UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA CIVIL

INSTRUMENTACIÓN Y ANÁLISIS SÍSMICO EXPERIMENTAL DE LA TORRE PEÑUELAS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL

MARTÍN JAVIER COX KUNSTMANN

PROFESOR GUÍA:
FRANCISCO HERNÁNDEZ PRADO

PROFESOR CO-GUÍA:
PABLO GUINDOS BRETONES

MIEMBROS DE LA COMISIÓN:
PEDRO SOTO MUÑOZ
MARIO WAGNER MUÑOZ

Este trabajo ha sido parcialmente financiado por:
Centro UC de Innovación en Madera

SANTIAGO DE CHILE
2022

INSTRUMENTACIÓN Y ANÁLISIS SÍSMICO EXPERIMENTAL DE LA TORRE PEÑUELAS

A 98 km de Santiago, en una colaboración entre el MINVU y el Centro de Innovación en Madera UC se edificó el proyecto de la Torre Peñuelas, uno de los edificios más altos de madera en Chile y Latinoamérica, que si bien tiene un carácter experimental, busca validar la viabilidad de la construcción y correcto desempeño de estructuras de mediana altura son constituidas de este material en el país.

El objetivo principal de este trabajo consistirá en continuar con esta investigación, realizando y colaborando en la instrumentación sísmica de la torre que es llevada en modo conjunto entre RENADIC (Red de Cobertura Nacional de Acelerógrafos) y el centro de innovación de la madera, midiendo su comportamiento de manera experimental y procesando datos ante microvibraciones, lo que permitirá encontrar, a través de distintos métodos computacionales, las propiedades dinámicas de la torre y analizar su comportamiento sísmico. Es de mencionar que se decidió utilizar equipos de bajo costo tipo MEMS (sistema microelectromecánico) referidos a equipos Raspberry Shake R4SD, siendo está instrumentación pionera en Chile en el uso de este tipo de sensores para el monitoreo de estructuras, por lo cual se debe implementar un sistema para sincronizar, procesar y extraer datos de forma remota, entre otras cosas.

Esta investigación será realizada en 3 etapas. En primer lugar, se realizará una investigación bibliográfica con el fin de estudiar y comprender los instrumentos que se utilizarán y las técnicas de identificación de propiedades dinámicas y de post-procesamiento a emplear.

Como segunda etapa, se realizará la instrumentación de la torre a través de equipos Raspberry Shake R4SD que serán instalados de forma estratégica en 2 pisos de la torre y en un pozo en la base, para medir los movimientos en las 3 direcciones principales de la estructura, lo que servirá para monitorear su comportamiento estructural ante distintas excitaciones sísmicas. Esto requiere que los instrumentos Raspberry trabajen en forma adecuada y sincrónica, proveyendo energía eléctrica y una adquisición sincronizada para los seis R4SD que se instalarán en la estructura. Con este propósito, se llevaran a cabo ensayos de los equipos en la mesa vibradora del laboratorio de Sólidos (3D) de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile que serán contrastados con mediciones realizadas con acelerómetros Episensors. Luego se desarrollarán programas computacionales que permitan extraer los datos de la respuesta sísmica asociado a eventos que superen ciertos niveles de aceleración en determinados acelerómetros, quedando de esta forma los datos almacenados en un computador base que se encuentra en la torre misma (y que permite vía uso de un programa de acceso remoto, extraer de forma remota los datos de respuesta y procesar los datos de microvibraciones de forma periódica).

Finalmente, el objetivo prioritario será registrar eventos sísmicos de la torre empleando la red de monitoreo en base a equipos RS4D, es decir es decir, dejar funcionando en forma efectiva un sistema operacional. Se espera que a partir de los datos registrados de microvibraciones, estos sean post-procesados para obtener las propiedades dinámicas de la estructura y estudiar como varían según las condiciones ambientales.

Se la dedico a todas las personas que
hicieron agradable mi paso por la universidad

Agradecimientos

En primer lugar agradezco a mi abuela quien me brindó siempre todo lo necesario para poder estudiar en Santiago. Quiero agradecer también a mis papás y hermanos por estar ahí durante los 6 años de la carrera; y a mis primos y tíos por sus buenos deseos y aliento.

Quiero agradecer también a mis amigos de la universidad quienes me apoyaron constantemente e hicieron que se disfrutara el paso por la universidad: Mati, Iván, Flo, Cristian, Benja, Isa, Feña, y a todos con los que conviví diariamente en la universidad.

También agradezco a mis amigos de Puerto Varas que siempre hicieron agradable la vuelta, siempre estuvieron cuando los necesité y me fueron a ver a Santiago: Carlitos, Bahamonde, Angel, Camila, Chio, Genesis, Joseph, Leiva, Jh, Vicho, Mati.

Igualmente quiero mencionar aquí al equipo de difusión colegios y en especial a la Pili y María José que hicieron muy grato el ambiente y me permitieron darme cuenta de lo mucho que disfruto hablando de mi carrera.

Quiero agradecer en especial a los profesores que me acompañaron en el transcurso de este trabajo de título. Al profe Francisco por su increíble apoyo, guía y dedicación que entregó durante todo el año; Al profe Pedro por su gran ayuda y disposición durante todo el proceso; Al profe Pablo por darme la oportunidad de trabajar en el proyecto; y al Profe Mario por su buena disposición y entregar la motivación para explorar este mundo de la madera.

Finalmente quiero agradecer a todos los equipos docentes con los que pude compartir en la universidad y que ayudaron a mi desarrollo, tanto como miembro del equipo como estudiante.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Aspectos Generales

Chile es uno de los 10 países que más produce madera en el mundo y es líder junto a Nueva Zelanda en la plantación de Pino Radiata. Sin embargo, pese a esta condición de país maderero, y considerando que a diferencia de los otros materiales de construcción la madera es renovable, solo el 14 % de las viviendas y 16,8 % de las construcciones en general en Chile son construidas con este material como predominante[1].

Bajo este contexto, el año 2017, en una colaboración entre el Ministerio de Vivienda y Urbanismo (MINVU) y el Centro Investigación en Madera UC inicia la construcción de la Torre Peñuelas; un proyecto de carácter experimental que serviría para iniciar el estudio de estructuras de mediana altura en Chile al edificar el edificio más alto de Chile con madera con un sistema soportante del tipo marco-plataforma como su material predominante. Su ubicación es a 98 km de Santiago, vecino a la ruta 68 en dirección a Valparaíso en el Parque Nacional Lago Peñuelas[2].

Hasta su fin de construcción el 31 de octubre de 2019, se hicieron todos los diseños sísmicos de la estructura para luego estudiar todos los procesos constructivos que requiere este tipo de proyectos. Sin embargo, estudios posteriores como el monitoreo sísmico de la torre, que ayude a comprender el comportamiento real de este sistema ante estas solicitaciones aún no se han desarrollado. En Chile, este tipo de estudios además ayudaría al diseño estructural de futuras obras civiles desarrolladas con madera ya que se podría obtener valiosas conclusiones como las propiedades dinámicas del edificio y como este se comporta y/o desempeña durante eventos sísmicos, permitiendo validar el uso de este tipo de soluciones o proponer propuestas para mejorar su diseño.

La madera como tal es un material ortotrópico y que varía sus propiedades según las condiciones ambientales a las que se expone, por lo que sus propiedades dinámicas pueden variar en el tiempo, complejizando identificar cambios debido a daño que se traducen en cambios de las propiedades dinámicas (SHM). Por lo cual para vislumbrar si los cambios en las propiedades dinámicas son producidas por daño, se debe desagregar la variación natural que la estructura experimenta debido a factores ambientales.

Existen varios métodos para estimar las propiedades dinámicas de una estructura, en-

tre los que se encuentra la Descomposición en el Dominio de la Frecuencia o FDD, el que utiliza las señales de respuesta a microvibraciones de la estructura proporcionadas por los sensores para obtener las propiedades dinámicas. De esta forma se podrán comparar estas propiedades en el tiempo y así verificar que tanto varían en el tiempo y en función de las condiciones ambientales. En caso que existan cambios de estado que no estén relacionados a efectos ambientales (desagregando el efecto), se podrá estudiar si estos son producidos por daño estructural o tienen otro origen.

En consecuencia, este trabajo de investigación presenta varias aristas de innovación en el sentido que se implementará una red de monitoreo utilizando equipos de bajo costo del tipo Raspberry Shake R4SD, los que podrían ser una alternativa plausible para desarrollar monitoreo sísmico a menores costos (10 % del costo actual) de otras soluciones que emplean equipos más exactos como los equipos de Kinemetrics[3] que han sido comúnmente empleados por RENADIC. Además, se espera poder procesar registros de microvibraciones para poder entender el comportamiento natural de este tipo de estructuras ante las condiciones ambientales y como estas afectan sus propiedades dinámicas.

Figura 1.1: Torre Peñuelas.

1.2. Objetivos

1.2.1. Objetivo Principal

El objetivo principal de este trabajo es la implementación de una red de monitoreo sísmico que quede operativa de la Torre Peñuelas, utilizando equipos de bajo costo del tipo R4SD. A partir de los registros de microvibraciones obtenidos se espera obtener las propiedades dinámicas de la torre.

1.2.2. Objetivos Específicos

1. Revisión bibliográfica sobre el equipo que se utilizará para instrumentar la torre, para de esta forma saber mejor como utilizarlo, instalarlo y registrar sus datos.

2. Revisión bibliográfica sobre distintos métodos de identificación que sirvan para obtener propiedades dinámicas a partir de microvibraciones, como lo es el método FDD (Descomposición en el Dominio de la Frecuencia).

3. Describir en forma detallada el equipamiento utilizado y los detalles de su implementación además de cada paso y algoritmo requerido para registrar, sincronizar y procesar los datos los datos adquiridos de manera de registrar eventos sísmicos y generar un sistema de SHM (sistema de monitoreo de salud estructural).

4. Registrar la respuesta de la torre ante algún un movimiento sísmico para comparar la respuesta de los R4SD con equipos QDR que fueron instalados en forma paralela, y dejar al sistema operativo.

5. Obtener las propiedades dinámicas de la torre a través de los métodos estudiados anteriormente y correlacionarlas con las condiciones ambientales que son medidas por el sistema para obtener un modelo que permita correlacionar las frecuencias fundamentales con la temperatura y humedad relativa medida al interior de la estructura.

1.3. Metodología

1.3.1. Etapa 1: Revisión bibliográfica

1. Revisión bibliográfica sobre los procesos constructivos de la torre.

2. Revisión bibliográfica de los acelerómetros “Raspberry Shake 4D” para asegurar la correcta instalación y obtención de datos en terreno.

3. Revisión bibliográfica de los métodos de identificación existentes para la determinación de propiedades dinámicas de una estructura a partir de microvibraciones de su respuesta.

1.3.2. Etapa 2: Instrumentación

1. Estudio de sensibilidad y funcionamiento de los instrumentos sísmicos “Raspberry Shake 4D” en la mesa vibradora del laboratorio de solidos de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile, comparando la respuesta de sus 3 acelerómetros con la de sensores de alta gama (Episensors) sometidos al mismo movimiento

tridimensional. De esta forma evaluar la precisión de los equipos Raspberry Shake R4SD o analizar si se requiere alguna corrección de los datos adquiridos.

2. Colocar los sensores en posiciones estratégicas de la torre para registrar la respuesta sísmica y realizar SHM. Se colocará uno protegido a nivel de terreno (en un pozo construido para ello) para medir las aceleraciones que se observen directamente desde el suelo a la torre. Los otros sensores se colocarán en los pisos 2 y 4 de la torre. Con esto se espera medir los movimientos y comportamientos de la estructura en el tiempo durante las solicitaciones. Se instalarán adicionalmente dos equipos QDR de la marca Kinemetrics (de RENADIC) para medir la excitación en el pozo y otro en la estructura (para comparar el correcto funcionamiento de los equipos Raspberry Shake R4SD).

3. Describir en forma detallada el equipamiento utilizado y los detalles de su implementación además de cada paso y algoritmo requerido para registrar, sincronizar y procesar los datos los datos adquiridos de manera de registrar eventos sísmicos y generar un sistema de SHM

4. Implementar las herramientas computacionales para registrar y procesar los datos adquiridos, empleando programación en Octave[], Python[], ObsPy[], Raspberry Pi OS[].

1.3.3. Etapa 3: Determinación de las Propiedades Dinámicas

1. A partir de los datos registrados, utilizar métodos de identificación de estructuras ya implementados en el DIC como lo es el método FDD (descomposición en el dominio de la frecuencia) o SSI (Método del Subespacio Estocástico); adaptándolos al sistema Raspberry Pi OS para procesar estos datos y obtener las propiedades dinámicas de una estructura con estas características de manera experimental.

2. Obtener las propiedades dinámicas de la torre a través de los métodos estudiados anteriormente y correlacionarlas con las condiciones ambientales que son medidas por el sistema, para obtener un modelo que permita correlacionar las frecuencias fundamentales con la temperatura y humedad relativa medida al interior de la estructura.

Capítulo 2

Torre Experimental Peñuelas

2.1. Detalles generales

Corresponde a una torre de uso experimental de 5 pisos más una cubierta constituida principalmente de madera bajo el sistema marco plataforma. Es la primera estructura sismo resistente de 6 pisos y resistente al fuego que incorpora tecnología ATS, que consiste en la incorporación de barras de acero tensadas al interior de la estructura, en Chile y Latinoamérica.

