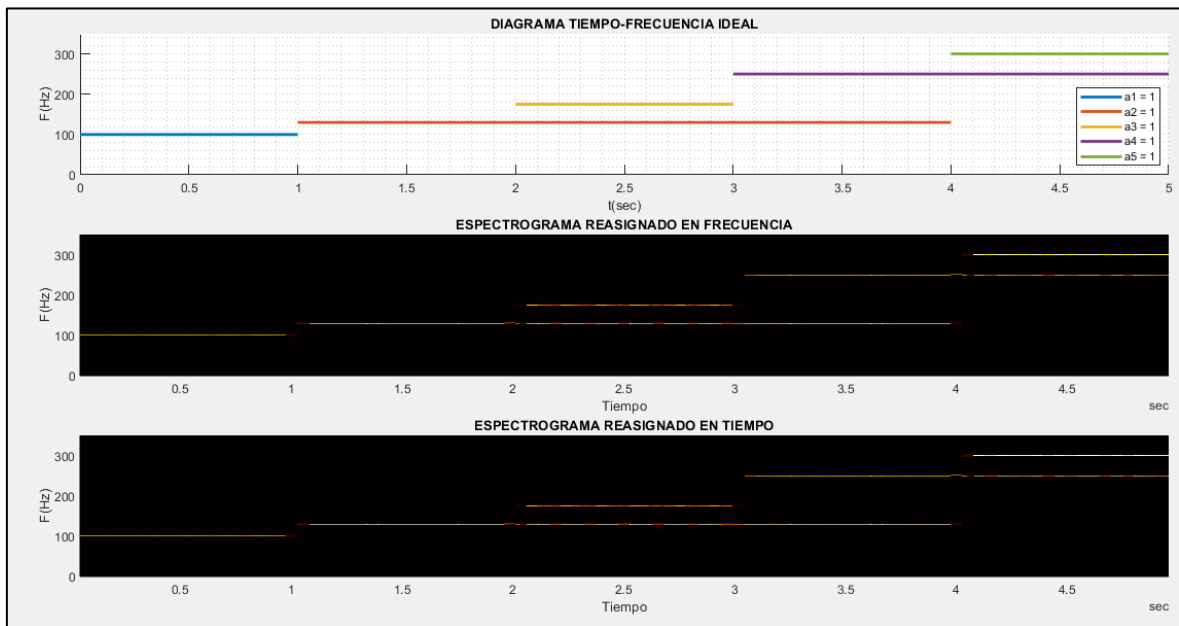


Figura 13. Diagramas tiempo-frecuencia empleando Reasignada de Fourier con resolución en frecuencia y en tiempo comparados con el diagrama tiempo-frecuencia ideal

2.4 IMPLEMENTACIÓN DE LA SINCRONIZADA DE FOURIER EN MATLAB

Para las pruebas realizadas con el método de la Sincronizada de Fourier se emplea nuevamente el Toolbox “*Signal Processing Toolbox*”, donde para este caso posee un objeto específico *fsst*. Este objeto realiza el proceso matemático establecido para el método, teniendo una mayor nitidez en los resultados espectrales y además se agudiza los resultados temporales, esto debido a que el objeto entrega un vector de tiempo el cual posee una longitud igual al número de columnas de la matriz correspondiente a la sincronizada resultante de la señal. A continuación, se muestra un ejemplo de la utilización de los parámetros del objeto.

```
[d, f, t]=fsst(xn, Fs, window)
```

Las opciones para los parámetros tanto de entrada como de salida se los tiene en la Tabla 4.

Tabla 4. Parámetros fsst [32].

PARÁMETRO	DESCRIPCIÓN
d	Este es el resultado de la transformada sincronizada de Fourier, devuelta como matriz
f	Es la frecuencia cíclica, devuelta como vector. La longitud de f es igual al número de filas de la matriz “d”.
t	Representa los instantes de tiempo, devueltos como un vector. La longitud de t es igual al número de columnas en “d”. Cada valor de tiempo en t es el punto medio de un segmento de ventana de la señal “xn”.
xn	Representa la señal de entrada.
Fs	Representa la frecuencia de muestreo, 10*(frecuencia respecto más exigente).
window	Este parámetro hace referencia a la ventana empleada para realizar el proceso de división en segmentos.

El siguiente código fue empleado para la obtención de los resultados en la prueba 1:

```
% ANALISIS EN T-F CON FSST
%MEJOR RESOLUCION EN TIEMPO
window_T=rectwin(fix(length(xn)/2));
[d_T, f_T, t_T] = fsst(xn,Fs,window_T);

figure

contour(t_T,f_T,abs(d_T));
title('FSST CON RESOLUCION EN TIEMPO')
xlabel('t(sec)')
ylabel('F(Hz)')
axis([0 5 0 350])
grid minor
%MEJOR RESOLUCION EN FRECUENCIA
window_F=blackmanharris(fix(length(xn)/5));
[d_F, f_F, t_F] = fsst(xn,Fs,window_F);

figure
```

```
contour(t_F, f_F, abs(d_F));  
title('FSST CON RESOLUCION EN FRECUENCIA')  
xlabel('t(sec)')  
ylabel('F(Hz)')  
axis([0 5 0 350])  
grid minor
```

Como se aprecia en el código empleado, como parámetro de entrada del objeto *fsst* se debe definir una ventana adecuada, es por tal motivo que se hace uso de las funciones adecuadas para la generación de ventanas en Matlab, que para el caso de mejor resolución en tiempo se tomó la ventana Rectangular (*rectwin*) y para el caso de la resolución en frecuencia la ventana adecuada es Blackman-Harris (*blackmanharris*) dado que con esta ventana se puede detectar las componentes de frecuencia muy pequeñas. .

- Para la resolución en tiempo se toma a la ventana de Rectangular del 20% de puntos de la señal total, en otras palabras, se toma la longitud total de la señal y se divide para 5.
- Para la resolución en frecuencia se toma a la ventana de BlackmanHarris del 50% de puntos de la señal, en otras palabras, se toma la longitud total de la señal y se divide para 2.

En la graficación se emplea la función *contour*, debido a que esta presenta los resultados de manera más clara y nítida. Esta función toma los parámetros de salida de tiempo, frecuencia y para el caso del parámetro “*d*” se usa *abs* lo que permitirá extraer el módulo de los complejos de la Transformada Sincronizada resultante.

A continuación, se presenta en la Figura 14 un ejemplo del objeto *fsst* en el análisis de señales en el dominio tiempo - frecuencia con el método Transformada Sincronizada de Fourier comparado con el diagrama de tiempo - frecuencia ideal. Para esto se hizo uso de la señal multicomponente con frecuencias: 25, 40, 50, 75 y 90 Hz. A continuación, las gráficas obtenidas.

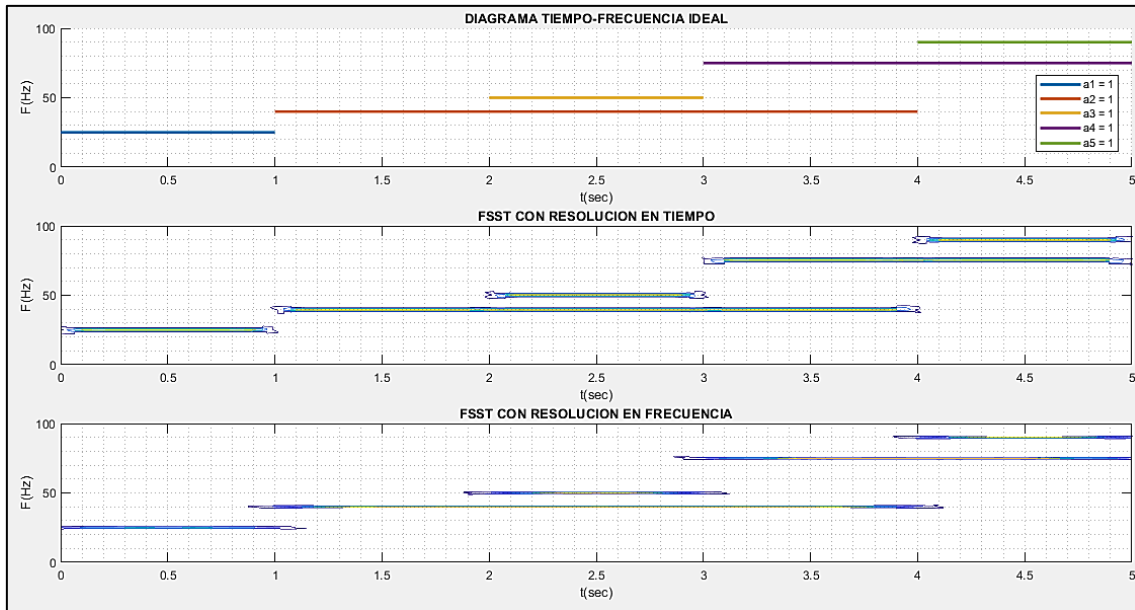


Figura 14. Diagramas tiempo-frecuencia empleando la Transformada Sincronizada de Fourier con resolución en frecuencia y en tiempo comparados con el diagrama tiempo-frecuencia ideal.

2.5 IMPLEMENTACIÓN DE LOS SUPERLETS EN MATLAB

En Matlab no existe una función directa para obtener los Superlets, por lo que se hace uso de 2 funciones desarrolladas por *Harald Bärzan* [33], dichas funciones son: **aslt** (Adaptive Superlets) el cual proporciona una mejor resolución en tiempo en todo el espectro y **faslt** (Adaptive fractional Superlets) el cual proporciona representaciones más nítidas en todo el dominio de frecuencia. Las funciones antes mencionadas se encuentran en la sección *Anexos*, Anexo I (ASLT) y Anexo II (FASLT).

Como cualquier otra función de Matlab esta emplea parámetros de entrada los cuales se definen en el siguiente ejemplo segmento de código:

```
% tiempo
RT_SL = aslt(input, Fs, fois, Nc, Ord, mult);
```

Donde las opciones para dichos parámetros están en la Tabla 5.

Tabla 5. Parámetros aslt y faslt [32].

PARÁMETRO	DESCRIPCIÓN
xn	Representa la señal de entrada.

Fs	Representa la frecuencia de muestreo, 10^* (frecuencia respecto más exigente).
fois	Representa un buffer con las frecuencias de interés
Nc	Es el valor del número ciclos iniciales de wavelets que se desea.
Ord	Es un intervalo del orden de super resolución (dato opcional)
Mul	Es un número que puede ser cero o distinto de cero, para especificar el uso de super resolución aditiva o multiplicativa: 0 para aditiva, distinto de cero para multiplicativa.

Con estos datos, la función tanto para ASLT como para FASLT, operan obteniendo su transformada de wavelet para producir una representación en el dominio tiempo-frecuencia. Para cada frecuencia de interés, se selecciona el orden entero más cercano del intervalo para producir cada Superlet. Este Superlet es un conjunto de wavelets que poseen la misma frecuencia central, pero diferente número de ciclos.