Se construyó específicamente en la Reserva Nacional Peñuelas, ubicada en la ruta 68, km 87.3 de la comuna de Valparaíso. Su dueño, el estar ubicado en una reserva nacional, es la Corporación Nacional Forestal (CONAF).

Figura 2.1: Torre Peñuelas.

2.2. Detalles Arquitectónicos

Posee una altura de 19.4 metros sobre el nivel del suelo, lo que la convierte en el edificio más alto de Latinoamérica hecho de madera. Todos sus pisos se encuentran sobre el nivel del suelo, por lo que no posee subterráneo y bajo el nivel del suelo solo se encuentran las fundaciones. Considerando lo anterior, la altura total de la torre asciende a 20 metros.

La superficie de cada piso es la misma, por lo que la estructuración será prácticamente la misma, pero con pequeñas diferencias entre el piso 1, el piso tipo (del 2do al 5to) y la terraza. La superficie total de la torre es de 95.75 m^2. Estas carácterísticas de superficie sumadas a su altura le otorgan una condición de esbeltez notable.

La torre al ser considerada experimental se encuentra vacía en los primeros 4 pisos, siendo los pisos 2, 3 y 4 destinados exclusivamente a investigación.
El 5to piso por otro lado se encuentra amueblado para mostrar cómo sería un departamento en un edificio habitacional de madera.
Finalmente, la terraza será habilitada como mirador del Parque Nacional Peñuelas.

Las figuras 2.2, 2.3, 2.4 y 2.5 a continuación muestran las plantas de cada piso.

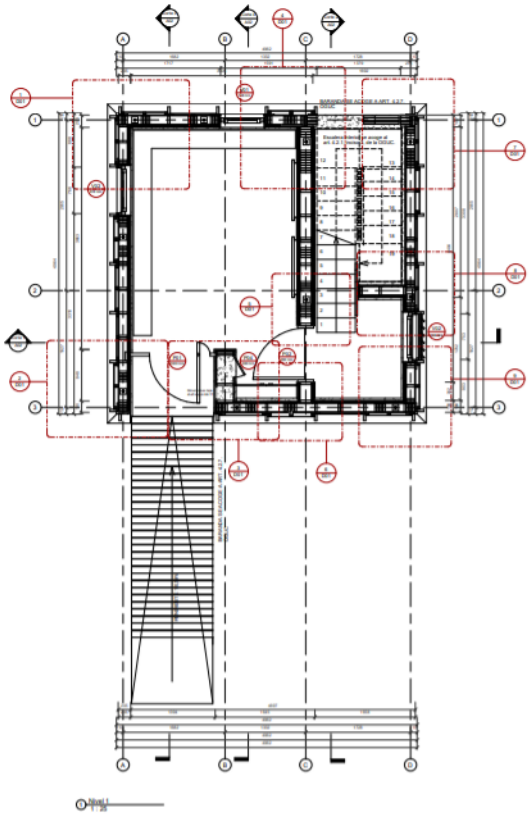

Figura 2.2: Planta piso 1.

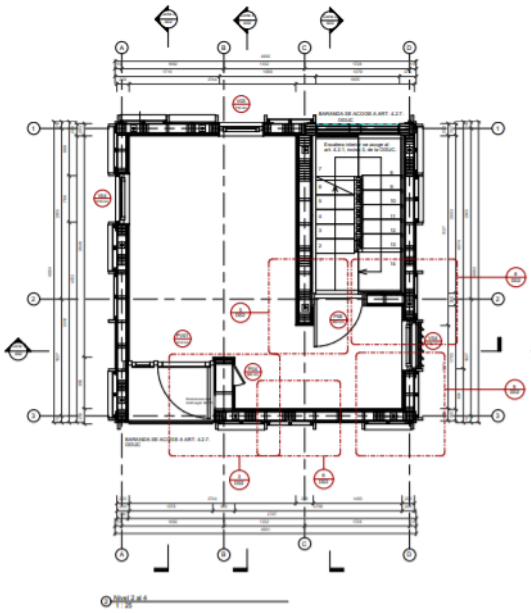

Figura 2.3: Planta pisos 2, 3 y 4.

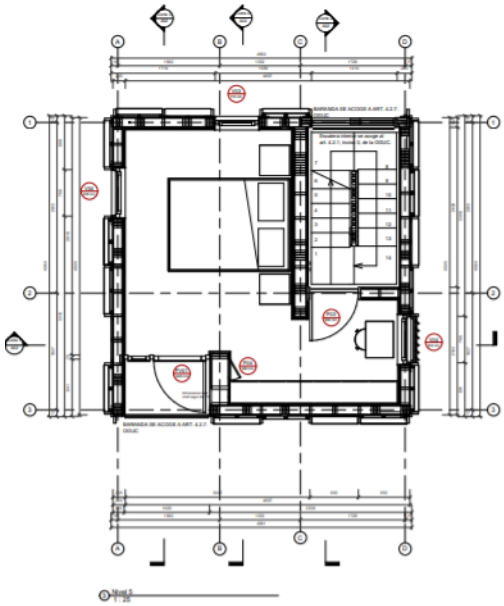

Figura 2.4: Planta piso 5.

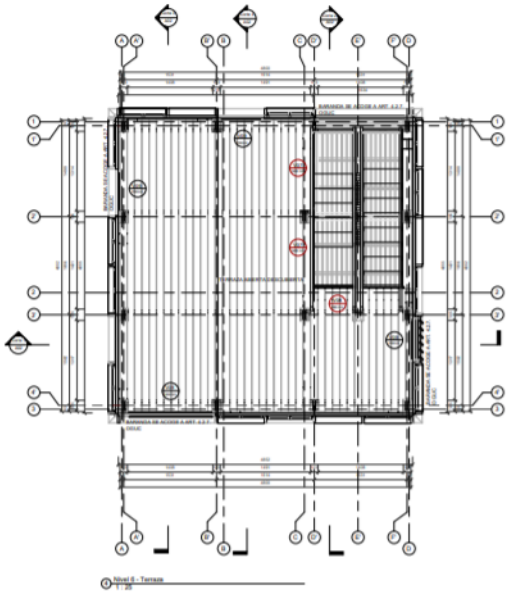

Figura 2.5: Planta piso 6 (terraza).

2.3. Estructuración de la Torre

La estructura está diseñada en madera según un sistema marco y plataforma. Cada piso se compone de 2 módulos prefabricados ensamblados y montados en obra con un total de 10 unidades modulares. Estos módulos fueron prefabricados en planta y se montaron en terreno. Cada módulo se compuso de muros estructurales, parte de las terminaciones y la losa superior de entrepiso.

Figura 2.6: Montaje de la Torre.

Capítulo 3

Sensores

3.1. Datos Generales

Los sensores utilizados para la instrumentación fueron de la marca Raspberry Shake", siendo 5 modelos RS4D indoor y 1 modelo RS4D outdoor. La diferencia de modelos radica simplemente en que el outdoor cuenta con una carcasa de protección ante los factores climáticos y externos para poder ser instalado en el exterior e interferir lo menos posible en la medición.

Estos instrumentos poseen 1 geófono que mide movimientos verticales y 3 acelerómetros del tipo MEMs. La diferencia de este modelo con los otros que presenta la marca es que el RS4D posee acelerómetros en las 3 direcciones principales en lugar de geófonos, ya que al ser Chile un país de alta sismicidad, los fuertes movimientos pueden saturar los geófonos.
Los geófonos en cambio son utilizados para aplicacions sismológicas de campo lejano ya que poseen una sensibilidad mucho mayor que los acelerómetros.

Figura 3.1: Sensor Raspberry Shake Indoor.

Figura 3.2: Sensor Raspberry Shake Outdoor.

Por otro lado los sensores Quake Data Recorder (QDR) de Kinemetrics corresponden a la versión precedente de su modelo ETNA2.

Actualmente los QDR se encuentran descontinuados por Kinemetrics, por lo que su remplazo directo sería el ETNA2 que se compone de 3 acelerómetros ortogonales.

Figura 3.3: Sensor ETNA2.

También se utilizó un sensor de temperatura, presión y humedad “BME280” el que en este caso se utilizará para medir solamente la humedad y temperatura del ambiente al que se encuentran los Raspberry Shake R4SD.

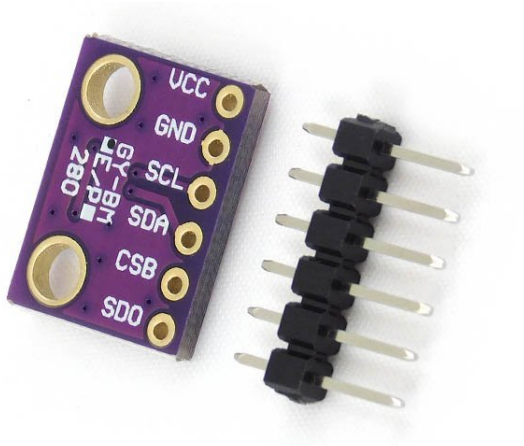

Figura 3.4: Sensor BME280.

3.2. Especificaciones Técnicas

Las Raspberry Shake son instrumentos de 135x110x50 mm con un peso estimado de 0.35 kg cada una, diseñados para operar a una temperatura entre 0 y 60 °C.
Poseen una memoria de 8Gb total y 5Gb de memoria efectiva descontando el mismo software y configuración del sensor.

Tanto el geófono como los acelerómetros del instrumento tienen una tasa de muestreo de 100 muestras por segundo con un ancho de banda de -3dB.

El geófono es único, de componente vertical y de 4,5 Hz y 390 Ohm.
Por otro lado, el acelerómetro es de tipo MEMS y cuenta de 3 componentes, en las 3 direcciones ortogonales principales con una frecuencia de 23 Hz.[4]

En cuanto a los instrumentos de Kinemetrics, la diferencia de los 2 modelos es que el ETNA 2 puede registrar movimientos en las 3 direcciones ortogonales. Los sensores QDR poseían un canal de digitalización de 11 bit a diferencia los 24 bit de los actuales junto a una frecuencia de 200Hz.
Actualmente los QDR se encuentran descontinuados por Kinemetrics, por lo que su remplazo directo sería el ETNA2.[3]

El sensor de temperatura y humedad BME280 es de 2,5 mm de ancho, 2,5 mm de largo y 0,93 mm de alto.
Posee una frecuencia de 1 Hz y su tolerancia permitida al error es de un 3 %.[5]

3.3. Calibración

Una de las grandes características de estos instrumentos es que son denominados “sismografos de bajo costo”.
Instrumentos como los “Kinemetrics”, marca experimentada y renombrada en el mundo de la instrumentación, en sus modelos que se utilizarían para esta clase de mediciones como el “ETNA2”, alcanzan valores para mayo de 2021 de U$7250 por sensor. En cambio, cada sensor

Indoor de la marca Raspberry Shake tiene un valor de U$624.99 y cada sensor Outdoor un costo de U$964.99[1].
Esto permite que los costos asociados a la instrumentación de una estructura puedan bajar al 10 % de lo que cuesta normalmente.

Por ser esta la primera vez que se utilizan estos instrumentos en estudios de este calibre, se deben comparar con sensores de alta precisión ya probados anteriormente.

Para lograrlo, se utilizó la mesa vibradora MOOG ubicada en el Laboratorio de Sólidos de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile, dónde se colocaron todos los sensores "Raspberry Shake" y se comparó la respuesta en todas las direcciones de cada acelerómetro por el que se compone, con un sensor "Kinemetric Episensor" que era registrado por un Daqbook/2005 de 16 bits de resolución; instrumentos reconocidos para su uso como sismógrafos.

Estos últimos instrumentos solo miden aceleraciones en una única dirección, por lo que se debieron colocar 3 para comparar con todas las direcciones del Raspberry Shake".

Los 5 sensores Raspberry Shake Indoor, el sensor Raspberry Shake Outdoor y los 3 sensores Episensor de Kinemetrics fueron colocados en la mesa al mismo tiempo, quedando de la siguiente forma:

[1] https://shop.raspberryshake.org/product/turnkey-iot-home-earth-monitor-rs-4d/

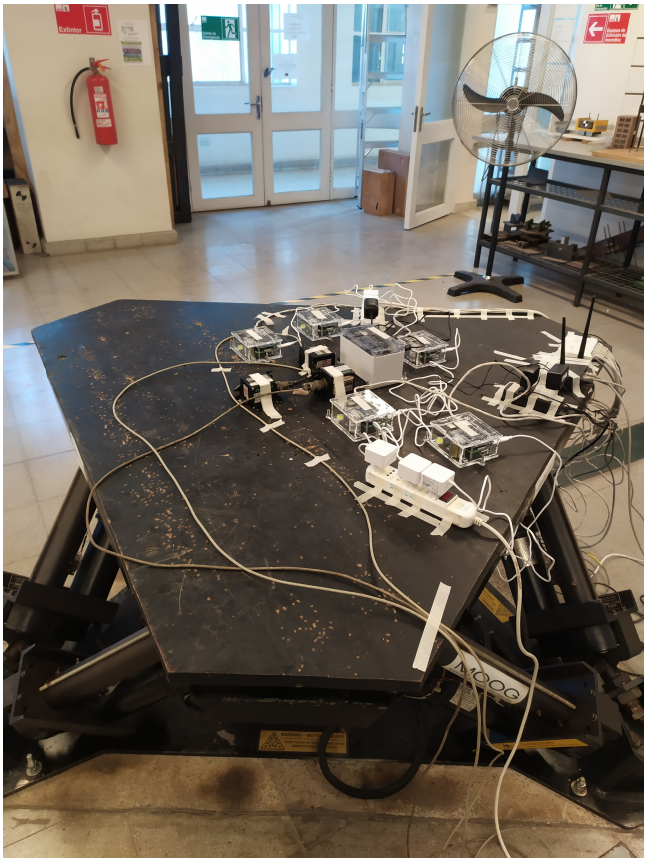

Figura 3.5: Comparación de los sensores en mesa vibradora.

Por consiguiente, todos los sensores fueron sometidos a los mismos registros sísmicos(indicados en la tabla), de donde se pudo realizar la comparación entre los sensores, obteniendo los siguientes resultados:

Tabla 3.1

	Registro Implementado	País
1	Terremoto de Constitución	Chile
2	Terremoto de Concepción	Chile
3	Terremoto de Kobe	Japón

Se muestran las comparaciones del sensor R4E16 como ejemplificación de lo que fueron las comparaciones.

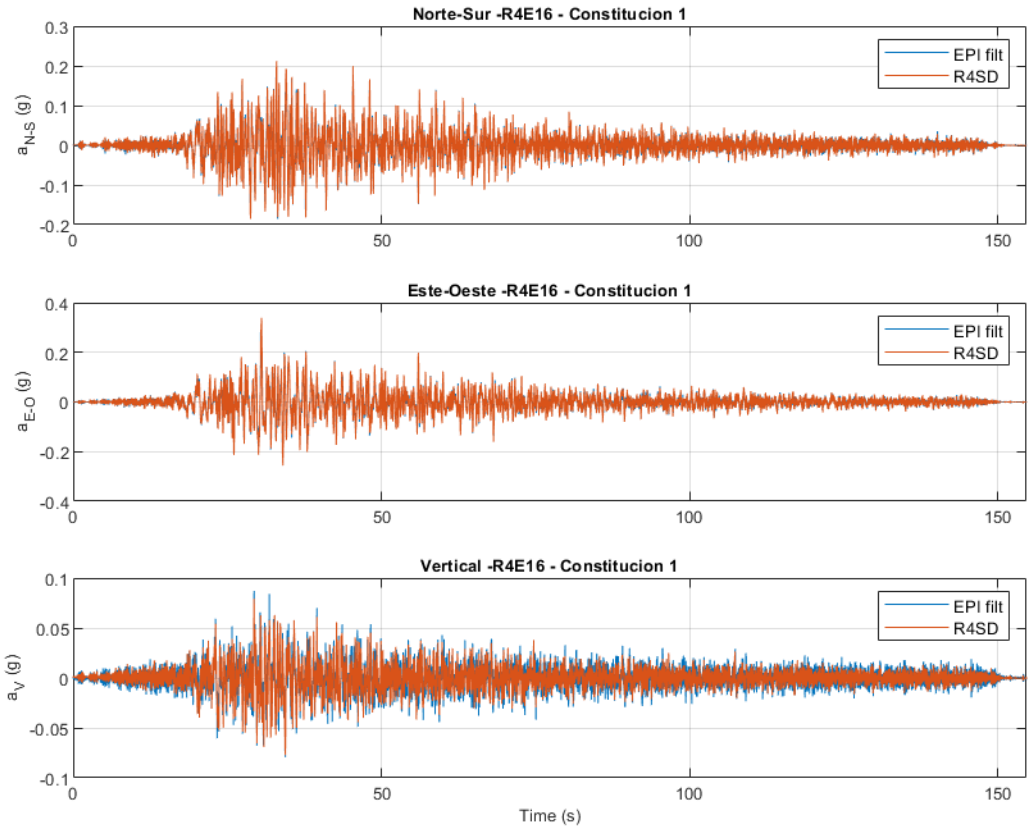

Figura 3.6: Respuesta de equipos comparadas al ser sometidos al registro sísmico de constitución en mesa vibradora.

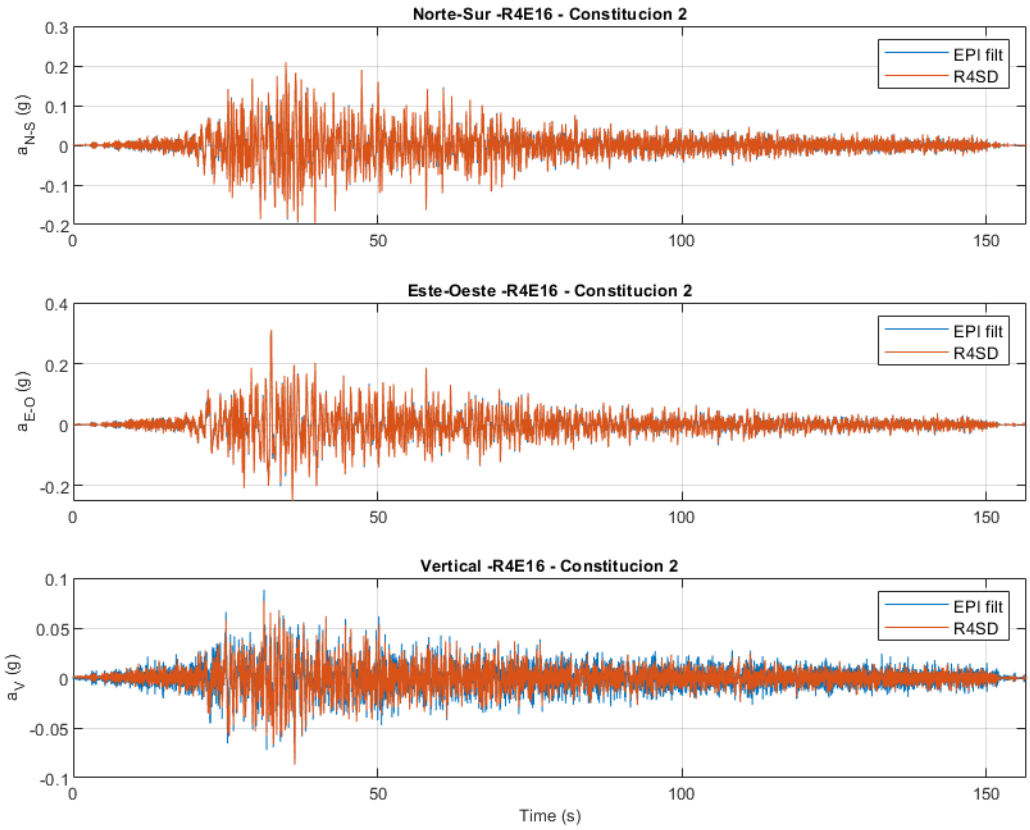

Figura 3.7: Respuesta de equipos comparadas al ser sometidos al registro sísmico de constitución en mesa vibradora.

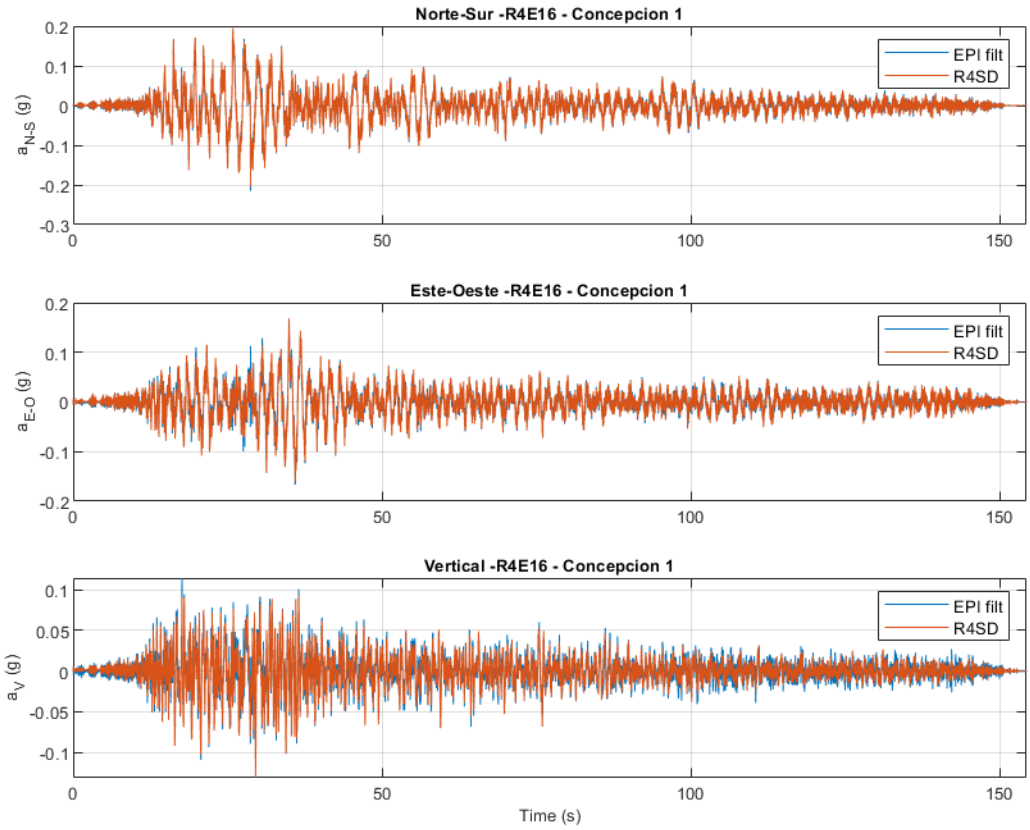

Figura 3.8: Respuesta de equipos comparadas al ser sometidos al registro sísmico de concepción en mesa vibradora.

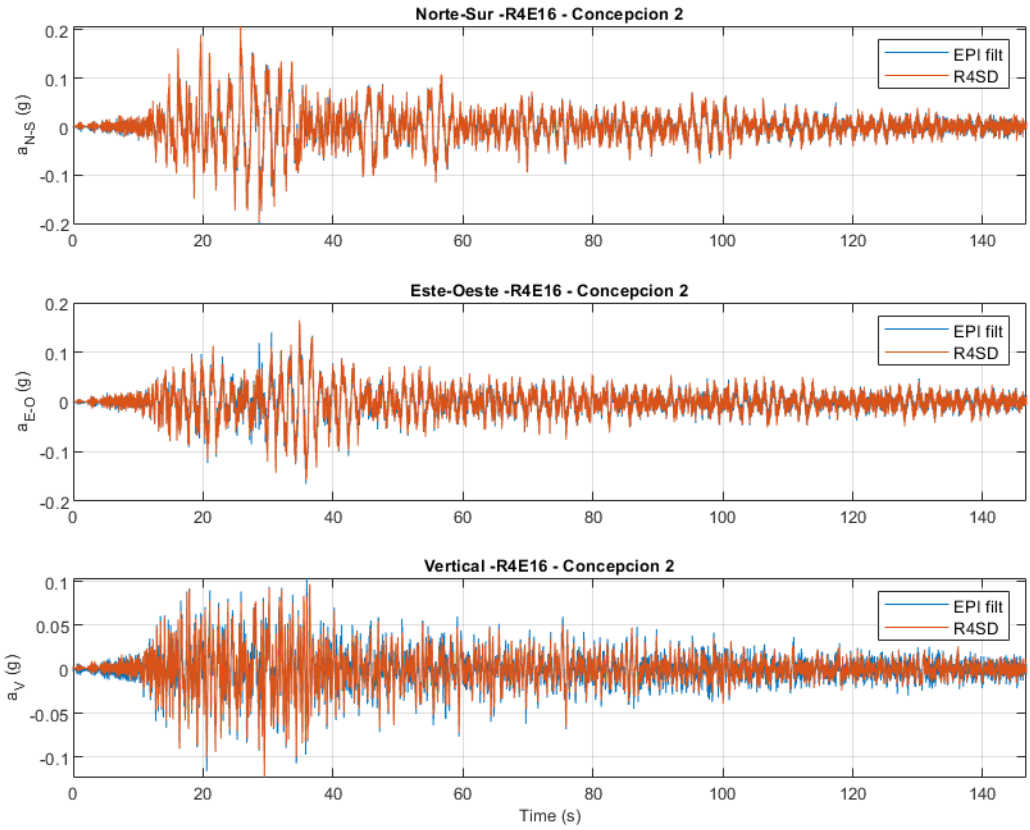

Figura 3.9: Respuesta de equipos comparadas al ser sometidos al registro sísmico de concepción en mesa vibradora.