Una vez realizado todo este proceso, para las frecuencias de interés, la función retorna el espectro del Superlet que se lo grafica con el tiempo de muestreo y el buffer de las frecuencias establecidas.

Para el uso de cada función se estable los parámetros de entrada los cuales se describen en la Tabla 5 antes descrita y cuyos valores representan la necesidad de cada prueba planteada. Es decir, para el caso del buffer de frecuencias (*fois*) se deberá modificar dependiendo del requerimiento, esto de igual manera en el vector de ciclos de tiempo (*srord*) y frecuencia (*srord2*).

Adicional en el caso de los Superlets se empleó una representación visual diferente a los otros métodos, esta representación se la consiguió al utilizar la función *imagesc*, la cual toma los parámetros antes descritos y la transformada resultante (*aslt* o *faslt*) imprime una imagen más nítida con una gama de colores más exactos y proporcionales para representar cada componente.

A continuación, se presenta el código empleado para el uso de las funciones *aslt* y *faslt*:

```
%% Superlets
```



```

fois = 0.1:0.1:50; %vector buffer para el rango de las
frecuencias
srord= [1, 30]; %vector de ciclos para tiempo
srord2= [1, 50]; %vector de ciclos para frecuencia
% tiempo
RT_SL = aslt(xn, Fs, fois, 5, srord, 0);
% frecuencia
RF_SL = faslt(xn, Fs, fois, 8, srord2, 0);

%Graficamos la resolucion en tiempo Superlets
figure
imagesc(t,fois,RT_SL);
set(gca, 'ydir', 'normal');
colormap jet;
axis([0 5 0 50]);
grid on;
xlabel('TIEMPO (segundos)');
title('SUPERLETS CON RESOLUCIÓN EN TIEMPO');
xlabel('t(sec)');
ylabel('F(Hz)');

%Graficamos la resolucion en frecuencia Superlets
figure
imagesc(t,fois,RF_SL);
set(gca, 'ydir', 'normal');
colormap jet;
axis([0 5 0 50]);
grid on;
xlabel('TIEMPO (segundos)');
title('SUPERLETS CON RESOLUCIÓN EN FRECUENCIA');
xlabel('t(sec)');
ylabel('F(Hz)');

```

La Figura 15 representa un ejemplo del uso de las funciones *aslt* y *faslt* en el análisis de señales en el dominio tiempo - frecuencia con el método Superlets comparado con el

diagrama de tiempo - frecuencia ideal. Para esto se hizo uso de la señal multicomponente con frecuencias: 25, 40, 50, 75 y 90 Hz. La Figura 15 muestra los resultados obtenidos.

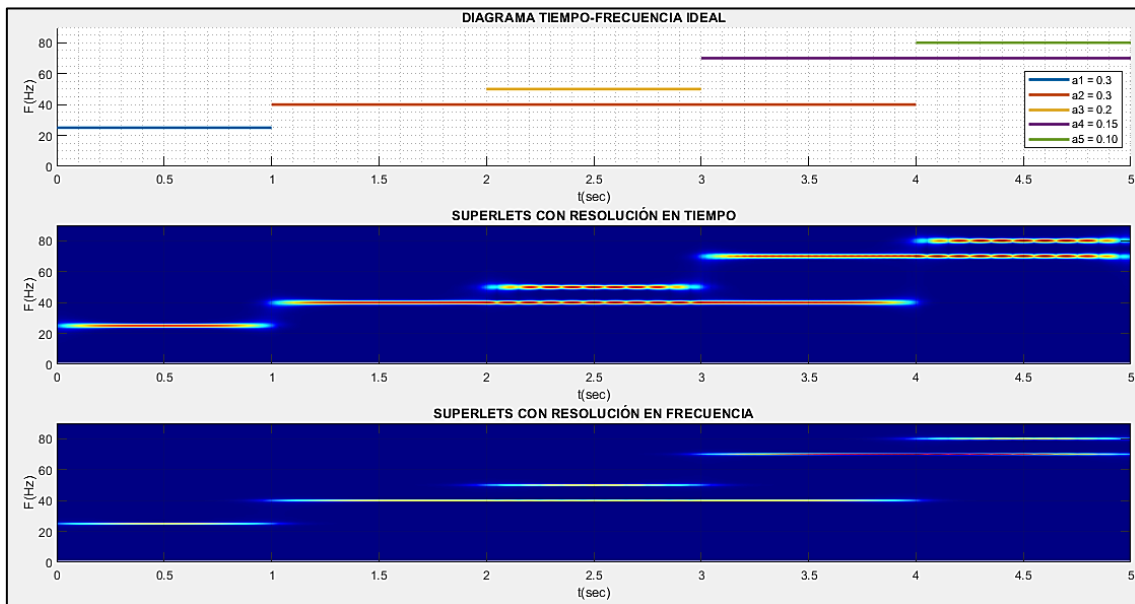


Figura 15. Diagramas tiempo-frecuencia empleando la Superlets con resolución en frecuencia (FASLT) y en tiempo (ASLT), comparados con el diagrama tiempo-frecuencia ideal.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

Este capítulo está destinado a la discusión de los resultados obtenidos con cada uno de los métodos tratados, haciendo énfasis en la resolución temporal, resolución en frecuencia, capacidad de detección de componentes muy pequeñas y carga computacional, teniendo como base el diagrama tiempo-frecuencia ideal.

3.1 RESULTADOS

3.1.1 COMPARACIÓN DE MÉTODOS - PRUEBA 1

En el caso de la resolución en frecuencial el método que más se acerca al diagrama ideal (Figura 16) es los SUPERLETS (Figura 17 - d). Por otro lado, el método que mostró excelentes resultados fue la Reasignada de Fourier (Figura 17 - b) cuyo principio se basa en la reasignación de las frecuencias vecinas para obtener los mejores resultados espectrales.

Por otro lado, la transformada Reasignada de Fourier obtiene resultados que se aproximan mucho, tanto en los valores de frecuencia como en tiempos de duración, al diagrama tiempo-frecuencia ideal.

El método que muestra mayor dispersión en los resultados es la Transformada de Gabor donde existe ensanchamiento de las frecuencias de cada componente (Figura 17 - a). Por lo anterior, este método es el que obtuvo peor resolución en frecuencia.

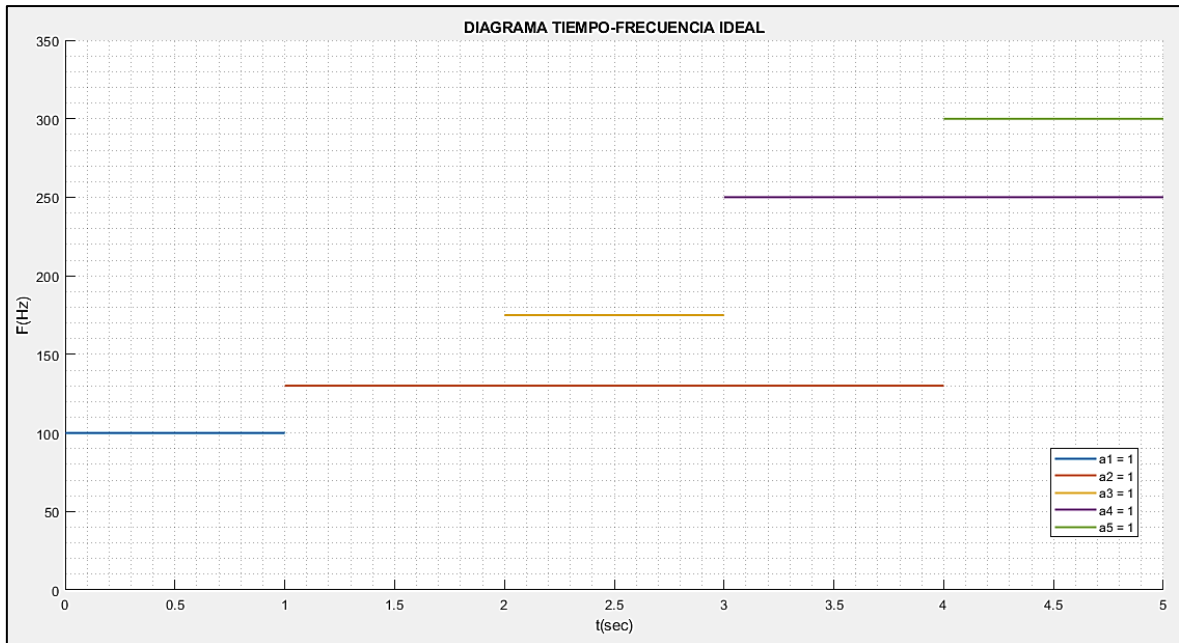


Figura 16. Diagrama tiempo-frecuencia ideal para la Prueba 1.

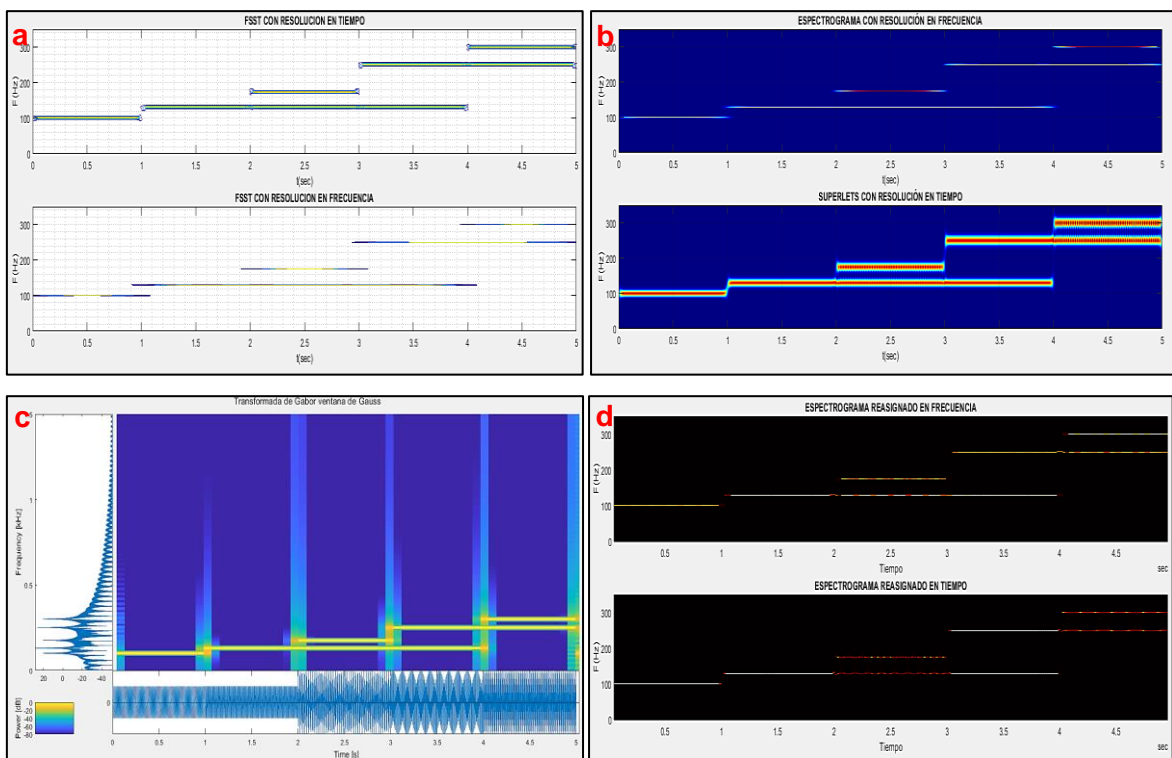


Figura 17. Resultados obtenidos en la Prueba 1. (a) Transformada de Gabor, (b) Reasignada de Fourier, (c) Sincronizada de Fourier y (d) Superlets.