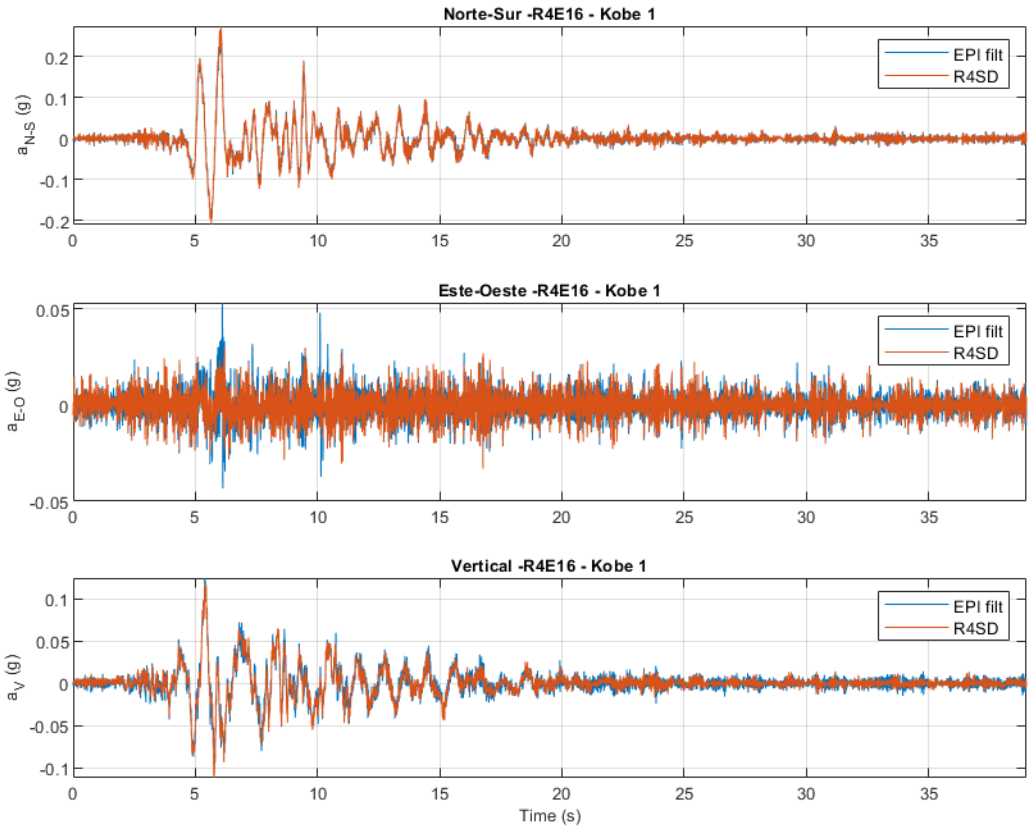

Figura 3.10: Respuesta de equipos comparadas al ser sometidos al registro sísmico de Kobe en mesa vibradora.

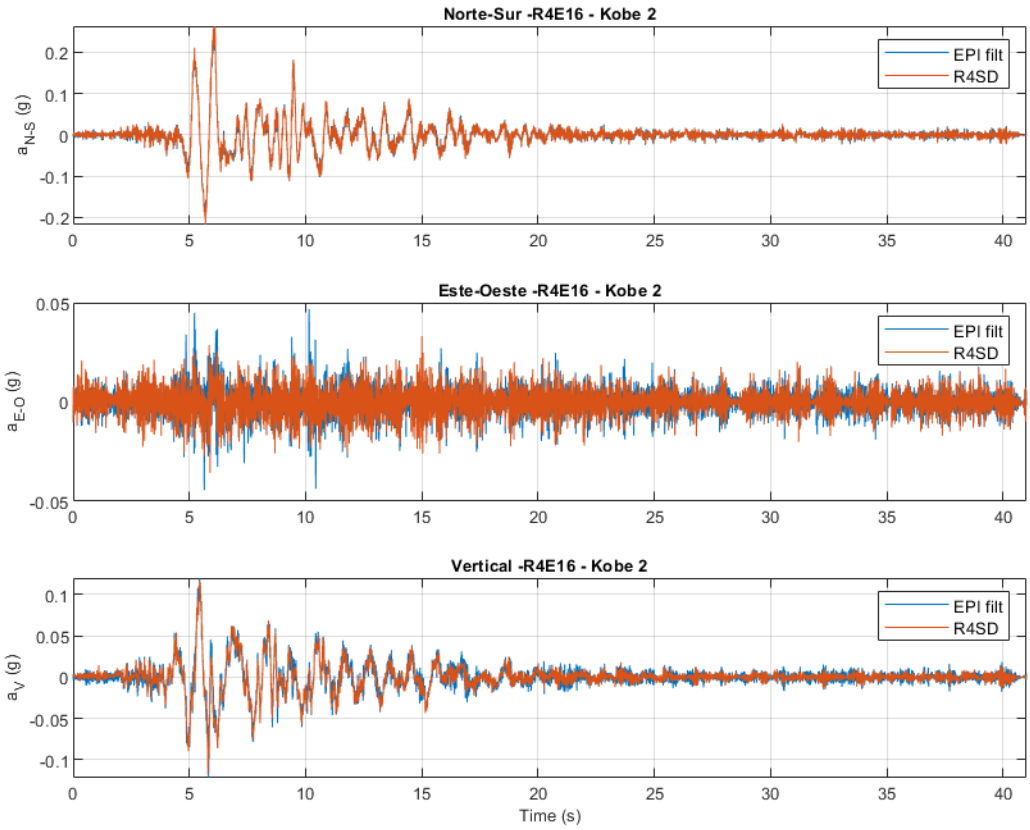

Figura 3.11: Respuesta de equipos comparadas al ser sometidos al registro sísmico de Kobe en mesa vibradora.

Luego estos resultados fueron analizados comparando su función de transferencia, desfase, coherencia y PSD:

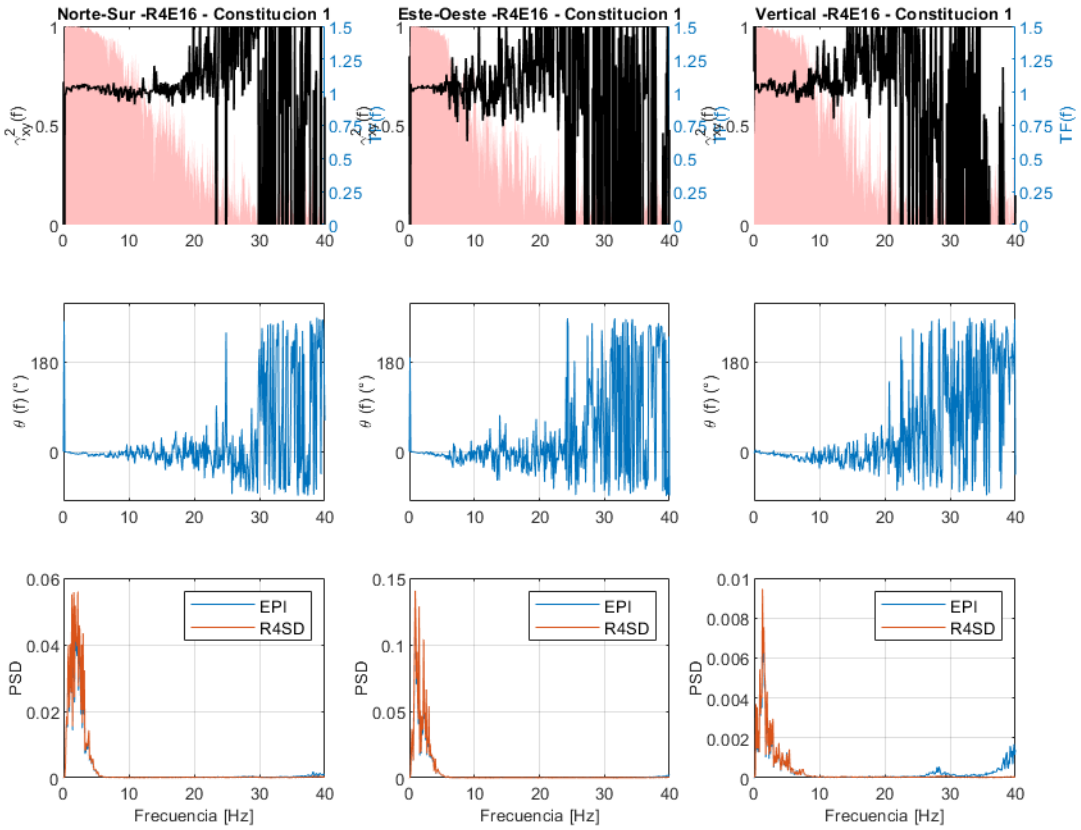

Figura 3.12: Análisis de resultados de respuestas a sismo de Constitución en sensores Episensor y RaspberryShake.

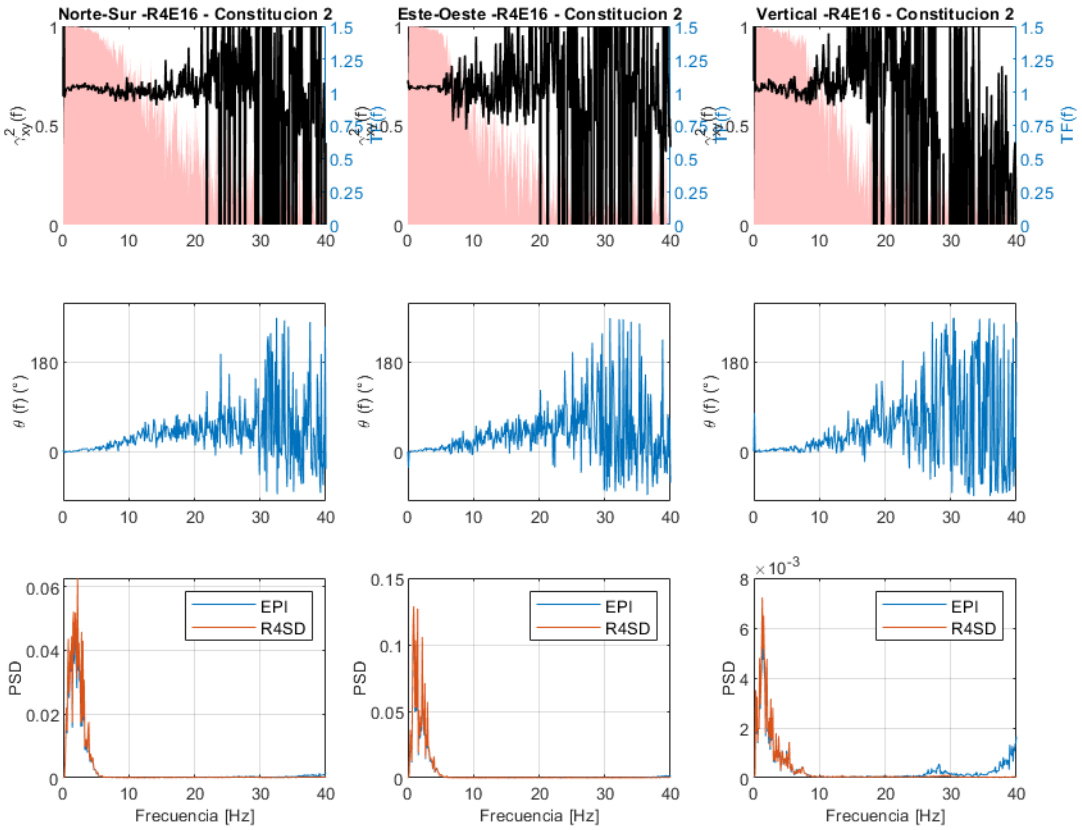

Figura 3.13: Análisis de resultados de respuestas a sismos de Constitución en sensores Episensor y RaspberryShake.

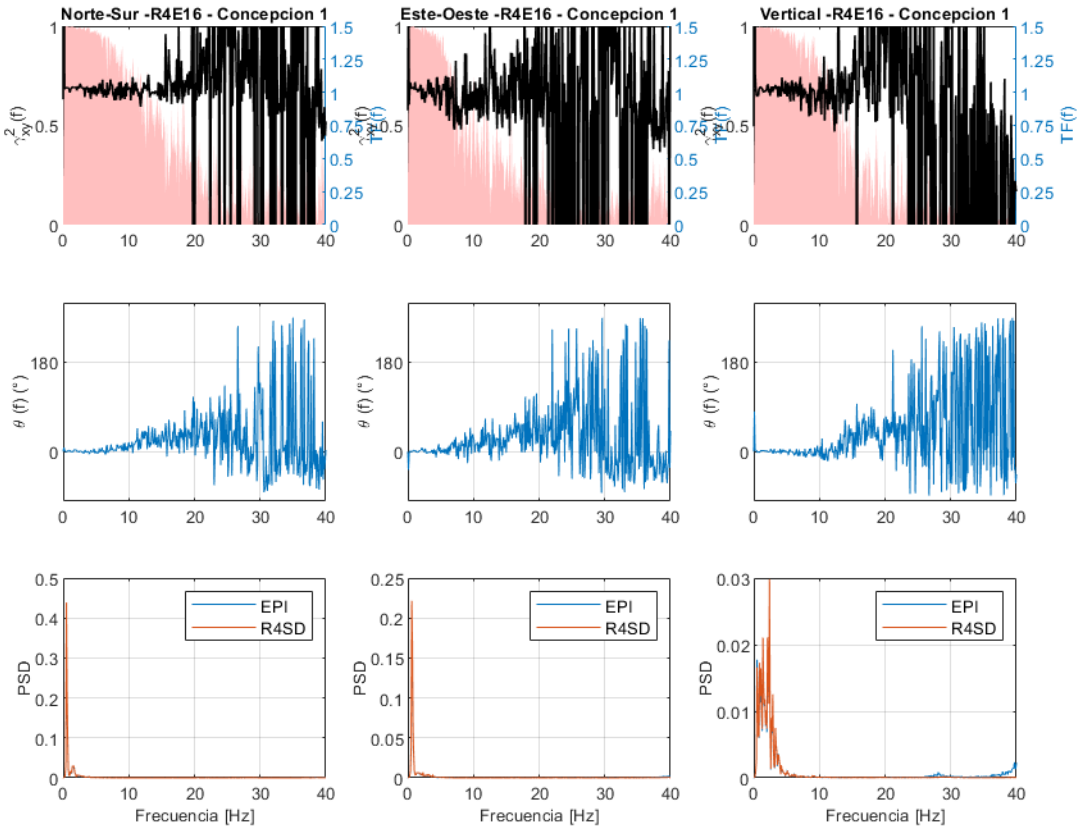

Figura 3.14: Análisis de resultados de respuestas a sismos de Concepción en sensores Episensor y RaspberryShake.

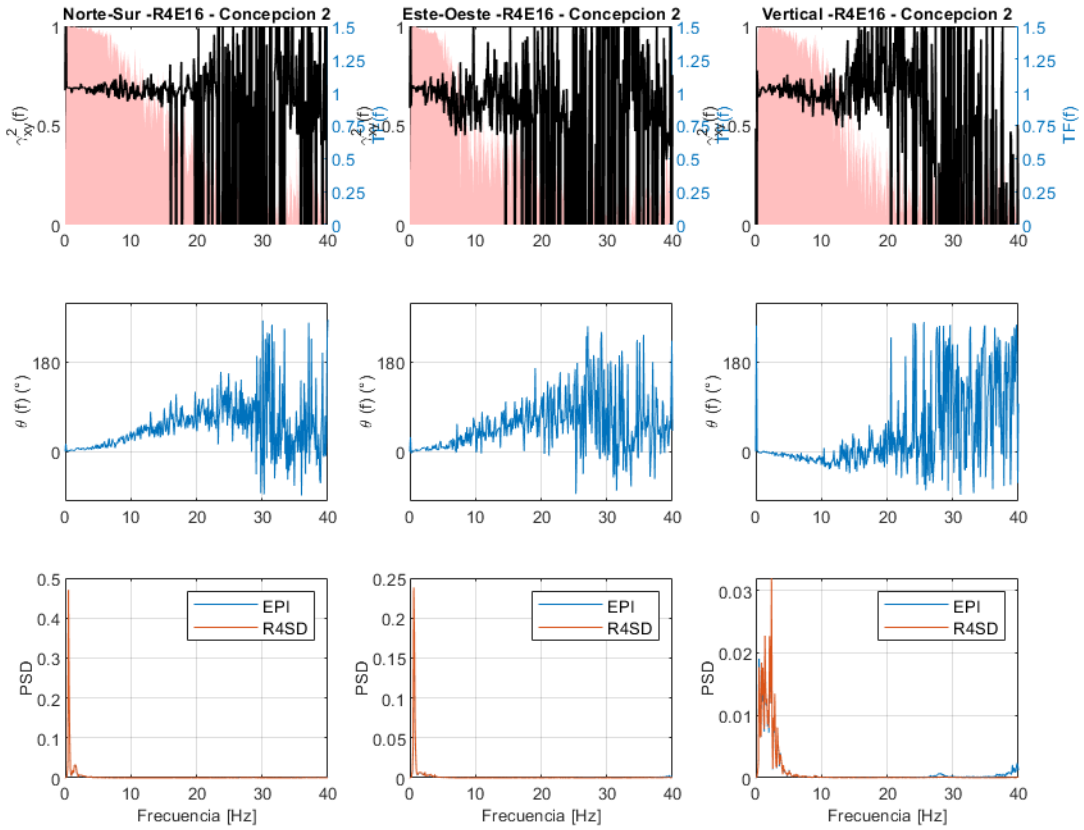

Figura 3.15: Análisis de resultados de respuestas a sismos de Concepción en sensores Episensor y RaspberryShake.

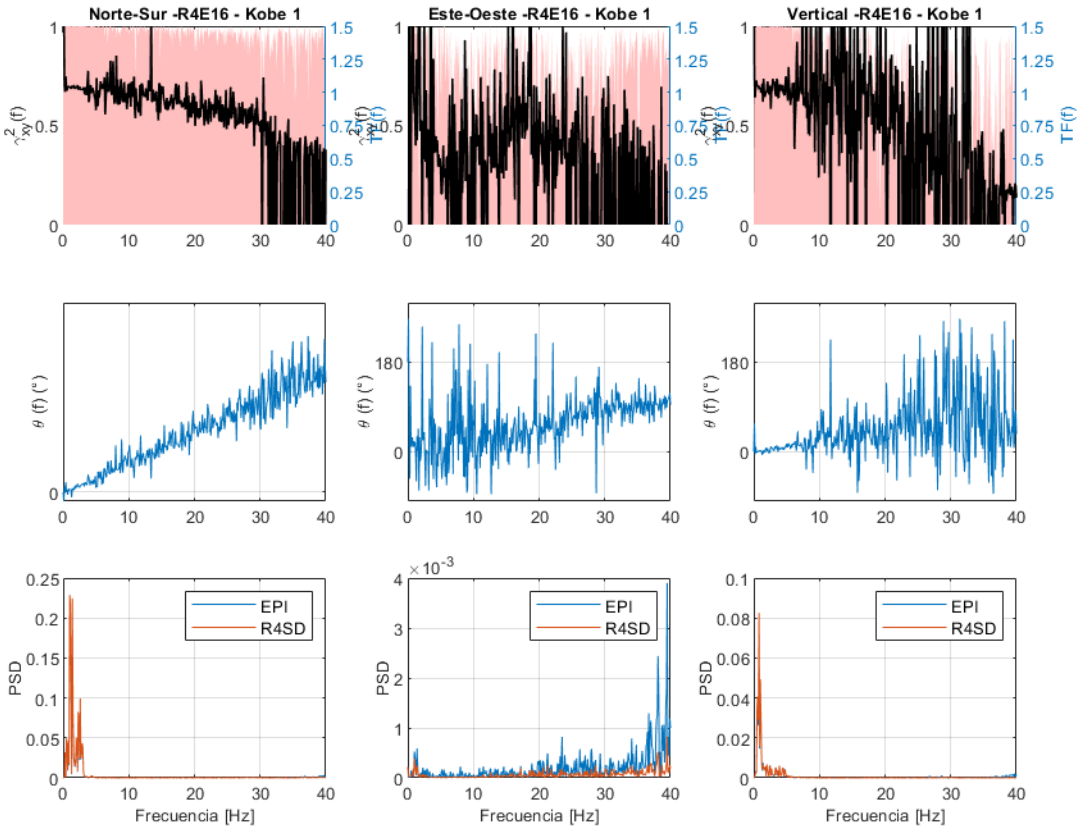

Figura 3.16: Análisis de resultados de respuestas a sismos de Kobe en sensores Episensor y RaspberryShake.

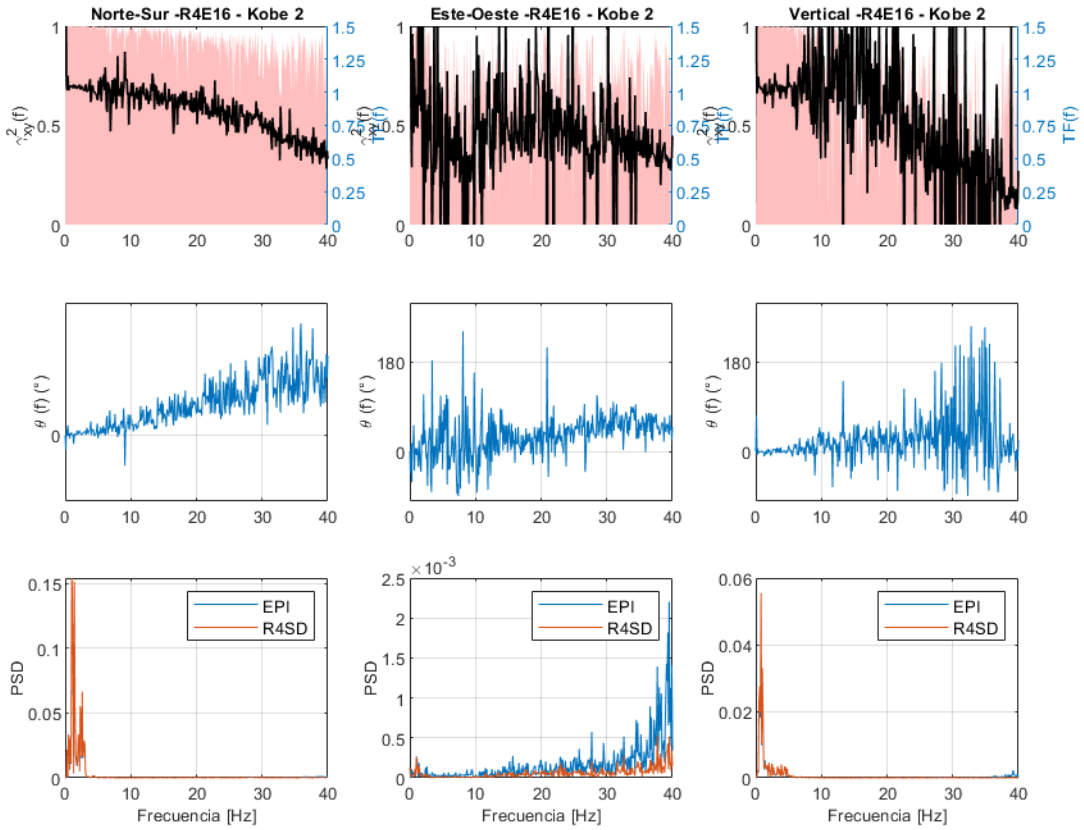

Figura 3.17: Análisis de resultados de respuestas a sismos de Kobe en sensores Episensor y RaspberryShake.

Notar que la mayor diferencia entre ambos sensores se nota en el componente E-O del registro dell movimiento sísmico de Kobe. Esto se produce debido que ese registro no posee esa componente, por lo que los registros efectuados en el eje dependen de la sensibilidad del sensor, pero todos muy cercanos a 0.

De las figuras de análisis se puede observar que la diferencia entre los sensores "Raspberry Shake" (línea naranja) y los "Episensor" (línea azul), es mínima en todas las direcciones encontrandose que los R4SD pueden reproducir en forma precisa señales de hasta una frecuencia de 20-25 Hz, generando un filtro para frecuencias mayores. Además se observo que en promedio los sensores registran una aceleración 2 % mayor a los registrados por los acelerometros Episensors.
Los espectros de potencia obtenidos en los sensores se acercan a los mismos peak de frecuencia y los factores de correlación se acercan en promedio a 1 en varias ocasiones.
De esta forma se puede proceder a instrumentar la torre con los instrumentos deseados, sin perjuicio de una mala respuesta a algún movimiento, por lo que se decide continuar con ellos.

Capítulo 4

Instrumentación

A continuación, se detalla la colocación de los sensores Raspberry Shake en la torre junto a información de tiempo y magnitud de los movimientos que han registrado.

4.1. Instalación de los Instrumentos

La torre fue instrumentada con 6 sensores de los previamente descritos. Cada uno fue registrado según su código que venía de fábrica con 5 caracteres.

Tabla 4.1: Disposición de los Instrumentos.

Número	Instrumento	Código	Ubicación
1	RS4D Outdoor	R3323	Pozo Exterior
2	RS4D Indoor	R81A0	2do Piso Esquina Sur
3	RS4D Indoor	R4E16	2do Piso Esquina Noreste
4	RS4D Indoor	RACCE	2do Piso Esquina Noroeste
5	RS4D Indoor	R5642	4to Piso Esquina Sur
6	RS4D Indoor	RD26F	4to Piso Esquina Noroeste

La disposición de los instrumentos se detalla en las figuras a continuación donde 5 de ellos fueron colocados dentro de la torre y 1 a las afueras de esta, en un pozo para medir de manera directa las ondas que llegaban desde el suelo y como reaccionó la estructura ante estas. Todos los sensores fueron colocados coincidiendo 2 de sus direcciones ortogonales con las direcciones longitudinal y transversal de la torre.

Se diseñó la instalación buscando suministrar los siguientes aspectos requeridos por los RS4D:

- Fuente de tiempo común para todos.

- Reloj de fuente de tiempo robusto.

- Mínimo uso de cable.

Figura 4.1: Ubicación en terreno de instrumento R3323.

Figura 4.2: Instrumento R3323 Outdoor en pozo.

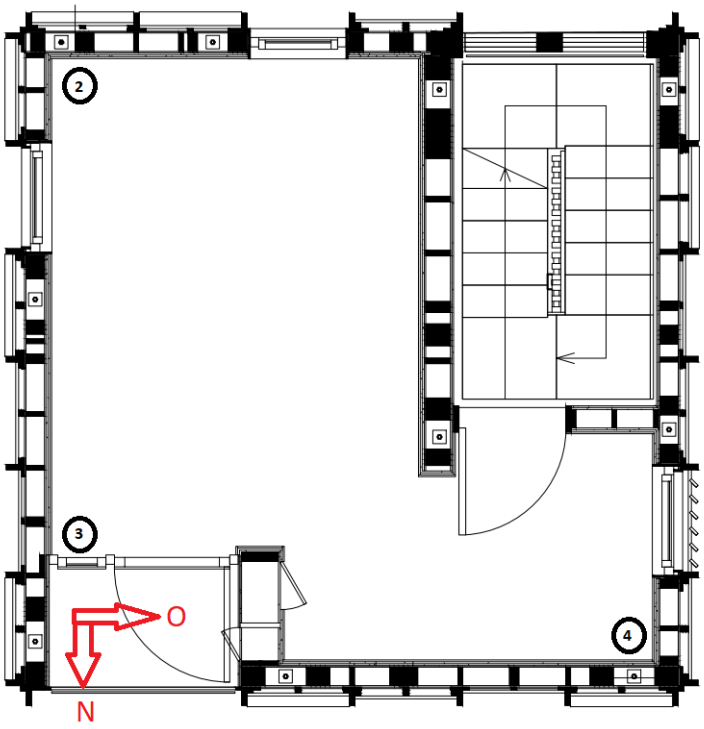

Figura 4.3: Ubicación en planta de los sensores 2, 3 y 4 ubicados en el 2do piso.

Figura 4.4: Sensor R81A0 en terreno.

Figura 4.5: Sensor R4E16 en terreno.