Para el análisis respecto de la duración, 3 de los 4 métodos poseen excelentes resultados: la Reasignada de Fourier, Sincronizada de Fourier y Superlets, es decir, se acercan más a los resultados esperados dados por el diagrama tiempo-frecuencia ideal anteriormente citado.

El método Transformada de Gabor tampoco obtuvo buenos resultados respecto de la resolución en tiempo, ya que para ninguna componente existe exactitud en los valores asignados en duración.

3.1.2 COMPARACIÓN DE MÉTODOS - PRUEBA 2

Los resultados obtenidos en la experimentación con la prueba 2 muestran como al tener un rango de frecuencias bajas y una amplitud fija, la dispersión tanto en resolución en tiempo como en frecuencia disminuye. Para el caso de la resolución en tiempo y frecuencia el método cuyos resultados se asemejan a lo ideal es los Superlets, donde se logra distinguir de forma clara cada componente y la duración específica (Figura 19 - d).

Con respecto a la Sincronizada de Fourier (Figura 19 - c) se evidencia como este método permite visualizar la duración casi exacta de cada componente con un mínimo de distorsión. Por otro lado, en la Reasignada de Fourier se observa las 5 componentes, no obstante, no se logra apreciar continuamente la duración de estas, por lo que existe pérdida de información en la parte inicial y final de la duración de cada componente (Figura 19 - b)

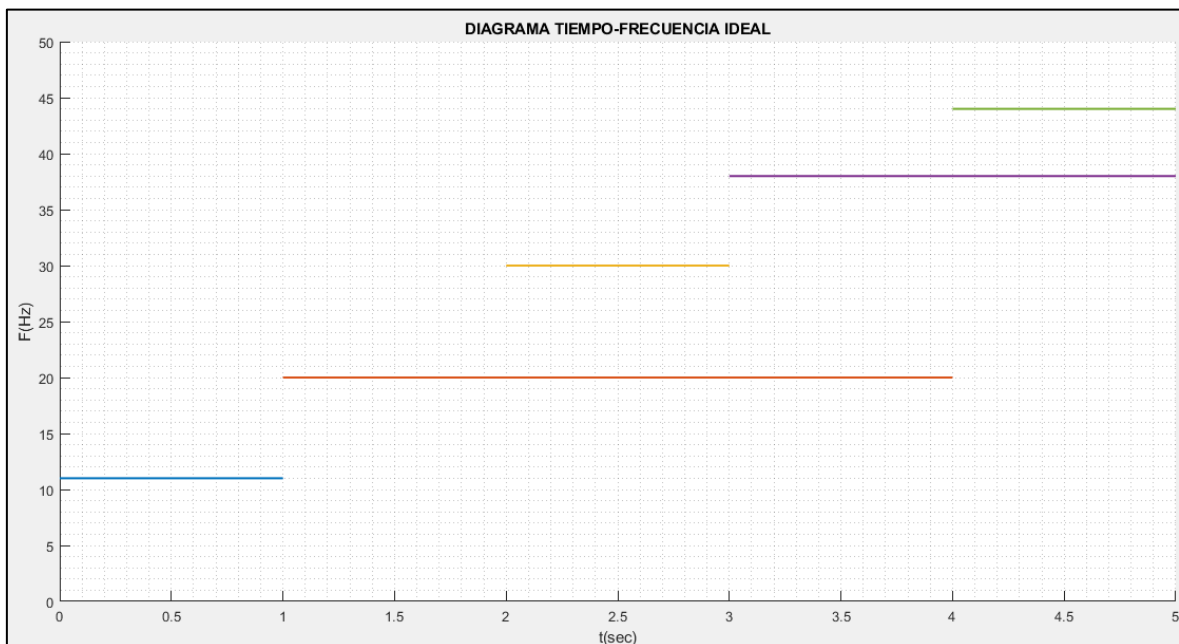


Figura 18. Diagrama tiempo-frecuencia ideal para la Prueba 2.

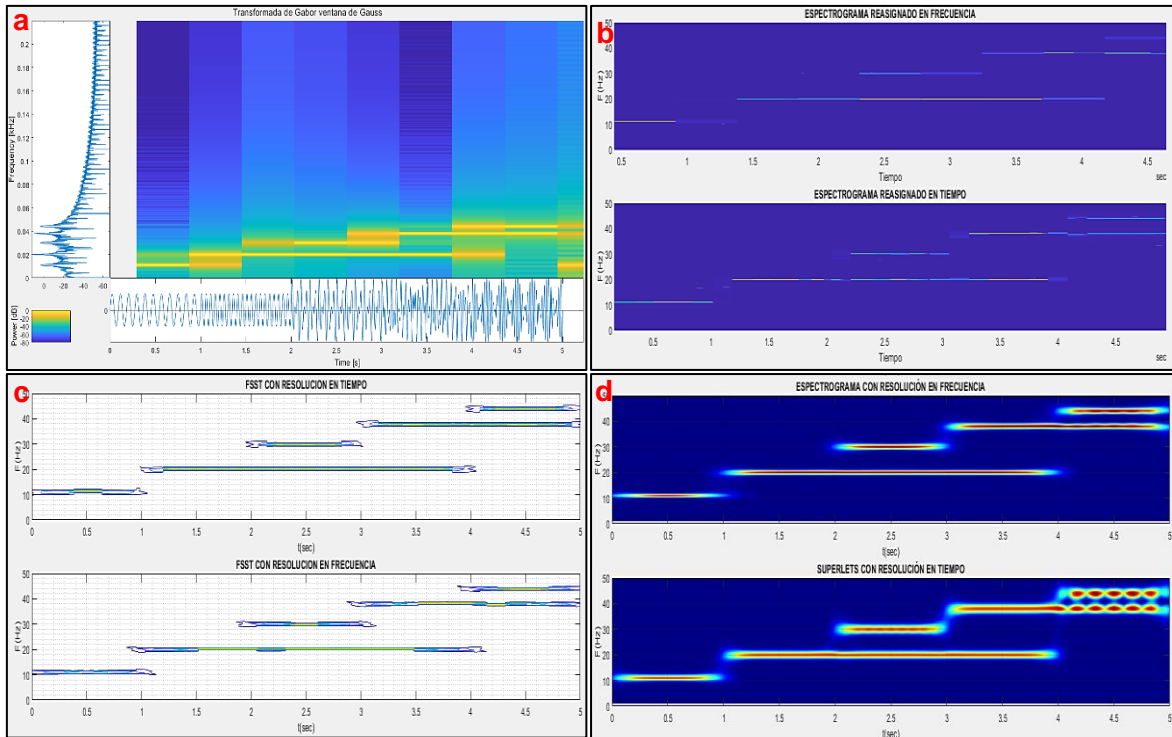


Figura 19. Resultados obtenidos en la Prueba 2. (a) Transformada de Gabor, (b) Reasignada de Fourier, (c) Sincronizada de Fourier y (d) Superlets.

Por otro lado, la Transformada de Gabor muestra como no existe ninguna aproximación a la duración establecida en cada componente, este problema se replica en la resolución en frecuencia donde se tiene ensanchamiento y variación en los resultados (Figura 19 - a), es por tal observación que este método posee los resultados más deficientes en comparación a los otros métodos.

3.1.3 COMPARACIÓN MÉTODOS - PRUEBA 3

En el caso del escenario 3 se tomó diferentes frecuencias a las pruebas anteriores, con un rango de [80 - 235] Hz y diferentes amplitudes que van desde 0.001 hasta 1. Esta prueba fue considerada para analizar la detección componentes muy pequeños en los diferentes métodos. De esto se obtuvo que para la detección de los componentes la gran mayoría de métodos no cumplieron con la totalidad del objetivo.

El método cuyos resultados muestran 3 componentes de los 5 fue la Reasignada de Fourier (Figura 21 - b), donde para mayor detalle de los componentes se modificó la configuración para la representación gráfica obteniendo mayor claridad en los resultados en comparación a los otros métodos. De esto se tiene que hay un mínimo de dispersión para la resolución en tiempo y frecuencia, dando un rango de error bajo.

Por otro lado, los resultados con la Transformada de Gabor (Figura 21 - a) muestran que este método logra detectar los 5 componentes, no obstante, se evidencia la dispersión y ensanchamiento tanto en los componentes con mayor y menor amplitud.

Es así como en esta prueba se pudo observar cómo los componentes con mayor nitidez y detalle fueron los 3 últimos componentes, cuya configuración de amplitud fue de 0.1 y 1; esto quiere decir que, para el caso de los objetos y funciones de Matlab, a mayor cercanía de 1 en amplitud la detección de componentes tendrá mejores resultados.