Figura 4.6: Sensor RACCE en terreno.

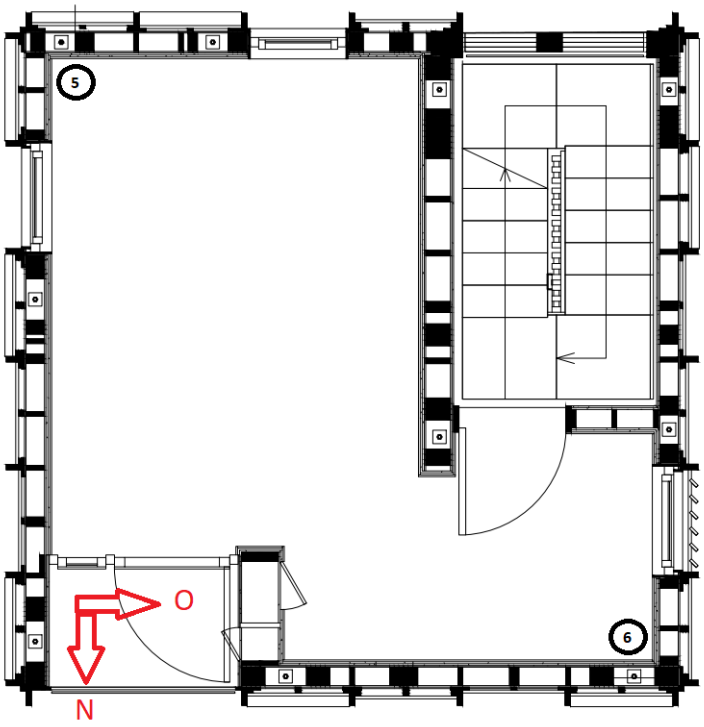

Figura 4.7: Ubicación en planta de los sensores 5 y 6 ubicados en el 4to piso.

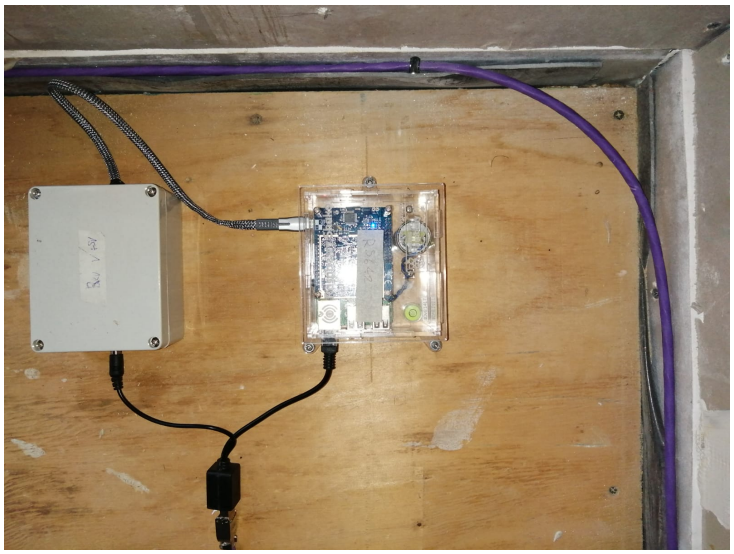

Figura 4.8: Sensor R5642 en terreno.

Figura 4.9: Sensor RD26F en terreno.

Se colocaron además 2 sensores QDR (de RENADIC) para medir la excitación en el pozo y otro en el 4to piso de la estructura junto al sensor nº6 (para evaluar el correcto funcionamiento de los equipos R4SD).

4.2. Preparación

4.2.1. Montaje en terreno

Cada sensor Raspberry Shake venía con un conector Micro USB(5V - 2.5 Amp) para conectarse a la corriente y un puerto de conexión Ethernet para conectarse a la red de internet.

Estas conexiones fueron intervenidas para conectarse a través de un único cable TRIMERX Categoría 6A U/FTP (2) que transporta tanto corriente como la información de red y así evitar cablerío innecesario en terreno. De esta forma los cables se alimentan de una batería de 12 V que actúa como fuente de energía y la comunicación se posiciona en un router o switch.

La conexión del conector micro USB fue a su vez modificado para pasar a conectarse directamente a la corriente a través de una batería de 12V 55Ah ciclo profundo KBL 12260W marca Kaise. Estos se conectaron a "cajas transformadoras de voltaje" dentro de las que se conectarían un "Módulo Step Down Ajustable LM2596S"(3) que actuaría como convertidor de voltaje para pasar de los 12V entregados por la batería a los 5V que necesitan los sensores y los 10 V requeridos por el router, un "Fusible de 5x20 mm 3A" como método de protección y un "Conector 5.5x2.1 con bloqueo" para terminar la intervención de lo que sería la parte de corriente que sale del sensor Raspberry Shake.

Finalmente se juntaría el cable de red "Ethernet" y la caja transformadora a través de un Inyector PoE(4).En este se juntaría el transporte de red y corriente, y cuya conexión a un

conector RJ45(5) permitiría a su vez la conexión al cable TRIMERX antes mencionado.

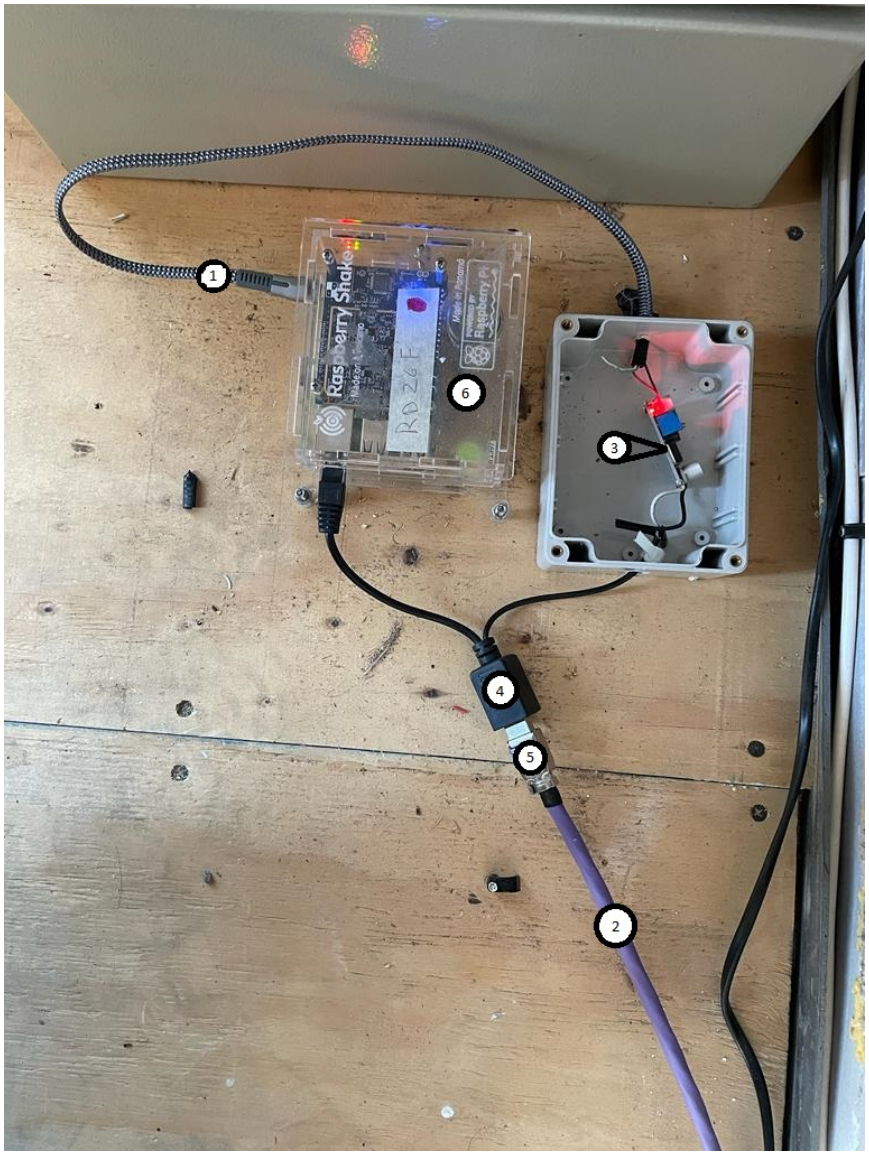

Figura 4.10: Sistema que transporta información y convierte el voltaje que llega desde la batería.

En la figura 4.10 se pueden apreciar fácilmente todos los componentes:

Tabla 4.2: Elementos de la figura 4.10

Número	Elemento
1	Cable micro USB corriente.
2	Cable TRIMERX Categoría 6A U/FTP.
3	Módulo Step Down Ajustable LM2596S
4	Inyector PoE
5	RJ45
6	Raspberry Shake

La conexión entre el cable TRIMERX y el conector RJ45 sigue en cada pin los estándares T-568B. De esta forma se puede saber con el color de cada cable donde va ubicado:

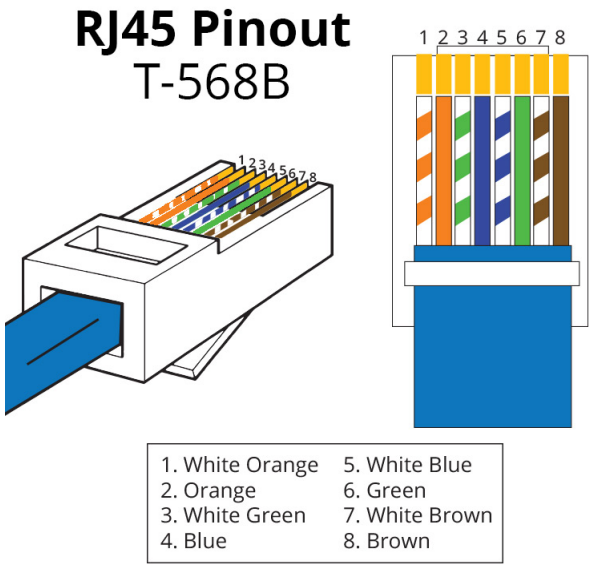

Figura 4.11: Conexión bajo estándar T-568B.

La batería se mantendrá cargada en todo periodo en el que haya luz en la torre, y en caso de que alguna eventualidad sísmica o de cualquier otro tipo corte la corriente, los aparatos aseguren continuar funcionando y puedan registrar todo movimiento. De esta forma la batería y su cargador actúa como una UPS que mantiene la alimentación eléctrica por más de 24 hrs sin alimentación.

En la práctica hubo varios problemas eléctricos en la torre peñuelas que fueron causa de inestabilidad de operación.
En primera instancia, se detectó que los fusibles elegidos eran de baja calidad e impedían el correcto flujo de energía, por lo que después de varios intentos de arreglar el paso de corriente, se decide quitarlos. Igualmente se quitan los conectores 5.5x2.1 ya que sin 2 elementos que conectar ya no eran necesarios. Con esto los instrumentos dejaron de medir de manera cortada y comenzaron a hacerlo de manera continua en todo momento que tuvieran corriente.

Otro problema encontrado fue en el fue en el cargador de batería inteligente de control automático y manual 4/12/25 Amp de la marca “Black amp; Decker”. El problema en primera instancia consistía en que una vez que llegaba a la capacidad total de la batería, dejaba de funcionar arrojando el error “F05” en el que “El modo de mantenimiento cambia al modo de carga por flotación” cuya causa indica que la batería está dañada o la corriente de carga es demasiado baja.
Ante esta situación se coloca el cargador en su modo de corriente más alto (de 25A). Sin embargo el problema persiste, por lo que fue necesario estar constantemente visitando la torre para analizar los problemas de alimentación teniendo que desenchufar y enchufar nuevamente el cargador.
Ante esta problemática se decide utilizar un Temporizador Timer Digital para que una vez al día el cargador se apague por 30 minutos y se vuelva a encender. De esta forma, si el problema se repite, se apagará y encenderá solo, permitiendo que el cargador cargue el sistema y que se resetee cuando ocurra el error F5 del cargador.

Otro de los posibles errores que se manejó a la hora de buscar el error fue que hubiese un error de tipo eléctrico en la torre. Después de experimentar se determinó que podía ser producto de que se había enchufado el cargador a un enchufe defectuoso. Por esto se utiliza una regleta o alargador para conectar todo el sistema a otro enchufe más lejano perteneciente a la red de enchufes interiores y que se comprobó su buen estado.

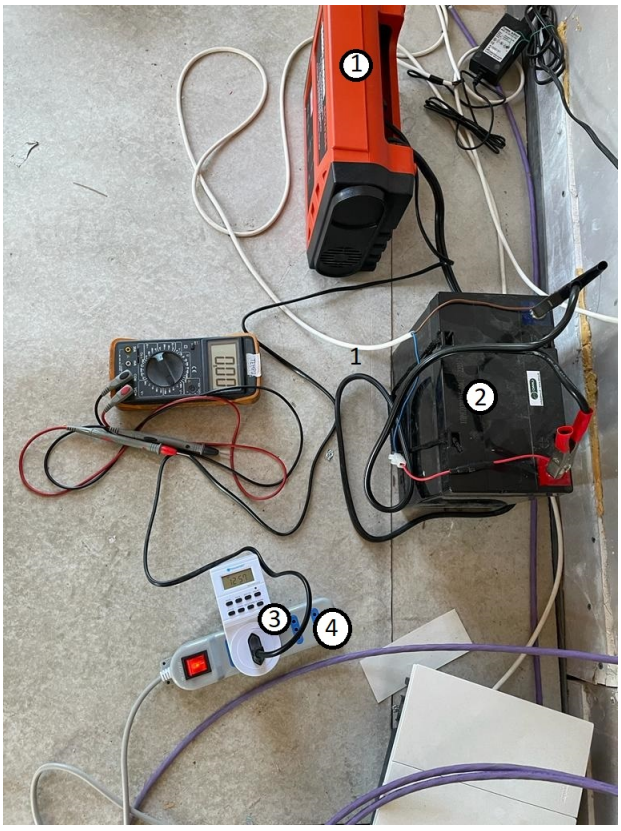

Figura 4.12: Solución final estable para operación.

En la imagen se pueden identificar:

Tabla 4.3: Elementos de la figura 4.12

Número	Elemento
1	Cargador de Batería Inteligente de Control Automático y Manual 4 / 12 / 25Amp Black+Decker
2	Batería de 12V 55Ah ciclo profundo KBL 12260W
3	Temporizador Timer Digital
4	Regleta

4.2.2. Visualización de Datos

Para la parte digital de la instrumentación se conectaron todos los sensores Raspberry Shake a la misma red a través de la conexión Ethernet a un Router "N300 DIR-615N" de 8 puertos de marca Dlink, en el que también se conectó una "Raspberry Pi 3". Este se conectaría a su vez a una BAM "Huawei E8372" de la marca Entel, el que permitiría el acceso de internet de forma precargada (comprando bolsas de internet móvil) y habilitaría el acceso de forma remota a la red de sensores a través de un software de escritorio remoto.

Es importante destacar que para evitar que eventuales problemas del Módem BAM comprometan el funcionamiento de la Raspberry Pi, se programó a través de crontab en la misma Raspberry con el comando "sudo crontab -e" para que la misma Raspberry se reinicie a las 16:00 y a las 23:59. De esta forma, se protege el sistema ante posibles dificultades que pueda causar el Módem.

```
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
59 23 * * * sudo reboot
59 15 * * * sudo reboot
```

Figura 4.13: Programación tareas automáticas a través de crontab.

Así, se utilizaría esta Raspberry Pi para monitorear desde la distancia y descargar los datos de los sensores sin la necesidad de ir a la torre.

Para facilitar este mismo proceso de monitoreo se instalan diversas aplicaciones. La primera de ellas es el navegador, en este caso se utilizó Google Chrome, el cual a través de la digitación de la IP de los sensores en el buscador, se puede ingresar a los equipos y su controlador principal por defecto.

En este controlador se puede verificar el estado inicial de los equipos, si se encuentran registrando, conectados al servidor, etc.
Es importante en este apartado destacar que los equipos fueron configurados en este apartado para registrar datos de máximo 30 días y luego ser sobreescritos automáticamente. Esto se puede lograr en el apartado "Configuración>Datos>Tiempo de almacenamiento de las formas de onda" (configuración en el ícono de tuerca en la figura 4.14).
El hecho se que se borren datos periódicamente es la razón por la cual es necesario tener un

sistema que guarde los datos sísmicos cuando un cierto umbral de aceleraciones es superado.

Una forma de verificar rápidamente que los datos se están registrando y de manera continua, es a través del uso de la herramienta “Helicorder” que se encuentra en el mismo servidor (abajo a la izquierda en la figura 4.14) y muestra los datos por día registrado.

Figura 4.14: Servidor de RaspberryShake al colocar IP en el buscador del navegador.

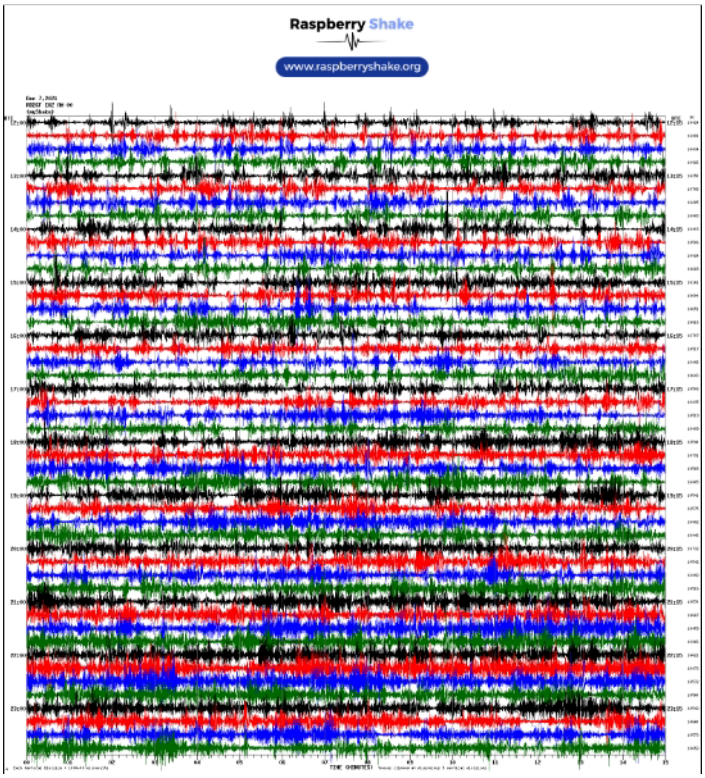

Figura 4.15: Datos registrados durante 12 horas el día 08/12/21 en el sensor RD26F vistos desde Helicorder.

Para saber que IP tiene cada Raspberry, es necesario acceder a la configuración del Router y ver los elementos conectados y su respectiva IP, observándose que el router fue configurado para proporcionar una IP fija a los sistemas conectados (Router, Raspberry, Computador externo, R4SD).

Tabla 4.4: Dirección IP fijada de cada instrumento

Instrumento	Dirección IP
R3323	192.168.0.104
R81A0	192.168.0.192
R4E16	192.168.0.169
RACCE	192.168.0.110
R5642	192.168.0.191
RD26F	192.168.0.189
Raspberry Pi 3	192.168.0.106
Router	192.168.0.1

Un segundo software necesario de instalar en la Raspberry Pi 3 es SWARM, que permite observar el registro sísmico de los equipos en tiempo real, además de algunos análisis de frecuencias también en tiempo real.

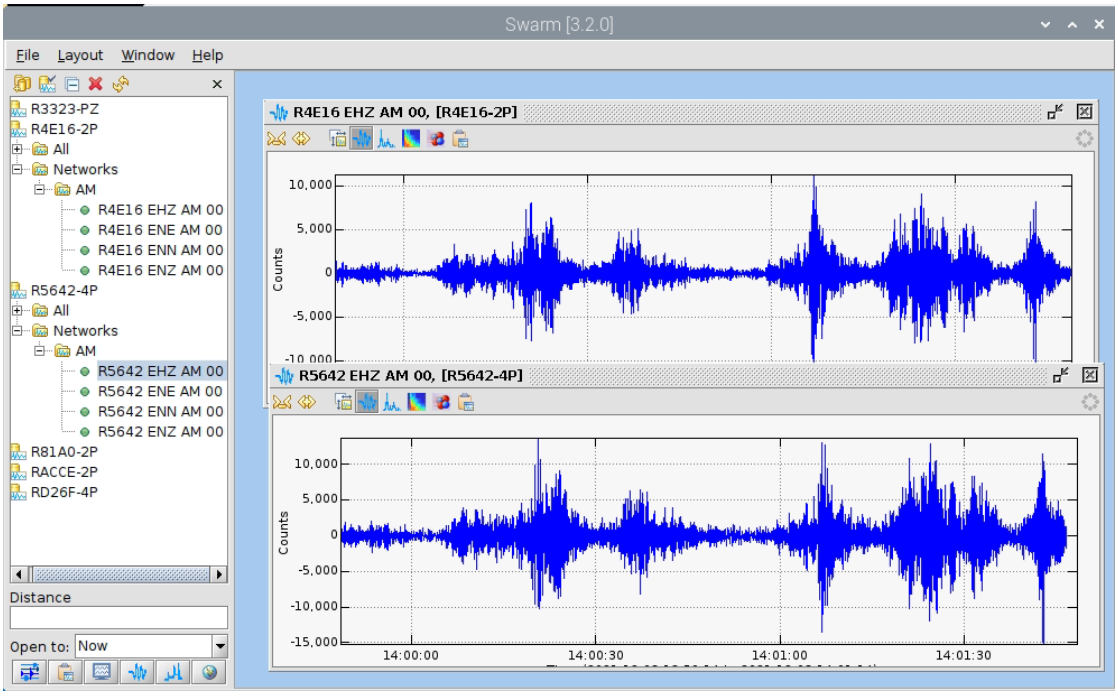

Figura 4.16: Datos registrados en tiempo real vistos a través de SWARM.

Para proveer un tiempo común a los sensores de manera de obtener registros sincronizados (incluso cuando no haya internet en el sistema), se instaló un reloj “DS3231” a la Raspberry Pi auxiliar, para así sincronizar a todos los sensores a la misma hora y lograr que los registros no esten desplazados en el tiempo. También se debió hacer que la fuente de tiempo estuviera asociada a una IP que no cambiara en el tiempo, para así trabajar con la estación y programar códigos que permitan la recolección de datos de manera más sencilla.
Este proceso de sincronización de los datos se especifica en la sección 4.2.3.

El tercer programa que se instala es PuTTY, para el que se ingresa la dirección IP de los equipos junto a su usuario y contraseña asignados.
En este caso se dejó el nombre de usuario y contraseña que vienen por defecto en los Raspberry Shake que son “myshake” y “shakeme” respectivamente.
Con esto se puede tener control de los equipos R4SD desde la Raspberry auxiliar, lo que permite saber si se encuentra sincronizado en el tiempo con los equipos (a través del comando “timedatectl”), saber si se encuentra sincronizado a la Raspberry Pi auxiliar, que a su vez define su tiempo del reloj adherido, y cuanto es su desfase(a través del comando “ntpq -p”) y otras utilidades como programar tareas en el sensor o ver los archivos que guarda.

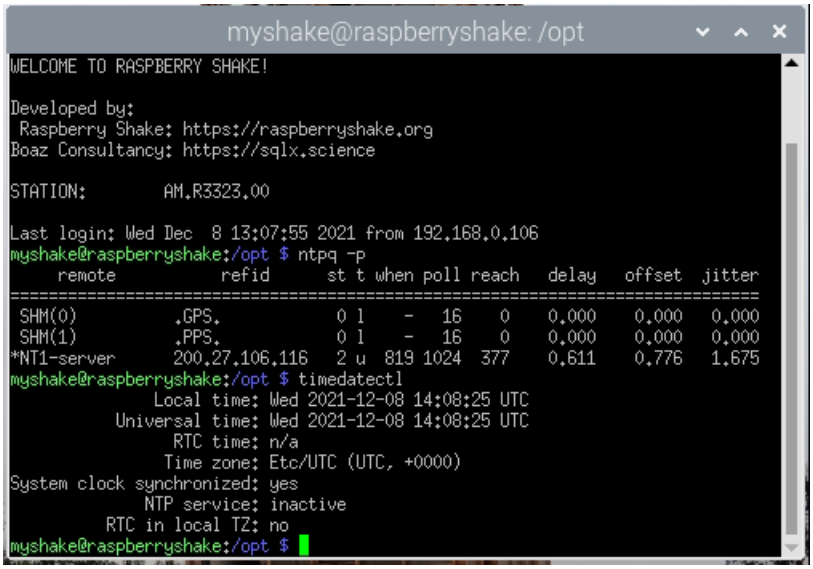

Figura 4.17: Interfaz programa PuTTY.

Notar que para que efectivamente se encuentre sincronizado el reloj, en la figura 4.17 debe aparecer el servidor al que se sincroniza marcado con un *. En este caso está sincronizado a NT1-Server y se puede visualizar una diferencia(offset) de 0,776 milisegundos.

Otra aplicación importante es “FileZilla”, el que permite a través de la dirección IP, usuario y contraseña de cada equipo; ingresar a estos y poder extraer los archivos de datos abriendo la carpeta “opt>data>archive>202X>AM>SENSOR”.
Con los datos pasados a la Raspberry Pi 3 se puede, a través del software de escritorio remoto y sus herramientas, pasar al computador de la persona que quiera bajarlos.
Notar que los archivos vienen en formato miniseed (que es leído desde MATLAB empleando la función que se muestra en el anexo A.2)

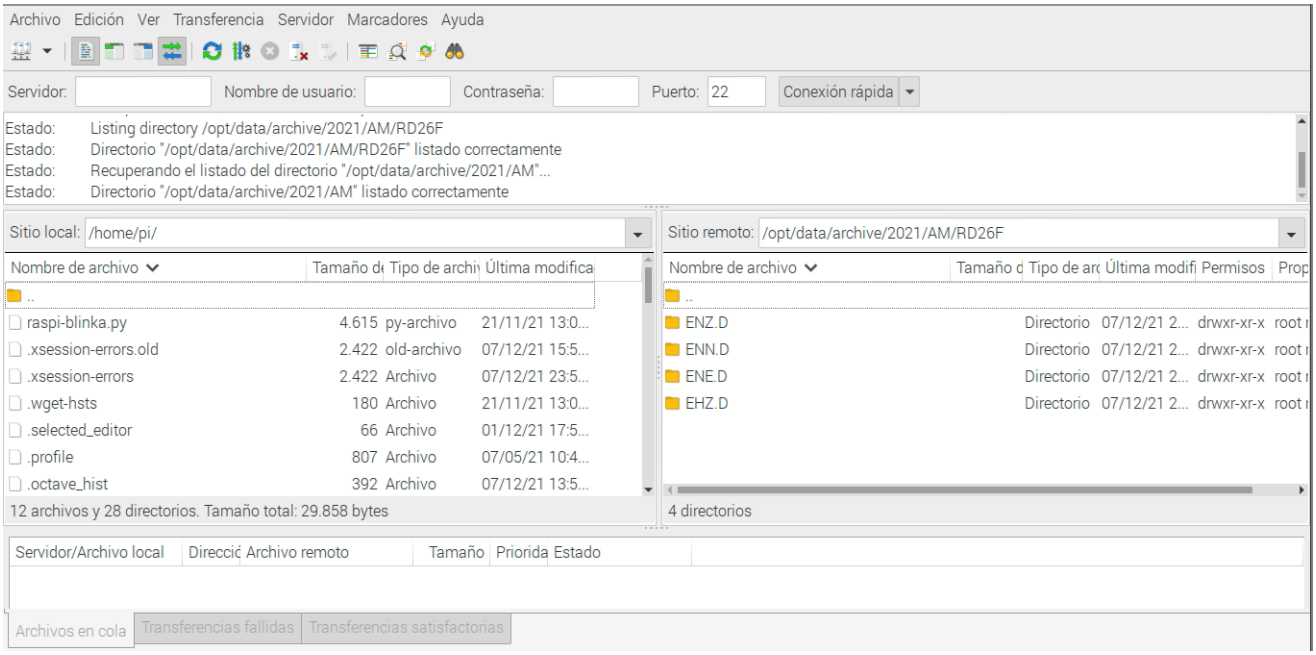

Figura 4.18: Interfaz programa FileZilla.