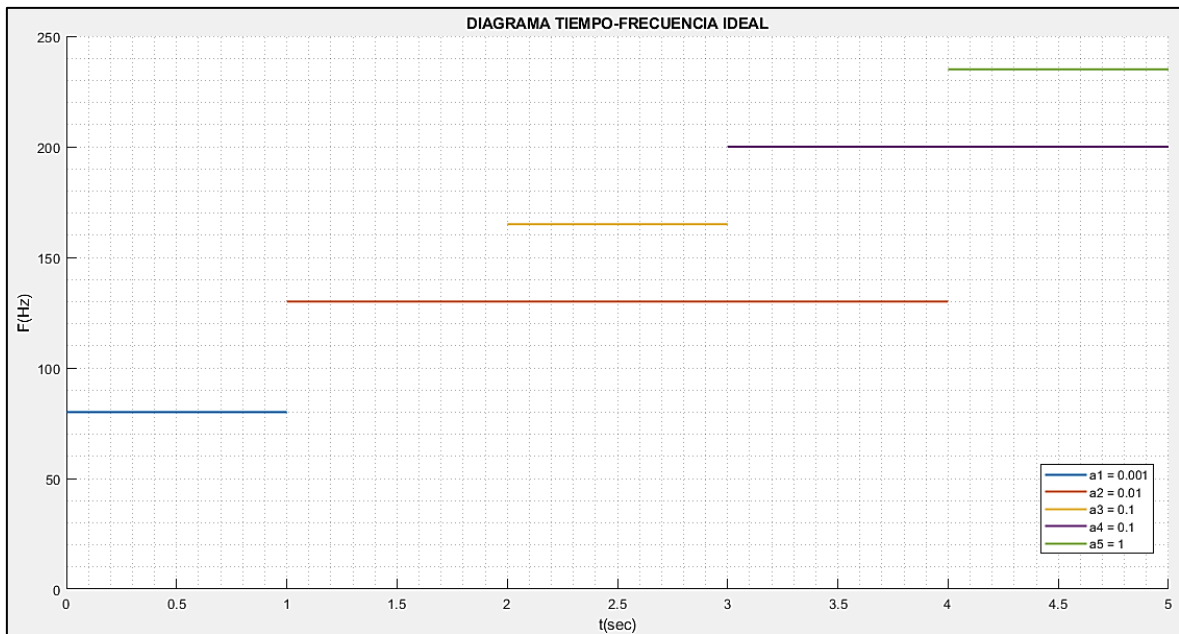


Figura 20. Diagrama tiempo-frecuencia ideal para la Prueba 3

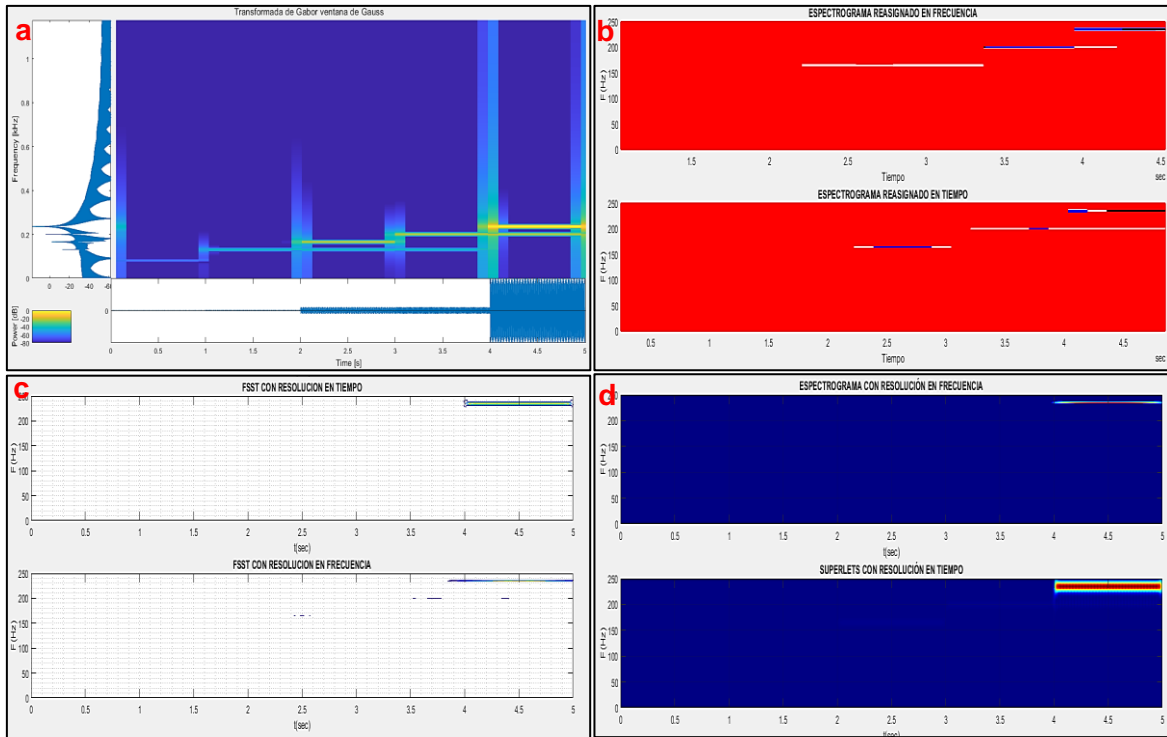


Figura 21. Resultados obtenidos en la Prueba 3. (a) Transformada de Gabor, (b) Reasignada de Fourier, (c) Sincronizada de Fourier y (d) Superlets.

3.1.4 RESULTADOS OBTENIDOS CON EL MÉTODO TRANSFORMADA DE GABOR

De la experimentación realizada se evidenció que este método posee una baja representación de los componentes. Si bien este método logró detectar la mayoría de estos en las diferentes pruebas, los resultados muestran que existe dispersión, además de ensanchamiento tanto en el eje temporal como espectral.



Figura 22. Zoom de resultados en la prueba 3, empleando la Transforma de Gabor

Como se muestra en la Figura 22 existe una mayor dilatación y alargamiento en los resultados tanto en el eje temporal (sección A), como en el eje espectral (sección B), esta

dispersión es más notoria en el eje de frecuencias donde se observa que existe un alargamiento total de cada componente. Estos resultados se muestran constantes en toda la experimentación, es decir, para las pruebas realizadas se obtuvo las mismas observaciones del ensanchamiento y poca precisión en la detección de componentes, en resolución en tiempo y en resolución en frecuencia.

Por otra parte, una ventaja que tuvo este método fue el tiempo computacional, esto se ve reflejado en la Tabla 6 donde para cada prueba el rango de tiempo no llegó a 1 segundo. Además, este método en su representación gráfica muestra no solo el diagrama tiempo - frecuencia, sino también la señal analizada y los picos de frecuencia, esta representación conjunta permite tener un mayor enfoque en la experimentación.

La Tabla 6 presenta los tiempos obtenidos para cada una de las pruebas empleando el objeto DGTtool y la función tic toc de Matlab para el método Transformada de Gabor.

Tabla 6. Tiempos resultantes de las diferentes pruebas utilizando la Transformada de Gabor

	Prueba 1	Prueba 2	Prueba 3
Duración (segundos)	0.039	0.036	0.833

De acuerdo a la Tabla 6 este método proporciona una menor carga computacional, debido a la poca exigencia en sus parámetros y en su representación visual, que en promedio es 0.3 segundos. Además de las pruebas planteadas, este método mostró que puede analizar señales con frecuencias por encima de los 500 [Hz] y con un tiempo aceptable de respuesta. Por lo anterior, si bien los resultados no fueron los mejores en resolución tiempo - frecuencia, este método proporciona un tiempo de procesamiento bajo en comparación de los otros métodos.

3.1.5 RESULTADOS OBTENIDOS CON EL MÉTODO REASIGNADA DE FOURIER

Los resultados obtenidos con este método prueba que, por un lado, en la prueba 1, al tener frecuencias altas no existió una exigencia computacional elevada teniendo no solo una buena resolución en tiempo sino también en frecuencia. Asimismo en esta prueba se evaluó la detección de componentes muy pequeñas, donde la calidad de resultados y la localización de los componentes se aproximaron a lo ideal.

Para la prueba 2 los resultados se replican a los de la prueba 1, donde se presenta una mayor exactitud en los componentes y en la duración asignada, con un error muy cercano a cero entre la experimentación y el diagrama tiempo - frecuencia ideal.

En el caso de la prueba 3 se presentó una disminución en la calidad de los resultados, esto se debió a que el método de reasignación pierde información al tener amplitudes relativas muy bajas. Por lo anterior, este método no mostró resultados con una dispersión pronunciada para amplitudes superiores a 0.1. En la Figura 23 se expone que el componente analizado C4, cuya frecuencia establecida es 200 [Hz] y la duración inicial es 3 segundos, no presenta una diferencia elevada entre los valores ideales y experimentales.



Figura 23. Zoom de resultados en la prueba 3, empleando la Reasignada de Fourier (Resultados en resolución en tiempo y frecuencia)

Una característica relevante del objeto empleado para la Reasignada de Fourier, es que este posee diferentes parámetros que permiten mejorar los resultados tanto en frecuencia como en tiempo, esto permite tener un mayor enfoque de cada resolución deseada, dichos parámetros son: “*TimeResolution*” y “*FrequencyResolution*”. En este método también fue necesario establecer una escala de color adecuada para cada prueba, esto se lo realizó configurando la función “*colormap*” teniendo una escala de colores apropiada para cada prueba.

La Tabla 7 presenta los tiempos obtenidos para cada una de las pruebas empleando el objeto pspectrum y la función tic toc de Matlab para el método Reasignada de Fourier.

Tabla 7. Tiempos resultantes de las diferentes pruebas utilizando la Reasignada de Fourier

	Prueba 1	Prueba 2	Prueba 3
Duración (segundos)	0.102	0.012	0.065

De los resultados obtenidos al emplear la función tic toc, se puede comentar que el objeto utilizado para la Reasignada de Fourier tiene una carga computacional aceptable, con un promedio de 0.059 segundos en el análisis de la señal multicomponente, además este

método tuvo una diferencia de algunos segundos para la impresión de los resultados gráficos. Es por tal razón que, para las pruebas realizadas, este método mostró excelentes resultados tanto en resolución en tiempo como en resolución en frecuencia, con una alta detección de componentes con amplitudes bajas y frecuencias cercanas, acercándose mucho al diagrama de tiempo-frecuencia ideal.

3.1.6 RESULTADOS OBTENIDOS CON EL MÉTODO SINCRONIZADA DE FOURIER

El método Sincronizada de Fourier mostró una serie de resultados que pusieron en evidencia que este no posee una carga computacional aceptable para frecuencias que sobrepasan los 100 [Hz], esto se reflejó en la prueba 1 donde al utilizar el objeto correspondiente “fssf”, el tiempo desde que el programa inicia hasta que culmina es elevado en comparación de los otros métodos.

Para el caso de la resolución en tiempo y en frecuencia, en la prueba 2 se mostraron buenos resultados respecto de la precisión en la duración de cada componente. De manera análoga los resultados en resolución en frecuencia muestran que también se acercan al resultado ideal.

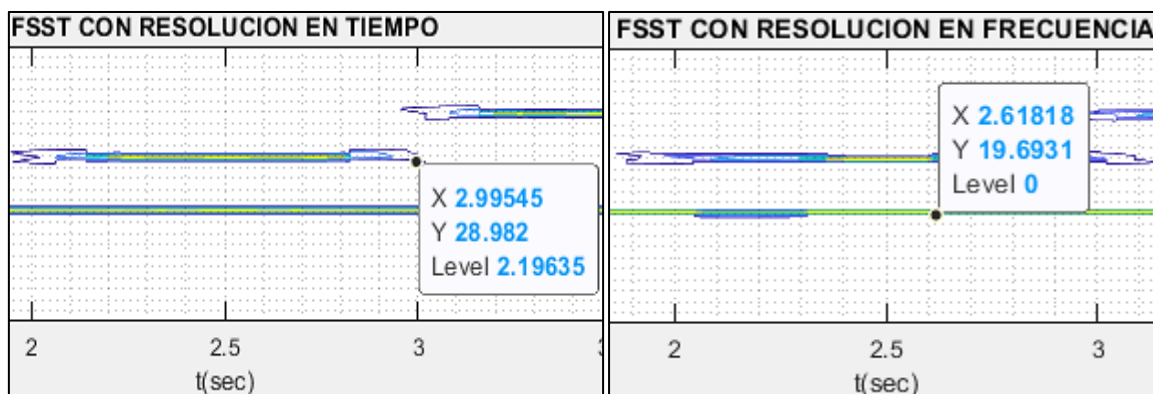


Figura 24. Zoom de resultados de la prueba 2, empleando la Sincronizada de Fourier (Resultados en resolución en tiempo y frecuencia)

En la Figura 24 se muestra que para el caso de la resolución en tiempo, la componente seleccionada tiene una duración de [2 - 3] segundos, donde al emplear el método y la señal compuesta se tiene una diferencia mínima de 0.004 segundos. De igual manera en la resolución en frecuencia la componente analizada (C2) posee una frecuencia de 20 [Hz] y la frecuencia resultante en la experimentación fue 19.693 [Hz], demostrando que existe un diferencia mínima al emplear este método.

Para el caso de la prueba 3 este método mostró que únicamente puede detectar los componentes con amplitudes cercanas o igual a 1, esto se debe al principio de funcionamiento del método el cual elimina las componentes bajas teniendo pérdida de información para el caso de señales con amplitudes pequeñas.

Una desventaja para el objeto correspondiente a la Sincronizada de Fourier es que este no posee un parámetro que permita mejorar la nitidez o calidad en los resultados, por lo que únicamente se ingresa la señal, frecuencia de muestreo y ventana.

La Tabla 8 presenta los tiempos obtenidos para cada una de las pruebas empleando el objeto *fsst* y la función *tic toc* de Matlab para el método Sincronizada de Fourier.

Tabla 8. Tiempos resultantes de las diferentes pruebas utilizando la Sincronizada de Fourier

	Prueba 1	Prueba 2	Prueba 3
Duración (segundos)	2.405	0.996	2.949

Como se observa en la Tabla 8, existe una mayor duración para las pruebas cuyas frecuencias superan los 100 [Hz], esto demuestra que el este método tiene mayor requerimiento computacional. Lo cual se nota para frecuencias elevadas. En promedio la duración de la experimentación utilizando “*fsst*” fue 2.116 segundos demostrando que existen limitaciones para el uso de este objeto.

3.1.7 RESULTADOS OBTENIDOS CON EL MÉTODO SUPERLETS

Se debe partir de que las funciones utilizadas en Matlab no son propias de un toolbox, sino que se empleó funciones (scripts) desarrollados por un autor externo. Partiendo de lo anterior se empleó dos funciones “*aslt*” y “*faslt*” donde cada función se encuentra diseñada para una mejor resolución en tiempo o en frecuencia respectivamente. Al realizar la respectiva configuración de cada parámetro de entrada se evidenció que este método obtuvo los mejores resultados en detección de componentes muy pequeñas, en resolución en tiempo y en resolución en frecuencia, en comparación a los demás.

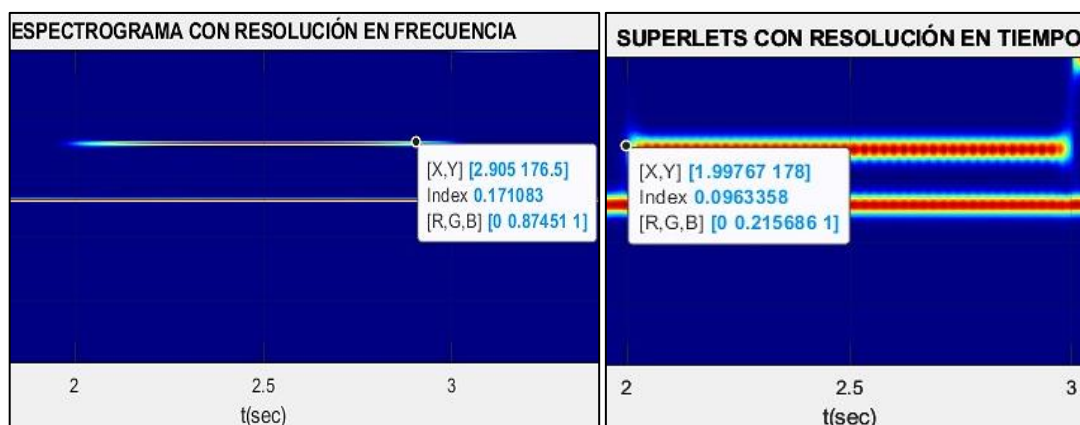


Figura 25. Zoom de resultados en la prueba 1, empleando la Superlets (Resultados en resolución en tiempo y frecuencia)

En la prueba 1 se obtuvieron resultados muy exactos en la detección de cada componente, para el caso de la componente de duración 2 segundos, se observa en la Figura 25 que se obtuvo 1.997 segundos. Esto se replica para la prueba 2 donde, al analizar la frecuencia de una componente al azar, se tiene que la función faslt logra afinar los resultados.

Para el caso de la prueba 3, este método responde igual que los otros, es decir, no logra detectar las componentes que están por debajo de 0.1.

A pesar de que los resultados en resolución y detección fueron los mejores, una desventaja de utilizar Superlets, es su alta carga computacional. Esto se evidenció en la Tabla 9, para las pruebas cuyas frecuencias se establecen sobre los 100 [Hz].

Tabla 9. Tiempos resultantes de las diferentes pruebas, utilizando Superlets

	Prueba 1	Prueba 2	Prueba 3
Duración (segundos)	7.441	0.781	4.790

Al emplear la función tic toc de Matlab se obtuvo que, en promedio, el tiempo invertido al emplear las funciones aslt y faslt, fue de 9.818 segundos, tiempo que no considera la duración extra hasta la presentación gráfica, que supera 1 minuto.

3.1.8 RESUMEN GENERAL

En la Tabla 10 se realiza un resumen de los resultados más relevantes de cada método, detallando las características obtenida en la experimentación.

Tabla 10. Resumen comparativo de los resultados obtenidos respecto de la resolución en frecuencia, resolución en tiempo y carga computacional

Método	Resolución Temporal	Resolución en frecuencias	Carga Computacional
Transformada de Gabor	Mala resolución en tiempo, no se logra distinguir duraciones exactas de cada componente.	Mala resolución, existe una dispersión considerable, las líneas de frecuencia muestran ensanchamiento y no hay una distinción exacta.	Se tiene un tiempo muy bajo, lo que se considera una característica positiva al momento de obtener resultados gráficos.
Reasignada de Fourier	Excelente respuesta en los tiempos, teniendo correctamente reflejado la duración asignada a cada componte.	Excelente resolución en frecuencia, se presenta resultados mucho más nítidos y afinados para todos los componentes.	Posee una carga computacional buena tanto para frecuencias pequeñas como elevadas.
Sincronizada de Fourier	Presenta muy buenos resultados en tiempo, teniendo una pequeña brecha con los resultados ideales.	Existe un ensanchamiento en las líneas de frecuencia obtenidas, no obstante, este alargamiento no presenta una diferencia elevada.	Alta carga computacional para frecuencias elevadas. Para el caso de pruebas con frecuencias bajas no presenta problema.
Superlets	Excelentes resultados, se puede apreciar de manera más clara las limitaciones en	Excelentes resultados, al emplear FASLT se obtiene un	Alta carga computacional para todos los casos de evaluación.

	la duración de cada componente.	afinamiento en las frecuencias.	
--	---------------------------------	---------------------------------	--

La Tabla 11 presenta un resumen sintetizado de los resultados obtenidos al emplear los cuatro métodos con las diferentes pruebas.

Tabla 11. Resumen comparativo de los resultados obtenidos respecto de la resolución en frecuencia, resolución en tiempo, carga computacional y capacidad de detección de componentes muy pequeñas.

Método	Resolución Temporal			Resolución en frecuencia			Carga Computacional			Detección de componentes muy pequeñas		
	Alta	Regular	Baja	Alta	Regular	Baja	Alta	Regular	Baja	Alta	Regular	Baja
Transformada de Gabor			X			X			X	X		
Reasignada de Fourier	X			X					X	X		
Sincronizada de Fourier	X				X			X			X	
Superlets	X			X			X			X		

3.2 CONCLUSIONES

Al realizar la Prueba 1, el método que reflejó mayor carga computacional frente a los otros fue el de los de Superlets. Por otra parte, el método con menor carga computacional es la Transformada de Gabor, este método demostró tener el menor tiempo de respuesta tanto en el análisis de la señal como en la representación gráfica de los resultados.

En la Prueba 2, el método que demostró tener una mejor resolución en tiempo y en frecuencia es el de los Superlets. Al emplear ASLT (Adaptive Superlets), se tuvo la mejor exactitud en la duración de cada componente. Al emplear la función FASLT (Fractional Adaptive Superlets), se obtuvieron componentes con una mejor localización en frecuencia. Por otra parte, el método cuya resolución en tiempo y en frecuencia se alejó de los

resultados ideales, fue la Transformada de Gabor, este método tuvo mayor dispersión en tiempo y en frecuencia.

La Prueba 3, diseñada para poner al límite la detección de componentes muy pequeñas, mostró que la Transformada de Gabor logra detectar los 5 componentes establecidos para esta prueba, pese a la baja resolución en tiempo y en frecuencia, dio buenos resultados de localización de componentes cuya amplitud está por debajo de 0.001. Otro método válido para este escenario es la Reasignada de Fourier la cual mostró 3 de los 5 componentes.

Si bien resultados obtenidos plantean como los métodos modernos reflejan un mejoramiento en la resolución en tiempo, resolución en frecuencia y detección de componentes muy pequeñas, cada método analizado mostró características positivas frente al otro. No obstante, no se puede asegurar que un método es mejor o peor que otro, esto debido al enfoque que se le asigne a cada uno, es así como la elección de un método dependerá del análisis o el problema específico a resolver.

3.3 RECOMENDACIONES

Ya que en este estudio se muestra un análisis general del uso de los métodos modernos en el dominio tiempo - frecuencia, se recomienda profundizar y desarrollar nuevas pruebas que pongan al límites las características de cada método.

Como se mostró en los resultados, cada método contempla una carga computacional diferente, por lo cual se recomienda verificar los requerimientos computacionales para el correcto funcionamiento de los scripts y pruebas planteadas.

4 REFERENCIAS BIBLIOGRÁFICAS

- [1] S. L. Marple, «Time-frequency signal analysis: Issues and alternative methods», *Proceedings of the IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis*, pp. 329-332, 1998, DOI: 10.1109/TFSA.1998.721427.
- [2] M. R. Canal, «Comparison of wavelet and short time Fourier Transform methods in the analysis of EMG signals», *J Med Syst*, vol. 34, n.º 1, pp. 91-94, feb. 2010, DOI: 10.1007/S10916-008-9219-8/METRICS.
- [3] X. Li *et al.*, «Comparison of different time-frequency analysis methods for sparse representation of PD-induced UHF signal», *China International Conference on Electricity Distribution, CICED*, vol. 2016-September, sep. 2016, DOI: 10.1109/CICED.2016.7575902.

- [4] Q. Zhu, Y. Wang, y G. Shen, «Research and comparison of time-frequency techniques for nonstationary signals», *J Comput (Taipei)*, vol. 7, n.º 4, pp. 954-958, 2012, DOI: 10.4304/JCP.7.4.954-958.
- [5] R. Alvarez, E. Borbor, y F. Grijalva, «Comparison of methods for signal analysis in the time-frequency domain», *2019 IEEE 4th Ecuador Technical Chapters Meeting, ETCM 2019*, nov. 2019, DOI: 10.1109/ETCM48019.2019.9014860.
- [6] Alfredo Rosado Muñoz, «Desarrollo de Técnicas de Detección de Fibrilación Ventricular Basadas en Algoritmos Tiempo-Frecuencia», Universitat de València, Valencia, 2000.
- [7] Samuel Case Bradford, «Time-Frequency Analysis of Systems with Changing Dynamic Properties», California Institute of Technology, California, 2006. Accedido: 5 de octubre de 2022. [En línea]. Disponible en: https://thesis.library.caltech.edu/4689/2/bradford_thesis_twoside.pdf
- [8] «Fourier Transform (FT) - Questions and Answers in MRI». <https://mriquestions.com/fourier-transform-ft.html> (accedido 5 de octubre de 2022).
- [9] R. H. Lara, R. Á. Rueda, y A. A. Pillajo, «Evaluación de las técnicas tiempo-frecuencia por medio de un equipo de adquisición de datos y un computador», *ACI Avances en Ciencias e Ingenierías*, vol. 3, n.º 1, pp. 33-39, dic. 2011, DOI: 10.18272/ACI.V3I1.60.
- [10] «Selección de la ventana temporal en la transformada de Fourier en tiempos cortos utilizada en el análisis de señales de vibración para determinar planos en las ruedas de un tren | Revista Facultad de Ingeniería Universidad de Antioquia». <https://revistas.udea.edu.co/index.php/ingenieria/article/view/14940> (accedido 5 de octubre de 2022).
- [11] N. Kehtarnavaz, «Digital signal processing system design: LabVIEW-based hybrid programming», *Digital Signal Processing System Design: LabVIEW-Based Hybrid Programming*, pp. 1-325, abr. 2008, DOI: 10.1016/B978-0-12-374490-6.X0001-3.
- [12] «Gabor Analysis and Algorithms», *Gabor Analysis and Algorithms*, 1998, DOI: 10.1007/978-1-4612-2016-9.
- [13] J. Pyra y A. Soltys, «Method for studying the structure of blast-induced vibrations in open-cast mines», *Journal of Vibroengineering*, vol. 18, n.º 6, pp. 3829-3840, ene. 2016, DOI: 10.21595/JVE.2016.17052.

- [14] F. Plante, G. Meyer, y W. A. Ainsworth, «Improvement of speech spectrogram accuracy by the method of reassignment», *IEEE Transactions on Speech and Audio Processing*, vol. 6, n.º 3, pp. 282-287, 1998, DOI: 10.1109/89.668821.
- [15] K. Fitz y A. Member, « On the Use of Time-Frequency Reassignment in Additive Sound Modeling», *J. Audio Eng. Soc.*, vol. 50, n.º 11, 2002.
- [16] Jesús Ponce de León Vázquez, «Análisis y síntesis de señales de audio a través de la Transformada Wavelet continua y compleja: el algoritmo CWAS», Universidad de Zaragoza, Zaragoza, 2012. Accedido: 5 de octubre de 2022. [En línea]. Disponible en: <https://zaguan.unizar.es/record/9623/files/TESIS-2012-111.pdf>
- [17] W. Elvira-García, F. Planas, y A. Ma, «El espectrograma reasignado: creación, uso y utilidad en fonética».
- [18] «Método de reasignación Introducción y El espectrograma como representación de tiempo-frecuencia». https://hmong.es/wiki/Reassignment_method (accedido 5 de octubre de 2022).
- [19] S. A. Fulop y K. Fitz, «Algorithms for computing the time-corrected instantaneous frequency (reassigned) spectrogram, with applications», *J Acoust Soc Am*, vol. 119, n.º 1, pp. 360-371, ene. 2006, DOI: 10.1121/1.2133000.
- [20] «Reassignment method - HandWiki». https://handwiki.org/wiki/Reassignment_method (accedido 5 de octubre de 2022).
- [21] T. Oberlin, S. Meignen, y V. Perrier, «The fourier-based synchrosqueezing transform», *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 315-319, 2014, DOI: 10.1109/ICASSP.2014.6853609.
- [22] S. Meignen, T. Oberlin, y D. H. Pham, «Synchrosqueezing transforms: From low- to high-frequency modulations and perspectives», *C R Phys*, vol. 20, n.º 5, pp. 449-460, jul. 2019, DOI: 10.1016/J.CRHY.2019.07.001.
- [23] A. Rueda y S. Krishnan, «Augmenting Dysphonia Voice Using Fourier-based Synchrosqueezing Transform for a CNN Classifier», *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2019-May, pp. 6415-6419, may 2019, DOI: 10.1109/ICASSP.2019.8682391.
- [24] T. Oberlin, S. Meignen, y V. Perrier, «The fourier-based synchrosqueezing transform», *ICASSP, IEEE International Conference on Acoustics, Speech and*

- Signal Processing - Proceedings*, pp. 315-319, 2014, DOI: 10.1109/ICASSP.2014.6853609.
- [25] G. Thakur y H. T. Wu, «Synchrosqueezing-Based Recovery of Instantaneous Frequency from Nonuniform Samples», <https://doi.org/10.1137/100798818>, vol. 43, n.º 5, pp. 2078-2095, sep. 2011, DOI: 10.1137/100798818.
- [26] V. v. Moca, H. Bârzan, A. Nagy-Dăbâcan, y R. C. Mureşan, «Time-frequency super-resolution with superlets», *Nature Communications 2021 12:1*, vol. 12, n.º 1, pp. 1-18, ene. 2021, DOI: 10.1038/s41467-020-20539-9.
- [27] H. Bârzan, V. v. Moca, A. M. Ichim, y R. C. Murean, «Fractional superlets», *European Signal Processing Conference*, vol. 2021-January, pp. 2220-2224, ene. 2021, DOI: 10.23919/EUSIPCO47968.2020.9287873.
- [28] Y. Tian, J. Gao, D. Wang, y Z. Li, «Super-Resolution Optimal Basic Wavelet Transform and Its Application in Thin-Bed Thickness Characterization», *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, 2022, DOI: 10.1109/TGRS.2022.3199450.
- [29] «Start stopwatch timer - MATLAB tic - MathWorks». <https://la.mathworks.com/help/matlab/ref/tic.html?lang=en> (accedido 6 de octubre de 2022).
- [30] K. Yatabe, «DGTtool», jun. 2021, DOI: 10.5281/ZENODO.5010751.
- [31] «Analyze signals in the frequency and time-frequency domains - MATLAB pspectrum - MathWorks América Latina». https://la.mathworks.com/help/signal/ref/pspectrum.html#mw_b88e6047-f250-4ed5-91b2-9d87da5f9f98_seealso (accedido 6 de octubre de 2022).
- [32] «Fourier synchrosqueezed transform - MATLAB fsst - MathWorks América Latina». <https://la.mathworks.com/help/signal/ref/fsst.html#d124e74018> (accedido 6 de diciembre de 2022).
- [33] Harald Bârzan, « Transylvanian Institute Of Neuroscience Superlets», 2019. <https://github.com/TransylvanianInstituteOfNeuroscience/Superlets> (accedido 6 de diciembre de 2022).

5 ANEXOS

ANEXO I. Función ASLT (Matlab)

ANEXO II. Función FASLT (Matlab)

ANEXO III. Código Completo (Matlab)

ANEXO IV. Manual de Usuario

ANEXO I Función ASLT

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   TRANSFORMACIÓN ADAPTABLE SUPERRESOLUTION WAVELET (SUPERLET)
%
%   AUTHOR:           Harald Bârzan
%   DATE:            April 2019
%   DESCRIPTION:
%
%   Calcula la transformada adaptativa de wavelets de
superresolución
%   (superlet) en los datos de entrada para producir una
representación de
%   tiempo-frecuencia. Para cada frecuencia de interés, se
elegirá el orden
%   entero más cercano del intervalo de orden para producir cada
superlet.
%   Un superlet es un conjunto de wavelets con la misma
frecuencia central
%   pero con diferente número de ciclos.
%
%   REFERENCE:
%
%   Superlets: time-frequency super-resolution using wavelet
sets
%   Moca, V.V., Nagy-Dăbâcan, A., Bârzan, H., Murean, R.C.
%   https://www.biorxiv.org/content/10.1101/583732v1.full
%
%   NOTES:
%
%   Si los datos de entrada consisten en múltiples buffers, se
calculará un
%   espectro wavelets para cada uno de los buffers y se
promediará para
%   producir el resultado final.
%   Si el parámetro de orden (ord) está vacío, esta función
devolverá el
%   CWT estándar (un wavelets por frecuencia de interés).
%
%   INPUT:
%   > input           - [buffers x muestras] matrix
%   > Fs             - frecuencia de muestreo en Hz
%   > F              - bufer de frecuencia de interés
%   > Ncyc          - número de ciclos de wavelets iniciales
%   > ord            - [1 x 2] intervalo de órdenes de
superresolución (opcional)
```

```

% > mult          - especifica el uso de la superresolución
multiplicativa
%                  (0 - aditivo, != 0 - multiplicativo)
%
% OUTPUT:
% > wtresult      - [frecuencias x muestras] espectro superlet
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [wtresult] = aslt(input, Fs, F, Ncyc, ord, mult)

% comprobar la frecuencia del parámetro de interés
if (isempty(F))
    error('frecuencias not defined');
end

% comprobar el parámetro de orden e inicializar el
% orden utilizado en cada frecuencia. si está vacío,
% ir con un orden de 1 para cada frecuencia (una sola wavelets
por conjunto)
if (~isempty(ord))
    order_ls          = fix(linspace(ord(1), ord(2),
numel(F)));
else
    order_ls          = ones(numel(F), 1);
end

% validar el buffer de entrada
if (isempty(input))
    error('input is empty');
end

% si la entrada es un vector de columnas, convertirlo en un
vector de filas
if (size(input, 2) == 1 && size(input, 1) > 1)
    input = input';
end

% obtener el tamaño de la entrada
[Nbuffers, Npoints] = size(input);

% el relleno será del tamaño de los cojines cero laterales, que
sirven para
% evitar los efectos de borde durante la convolución
padding = 0;

% los conjuntos de wavelets
wavelets = cell(numel(F), max(ord));

```

```

% inicializar conjuntos de wavelets para aditivo
% o multiplicativo superresolución
if (mult ~= 0)
    for i_freq = 1 : numel(F)
        for i_ord = 1 : order_ls(i_freq)
            % cada nueva wavelets tiene Ncyc ciclos adicionales
            % (superresolución multiplicativa)
            wavelets{i_freq, i_ord} = cxmorlet(F(i_freq), Ncyc *
i_ord, Fs);

            % el margen será la mitad del tamaño de la wavelets
            más grande
            padding = max(padding, fix(numel(wavelets{i_freq,
i_ord}) / 2));
        end
    end
else
    for i_freq = 1 : numel(F)
        for i_ord = 1 : order_ls(i_freq)
            % cada nueva wavelets tiene un ciclo adicional
            (superresolución aditiva)
            wavelets{i_freq, i_ord} = cxmorlet(F(i_freq), Ncyc +
(i_ord - 1), Fs);

            % el margen será la mitad del tamaño de la wavelets
            más grande
            padding = max(padding, fix(numel(wavelets{i_freq,
i_ord}) / 2));
        end
    end
end

% el búfer relleno de cero
buffer = zeros(Npoints + 2 * padding, 1);

% el escalograma de salida
wtresult = zeros(numel(F), Npoints);

% indexadores de conveniencia para el búfer de relleno cero
bufbegin    = padding + 1;
bufend      = padding + Npoints;

% bucle sobre los buffers de entrada
for i_buf = 1 : Nbuffers
    for i_freq = 1 : numel(F)

```

```

        % buffer de agrupación, comienza con 1 porque estamos
        haciendo la media geométrica
        temp = ones(1, Npoints);

        % llenar la parte central del buffer con datos de
        entrada
        buffer(bufbegin : bufend) = input(i_buf, :);

        % calcular la convolución del buffer con cada wavelets
        % del conjunto actual
        for i_ord = 1 : order_ls(i_freq)
            % convolución restringida (tamaño de entrada ==
            tamaño de salida)
            tempcx = conv(buffer, wavelets{i_freq, i_ord},
            'same');

            % acumular la magnitud (por 2 para obtener la
            energía
            % espectral completa
            temp = temp .* (2 .* abs(tempcx(bufbegin : bufend))
            .^ 2)';
        end

        % calcular la potencia de la media geométrica
        root = 1 / order_ls(i_freq);
        temp = temp .^ root;

        % acumular el FOI actual al espectro de resultados
        wresult(i_freq, :) = wresult(i_freq, :) + temp;
    end
end

% escalar la salida por el número de buffers de entrada
wresult = wresult ./ Nbuffers;

return

% calcula la wavelets Morlet compleja para la frecuencia central
deseada Fc
% con Nc ciclos, con una frecuencia de muestreo Fs.
function w = cxmorlet(Fc, Nc, Fs)
    %queremos tener el último pico a 2,5 SD
    sd = (Nc / 2) * (1 / Fc) / 2.5;
    wl = 2 * floor(fix(6 * sd * Fs)/2) + 1;
    w = zeros(wl, 1);
    gi = 0;

```

```

off = fix(wl / 2);

for i = 1 : wl
    t      = (i - 1 - off) / Fs;
    w(i)   = bw_cf(t, sd, Fc);
    gi     = gi + gauss(t, sd);
end

w = w ./ gi;
return

% calcular los coeficientes de las wavelets complejas para el
punto de
% tiempo deseado t, ancho de banda bw y frecuencia central cf
function res = bw_cf(t, bw, cf)
    cnorm   = 1 / (bw * sqrt(2 * pi));
    expl    = cnorm * exp(-(t^2) / (2 * bw^2));
    res     = exp(2i * pi * cf * t) * expl;
return;

% calcular el coeficiente gaussiano para el punto de tiempo
deseado t
% y la desviación estándar sd
function res = gauss(t, sd)
    cnorm   = 1 / (sd * sqrt(2 * pi));
    res     = cnorm * exp(-(t^2) / (2 * sd^2));
return;

```


ANEXO II Función FASLT

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Transformación wavelets (superlet) de superresolución
% adaptativa fraccional
%
% Autor:          Harald Bârzan
% Fecha:          April 2019
% DESCRIPCION:
%
% Calcula la transformada adaptativa de wavelets de
% superresolución
% (superlet) en los datos de entrada para producir una
% representación de
% tiempo-frecuencia. Para cada frecuencia de interés, se
% elegirá el
% orden entero más cercano del intervalo de orden para
% producir cada
% superlet. Un superlet es un conjunto de wavelets con la
% misma
% frecuencia central pero con diferente número de ciclos.
%
% REFERENCIA:
%
% Time-frequency super-resolution with superlets
% Moca, V.V., Nagy-Dăbâcan, A., Bârzan, H., Mureșan, R.C.
% https://www.nature.com/articles/s41467-020-20539-9
%
% NOTES:
%
% Si los datos de entrada consisten en múltiples buffers, se
% calculará
% un espectro wavelets para cada uno de los buffers y se
% promediará para
% producir el resultado final.
% Si el parámetro de orden (ord) está vacío, esta función
% devolverá el
% CWT estándar (un wavelets por frecuencia de interés).
%
% INPUT:
% > input          - [buffers x muestras] matriz
% > Fs             - frecuencia de muestreo en Hz
% > F              - bufer de frecuencia de interés
% > c1             - número de ciclos de wavelets iniciales
% > o              - [1 x 2] intervalo de órdenes de
% superresolución (opcional)
```

```

% > mult          - especifica el uso de la superresolución
multiplicativa
%                  (0 - aditivo, != 0 - multiplicativo)
%
% OUTPUT:
% > wtresult      - [frecuencias x muestras] espectro superlet
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [wtresult] = faslt(input, Fs, F, c1, o, mult)

% comprobar la frecuencia del parámetro de interés
if (isempty(F))
    error('frecuencias not defined');
end

fix(8.2)

% compruebe el parámetro de orden e inicialice el orden
utilizado en cada
% frecuencia. si está vacío, vaya con un orden de 1 para cada
frecuencia
% (un solo wavelets por conjunto)
if (~isempty(o))
    % el orden fraccionario es lo que el usuario ponga
    order_frac = linspace(o(1), o(2), numel(F));
    % orden entero es el número real de wavelets en el conjunto
    order_int  = ceil(order_frac);
else
    order_frac = ones(numel(F), 1);
    order_int  = order_frac;
end

% validar el buffer de entrada
if (isempty(input))
    error('input is empty');
end

% si la entrada es un vector de columnas, convertirlo en un
vector de filas
if (size(input, 2) == 1 && size(input, 1) > 1)
    input = input';
end

% obtener el tamaño de la entrada
[Nbuffers, Npoints] = size(input);

% el relleno será del tamaño de los zero-pads laterales,

```

```

% que sirven para evitar efectos de borde durante la convolución

% los conjuntos de wavelets
wavelets = cell(numel(F), max(order_int));

% inicializar los conjuntos de wavelets para el aditivo o el
multiplicativo
% superresolution

for i_freq = 1 : numel(F)
    for i_ord = 1 : order_int(i_freq)

        % calcular el número de ciclos (aditivo o
multiplicativo)
        if (mult ~= 0)
            n_cyc = i_ord * c1;
        else
            n_cyc = i_ord + c1;
        end

        % añadir la wavelets al conjunto
        wavelets{i_freq, i_ord} = cxmorlet(F(i_freq), n_cyc,
Fs);

        % el margen será la mitad del tamaño de la wavelets más
grande
        padding = max(padding, fix(numel(wavelets{i_freq,
i_ord}) / 2));
    end
end

% el búfer de relleno cero
buffer = zeros(Npoints + 2 * padding, 1);

% el escalograma de salida
wresult = zeros(numel(F), Npoints);

% indexadores de conveniencia para el búfer de relleno cero
bufbegin    = padding + 1;
bufend      = padding + Npoints;

% bucle sobre los buffers de entrada
for i_buf = 1 : Nbuffers
    for i_freq = 1 : numel(F)
        % buffer de agrupación, comienza con 1 porque estamos
haciendo la media geométrica
        temp = ones(1, Npoints);
    end
end

```

```

        % llenar la parte central del buffer con datos de
entrada
        buffer(bufbegin : bufend) = input(i_buf, :);

        % obtener el número de wavelets enteros
        n_wavelets = floor(order_frac(i_freq));

        % calcular la convolución del buffer con cada wavelets
        % del conjunto actual (wavelets enteros)
        for i_ord = 1 : n_wavelets
            % convolución restringida (tamaño de entrada ==
tamaño de salida)
            tempcx = conv(buffer, wavelets{i_freq, i_ord},
'same');

            % acumular la magnitud (por 2 para obtener la
energía espectral
            % completa energía), pool con exponente = 1
            temp = temp .* (2 .* abs(tempcx(bufbegin : bufend))
.^ 2)';
        end

        % manejar el exponente fraccionario
        if (is_fractional(order_frac(i_freq)) && ...
            ~isempty(wavelets{i_freq, order_int(i_freq)}))
            % establecer el índice de la orden
            i_ord = order_int(i_freq);

            % el exponente es el resto fraccionario
            exponent = order_frac(i_freq) -
fix(order_frac(i_freq));

            % convolución restringida (tamaño de entrada ==
tamaño de salida)
            tempcx = conv(buffer, wavelets{i_freq, i_ord},
'same');

            % acumular la magnitud (multiplicada por 2 para
obtener la
            % energía espectral completa), pool con exponente =
1
            temp = temp .* ((2 .* abs(tempcx(bufbegin : bufend))
.^ 2)') .^ exponent;
        end

        % calcular el orden de la media geométrica

```

```

        root = 1 / order_frac(i_freq);
        temp = temp .^ root;

        % acumular el FOI actual al espectro de resultados
        wtresult(i_freq, :) = wtresult(i_freq, :) + temp;
    end
end

% escalar la salida por el número de buffers de entrada
wtresult = wtresult ./ Nbuffers;

return

% calcula la wavelets Morlet compleja para la frecuencia central
deseada Fc
% con Nc ciclos, con una frecuencia de muestreo Fs.
function w = cxmorlet(Fc, Nc, Fs)
    % queremos tener el último pico a 2,5 SD
    sd = (Nc / 2) * (1 / Fc) / 2.5;
    wl = 2 * floor(fix(6 * sd * Fs)/2) + 1;
    w = zeros(wl, 1);
    gi = 0;
    off = fix(wl / 2);

    for i = 1 : wl
        t = (i - 1 - off) / Fs;
        w(i) = bw_cf(t, sd, Fc);
        gi = gi + gauss(t, sd);
    end

    w = w ./ gi;
return

% compute the complex wavelet coefficients for the desired time
point t,
% bandwidth bw and center frequency cf
function res = bw_cf(t, bw, cf)
    cnorm = 1 / (bw * sqrt(2 * pi));
    expl = cnorm * exp(-(t^2) / (2 * bw^2));
    res = exp(2i * pi * cf * t) * expl;
return;

% calcular los coeficientes de las wavelets complejas para el
punto
% de tiempo deseado t, ancho de banda bw y frecuencia central cf
function res = gauss(t, sd)

```

```
    cnorm = 1 / (sd * sqrt(2 * pi));  
    res = cnorm * exp(-(t^2) / (2 * sd^2));  
return;  
  
% tell me if a number is an integer or a fractional  
function res = is_fractional(x)  
    res = fix(x) ~= x;  
return;
```

ANEXO III Código Completo

```
%MAIN PRINCIPAL
%Metodos Modernos
%Autor: Leandro Celorio

clc,
close all,
clear all,

%% 1. Recepción de datos (frecuencias, amplitudes y duraciones)
de todas las componentes:
% Los datos siguientes solo deben ser remplazados según se desee
introducir problemas de:
% - Resolución en frecuencia: tonos muy juntos
% - Detección: componentes muy pequeñas relativas a la mayor de

% %1.1 Frecuencias de las componentes (ordenadas de menor a
mayor):
% Prueba 1
f1=100; f2=130; f3=175; f4=250; f5=300;

% Prueba 2
%f1=11; f2=20; f3=30; f4=38; f5=44;

%Prueba 3
%f1=80; f2=130; f3=165; f4=200; f5=235;

% Prueba 1 y 2
a1 = 1; a2 = 1; a3 = 1; a4 = 1; a5 = 1;

% Prueba 3
%a1 = 0.001; a2 = 0.01; a3 = 0.1; a4 = 0.1; a5 = 1;

% 1.3 Duraciones de las componentes:
t1=[0 1]; t2=[1 4]; t3=[2 3]; t4=[3 5]; t5=[4 5];
% 2. Generación de la señal compuesta (suma de todas las
componentes):
Fs=10*f5; % Fs respecto de la f5 que es la más exigente
Ts=1/Fs;
duracion = 5; % es el tiempo máximo que ocupan las componentes
t=0:Ts:duracion;
% se realiza el método anterior pero ahora de manera compacta:
% Primeramente se generan cada una de las componentes en todo el
intervalo (8 segundos):
comp1=a1*sin(2*pi*f1*t);
```

```

comp2=a2*sin(2*pi*f2*t);
comp3=a3*sin(2*pi*f3*t);
comp4=a4*sin(2*pi*f4*t);
comp5=a5*sin(2*pi*f5*t);

% Se llena de ceros las componentes en los intervalos donde no
existen pero se aprovecha
% las posibilidades de Matlab:
comp1=(t<=1).*(comp1);
comp2=((t>=1)&(t<=4)).*(comp2);
comp3=((t>2) & (t<=3)).*(comp3);
comp4=((t>=3)&(t<=5)).*(comp4);
comp5=((t>=4)&(t<=5)).*(comp5);

% Presentación de las componentes en forma ascendente tal como
es el
% diagrama TF ideal:
figure

subplot(6,1,1);plot(t,comp5);xlim([0 5]);title('componente 5');
xlabel('t(s)');grid on;
subplot(6,1,2);plot(t,comp4);xlim([0 5]);title('componente 4');
xlabel('t(s)');grid on;
subplot(6,1,3);plot(t,comp3);xlim([0 5]);title('componente 3');
xlabel('t(s)');grid on;
subplot(6,1,4);plot(t,comp2);xlim([0 5]);title('componente 2');
xlabel('t(s)');grid on;
subplot(6,1,5);plot(t,comp1);xlim([0 5]);title('componente 1');
xlabel('t(s)');grid on;

% Una vez generadas las componentes, las sumamos para obtener la
señal compuesta:
yt1 = comp1+comp2+comp3+comp4+comp5;
subplot(6,1,6);plot(t,yt1,'r');title('Señal compuesta');
xlabel('t(s)');grid on;
sgtitle('Componentes individuales y señal compuesta')
xlim([0 5])
% Visualización del TF ideal en conjunto con la señal compuesta
obtenida de
% manera que se vea la existencia de una o varias componentes
según los intervalos
figure
hold on
grid minor
plot (t1,[f1 f1],'linewidth',1.5)
plot (t2,[f2 f2],'linewidth',1.5)
plot (t3,[f3 f3],'linewidth',1.5)

```



```

plot (t4,[f4 f4], 'linewidth',1.5)
plot (t5,[f5 f5], 'linewidth',1.5)
hold off
title('DIAGRAMA TIEMPO-FRECUENCIA IDEAL')
xlabel('t(sec)')
ylabel('F(Hz)')
xlim([0 5])
ylim([0 250])
legend(['a1 = ' num2str(a1)], ['a2 = ' num2str(a2)], ['a3 = '
num2str(a3)], ['a4 = ' num2str(a4)], ['a5 = ' num2str(a5)])

%% METODO: TRANSFORMADA DE GABOR
tic
tic
xn = yt1;
inputSignal = yt1'; %señal de entrada

% Se crea el objeto DGTtool
% Algunas de las opciones se pueden omitir (se utilizarán los
valores predeterminados).
% El orden de las opciones no está restringido (cualquier orden
es aceptable).
% La lista de ventanas disponibles se puede obtener mediante
DGTtool.windowList.
% El nombre de la ventana se puede acortar (por ejemplo, 'b' es
aceptable en lugar de 'Blackman').f
% Si la versión de MATLAB es 2021a o posterior, se puede
utilizar la siguiente sintaxis.
% F =
DGTtool(windowShift=20,windowLength=250,FFTnum=400,windowName='B
lackman')
%F =
DGTtool('windowShift',5,'windowLength',250,'FFTnum',256,'windowN
ame','Blackman')
%F =
DGTtool('windowShift',20,'windowLength',250,'windowName','Blackm
an')
F = DGTtool('windowName', 'Gauss');

% F =
DGTtool('windowShift',100,'windowLength',1000,'FFTnum',nFFT_vent
ana_RT,'windowName','Gauss')
% Se calcula el espectrograma
% La señal de entrada debe ser un vector de columna.
X = F(inputSignal); %Se calcula el espectrograma de la señal de
entrada
x = F.pinv(X);% convierte el espectrograma de nuevo en señal

```

```

toc
% Trazar el espectrograma
% Gráfica del espectrograma. La frecuencia de muestreo se puede
omitir: F.plot(x).
% Nota: Las funciones de trazado se pueden usar directamente
después de definir F.
F.plot(x,Fs); %Graficacion del espectrograma
sgtitle('Transformada de Gabor ventana de Gauss');
toc

%% REASIGNADA DE FOURIER

% Para Frecuencia
figure
tic
tic
[P, F, T]=pspectrum(yt1,Fs,'FrequencyLimits',[0
250],'FrequencyResolution',1.1,'Leakage',0.6,'spectrogram','Reas
sign',true);
toc
mesh(seconds(T),F,P,'LineWidth',1.5)
%colormap parula

title('ESPECTROGRAMA REASIGNADO EN FRECUENCIA')
xlabel('Tiempo')
ylabel('F(Hz)')
axis tight
view(2)
toc

%Para tiempo
[P1, F1, T1]=pspectrum(yt1,Fs,'FrequencyLimits',[0
250],'TimeResolution',0.497,'Leakage',0.7,'spectrogram','Reassig
n',true);
figure
mesh(seconds(T1),F1,P1,'LineWidth',1.5)
colormap(flag(200))
%p.MarkerSize = 8;
title('ESPECTROGRAMA REASIGNADO EN TIEMPO')
xlabel('Tiempo')
ylabel('F(Hz)')
axis tight
view(2)

%% SINCRONIZADA DE FOURIER
% FSST RESOLUCION EN TIEMPO

```

```

figure
window_T=rectwin(fix(length(xn)/2));
[d_T, f_T, t_T] = fsst(xn,Fs,window_T);

contour(t_T,f_T,abs(d_T));
title('FSST CON RESOLUCION EN TIEMPO')
xlabel('t(sec)')
ylabel('F(Hz)')
axis([0 5 0 50])
grid minor

% FSST RESOLUCION EN FRECUENCIA
figure
window_F=blackmanharris(fix(length(xn)/5));
[d_F, f_F, t_F] = fsst(xn,Fs,window_F);
contour(t_F,f_F,abs(d_F));
title('FSST CON RESOLUCION EN FRECUENCIA')
xlabel('t(sec)')
ylabel('F(Hz)')
axis([0 5 0 50])
grid minor

%% SUPERLETS

%Parametros de entrada
fois = 1:0.5:250; %vector buffer para el rango de las
frecuencias
srord= [1, 30]; %vector de ciclos para tiempo
srord2= [1, 250]; %vector de ciclos para frecuencia

% tiempo
tic
RT_SL = aslt(xn, Fs, fois, 5, srord, 0);
toc

% frecuencia
tic
RF_SL = faslt(xn, Fs, fois, 8, srord2, 0);
toc

%Resolucion en frecuencia Superlets
figure
imagesc(t,fois,RF_SL);
set(gca, 'ydir', 'normal');
colormap jet;
ylim([0 250]);
grid on;
xlabel('TIEMPO (segundos)');

```

```
title('ESPECTROGRAMA CON RESOLUCIÓN EN FRECUENCIA');
xlabel('t(sec)');
ylabel('F(Hz)');

%Resolucion en tiempo Superlets
figure
imagesc(t,fois,RT_SL);
set(gca, 'ydir', 'normal');
colormap jet;
axis([0 5 0 250]);
grid on;
xlabel('TIEMPO (segundos)');
title('SUPERLETS CON RESOLUCIÓN EN TIEMPO');
xlabel('t(sec)');
ylabel('F(Hz)');
```

ANEXO IV Manual de Usuario

El presente manual tiene como finalidad mostrar el uso correcto del ANEXO III. En este Anexo se encuentra unificado los códigos empleados para la experimentación, el mismo consta de los siguientes segmentos:

1. Ingreso de frecuencias, amplitudes y duración de cada componente.
2. Creación de la señal multicomponente.
3. Creación del diagrama tiempo - frecuencia.
4. Método Transformada de Gabor.
5. Método Reasignada de Fourier.
6. Método Sincronizada de Fourier.
7. Método Superlets.

Para emplear el Código Completo se debe crear un nuevo script en Matlab es su versión R2021a o superior, adicional se debe instalar el siguiente toolbox : "*Signal Processing Toolbox*". Para analizar un solo método se debe comentar los otros 3 para no tener una exigencia elevada en la carga computacional.

Adicional, para el caso de los Superlets se debe emplear funciones externas ANEXO I (ASLT) y ANEXO II (FASLT), estas funciones deben ser creadas en el mismo directorio donde se encuentre el script Código Completo.