Finalmente, el día 29/11/2021 se le instala un sensor de temperatura y humedad "BME280" a la Raspberry Pi para obtener como datos adicionales los parámetros atmosféricos al interior de la estructura sin necesidad de revisar las condiciones ambientales en la estación meteorológica más cercana.
Con esto se programa en Octave un código que permite procesar, a cada hora del día, los datos obtenidos de la última media hora, y registrar las primeras tres frecuencias naturales obtenidas por el método FDD y registrar la temperatura y humedad de ese momento.
Este código se puede encontrar en el Anexo A.11

Este sensor se conecta directamente junto al reloj que se instaló anteriormente en la Raspberry Pi, de manera que quedan conectados en serie.

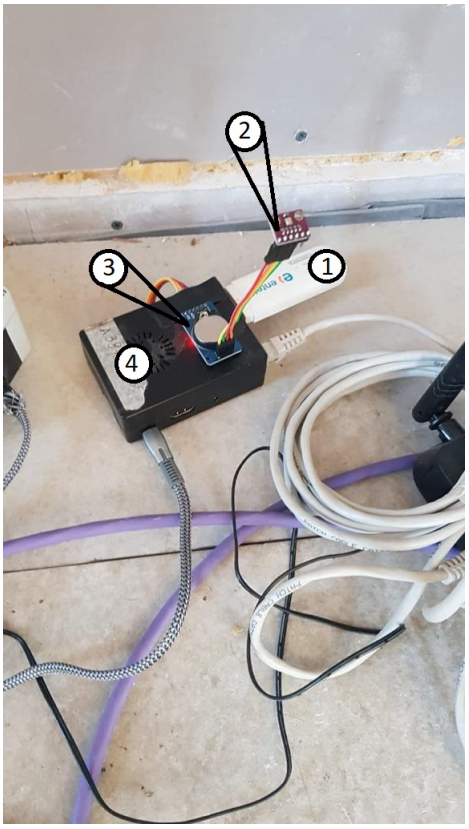

Figura 4.19: Raspberry Pi con el sensor, reloj y Modem instalados.

Donde se puede identificar en la imagen:

Tabla 4.5: Elementos de la figura 4.19

Número	Elemento
1	BAM Entel "Huawei E8372"
2	Sensor de Temperatura y Humedad "BME280"
3	Reloj DS3231
4	Raspberry Pi 3 auxiliar

Para la instalación del software del sensor de temperatura y poder registrar datos se coloca el comando “pip3 install adafruit-circuitpython-bme280” en la terminal [12]. Con esto se puede empezar a registrar datos a través de la programación como los códigos mostrados en el anexo A.13.

4.2.3. Sincronización de Datos.

En primer lugar se debe instalar un sistema NTP (Network Time Protocol) en la Raspberry pi, que sirve para proveer de tiempo común a todos los equipos conectados a una red local.
Para esto se utiliza el comando “sudo apt-get install ntp”.

Se configura de donde obtendrá la hora la Raspberry Pi accediendo a su terminal y utilizando el comando “cd /etc” seguido de “sudo nano ntp.conf”.
Así, se agrega a los que vienen por defecto, el servidor del shoa en caso de que la raspberry se encuentre conectada a internet a través del comando “server ntp.shoa.cl iburst” y se le agrega “prefer” para que este servidor sea el predeterminado en caso de tener conexión a internet.
Para el caso de no tener intenet, se programa para que se busque la hora de manera interna en el reloj DS3231 utilizando el comando “server 127.127.1.0” y a continuación “fudge 127.127.1.0 stratum 10”, siendo esta IP la entrada de los terminales asociados al reloj.

De esta forma la Raspberry PI auxiliar toma tiempo NTP desde el servidor SHOA (actualizando al reloj XXX) y en caso de no existir internet, el tiempo es definido por el reloj agregado. Es de mencionar que los equipos Raspberry (auxiliar o Shake) no tienen un reloj interno, razon que hace que cuando no tengan alimentación electrica no mantengan la hora. Es decir, una vez reencendidos continuan con su hora anterior. Razón por la cual es imperativo proveer de tiempo al sistema. Más aún puede existir un desfase de los relojes internos que puede generar la desincronización de los datos. Motivo por el que es necesario que las R4SD tomen tiempo común que es provista por la Raspberry Pi auxiliar. Una solución alternativa sería proveer a cada R4SD de internet para que ajusten sus tiempos en forma individual, sin embargo esta solución no es idónea porque eso requiere de colocar una red local de internet y además deja de ser funcional ante la ausencia de internet o electricidad.

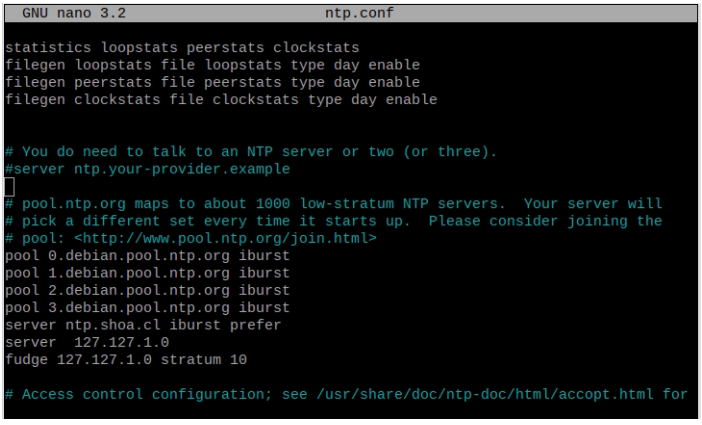

Figura 4.20: Configuración ntp server.

Con esto se sigue más abajo para especificar el buscar la hora de manera de local con el comando “broadcast 192.168.0.255” seguido de “broadcast 192.168.0.106” para especificar que busque la hora en la Raspberry pi.
Notar que para este caso se utiliza 192.168.0.106 ya que es la ip de la Raspberry Pi 3.

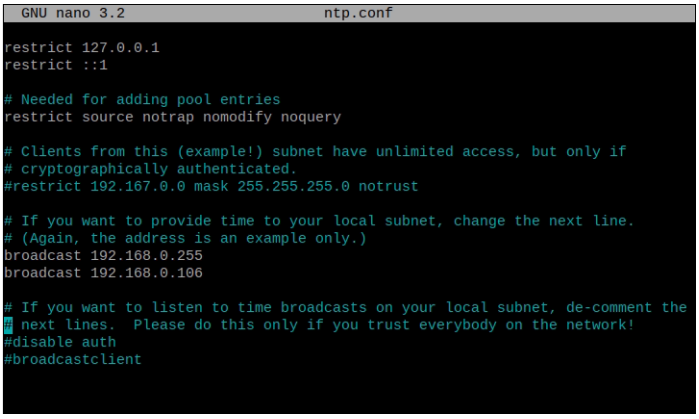

Figura 4.21: Configuración ntp server.

A continuación, se reinicia el servidor con “sudo service ntp restart” y se verifica que esté funcionando utilizando “sudo service ntp status”

Finalmente se deja que las demás Raspberry Shake puedan acceder a la Raspberry Pi a buscar la hora del sistema con “sudo ufw allow from any to any port 123 proto udp”.

Por otro lado, se necesita configurar las Raspberry Shake para que utilicen este tiempo que se ha configurado en la Raspberry Pi.

Primero se debe instalar el servidor ntpdate en los sensores utilizando el terminal a través de PuTTY, o conectando cada uno en forma individual a un computador, conectándolos a internet y colocando en cada uno “sudo apt-get install ntpdate”.

Luego, se define en las Raspberry Shake que el lugar de donde se obtendrá la hora es la Raspberry pi a través de la definición de la IP de esta última y el nombre que se le definirá a la red, en este caso “NT1-server”.

Para entrar a determinar esto último, se debe entrar a modificar la carpeta “hosts” a través de los siguientes comandos:
“cd /etc”
“sudo nano hosts”

Figura 4.22: Definición de Raspberry pi como el que actualiza la hora para la red en las Raspberry Shake.

Luego se desactiva la sincronización del tiempo en la Raspberry shake utilizando "sudo timedatectl set-ntp off" y se instala el servidor ntp con "sudo apt-get install ntp".

A partir de aquí, se configura la red a través de "cd /etc" seguido de "sudo nano ntp.conf". Aquí, bajo la misma lógica que se usó en la Raspberry pi, se coloca que los Raspberry Shake extraigan la hora desde el servidor "NT1-server" y que se prefiera esto a las demás opciones.

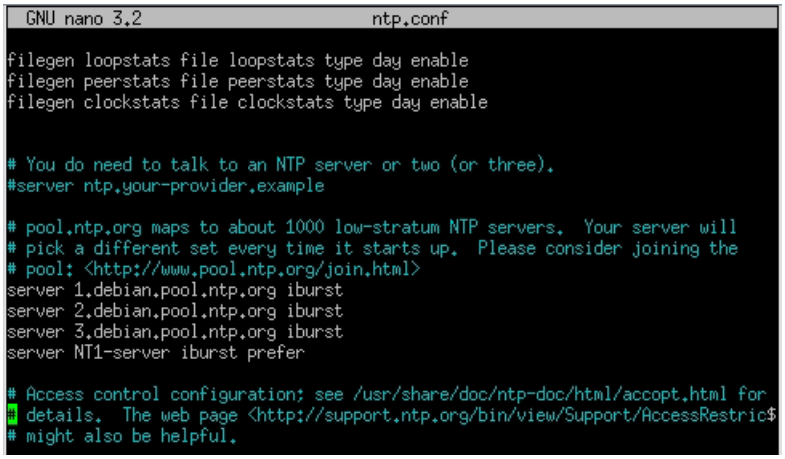

Figura 4.23: Definición de del servidor NT1-server como el predefinido.

Después se desconecta la sincronización propia y se configura para obtener la hora del reloj interno que posee la Raspberry pi auxiliar a través de los siguientes comandos:
"Disable auth"
"Broadcastclient"
"server 127.127.28.0 minpoll 4 maxpoll 4 noselect"
"fudge 127.127.28.0 time1 0.500 refid GPS"
"server 127.127.28.0 minpoll 4 maxpoll 4 prefer"
"fudge 127.127.28.0 refid GPS"

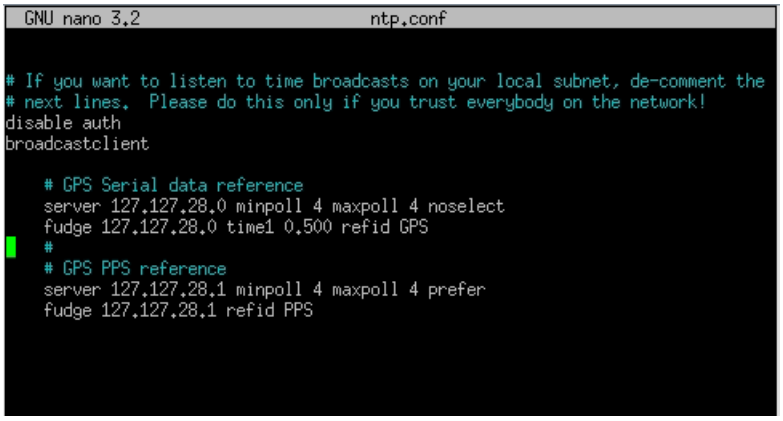

Figura 4.24: Definición de de que se obtendrá la hora de otro lado.

Para terminar la configuración de los Raspberry Shake, se reinicia el servidor ntp a través de “sudo service ntp restart”.

La sincronización de los datos se termina con la configuración del reloj “DS3231” instalado.

En primer lugar, se debe instalar el software para identificar en que puerto está conectado el reloj. Esto se hace a través de “sudo apt-get install python-smbus i2c-tools”. A continuación, se revisa el puerto con “sudo i2cdetect -y 1”.

Luego se comienza la configuración utilizando “sudo nano /boot/config.txt” y agregando al final “dtoverlay=i2c-rtc,ds3231”.

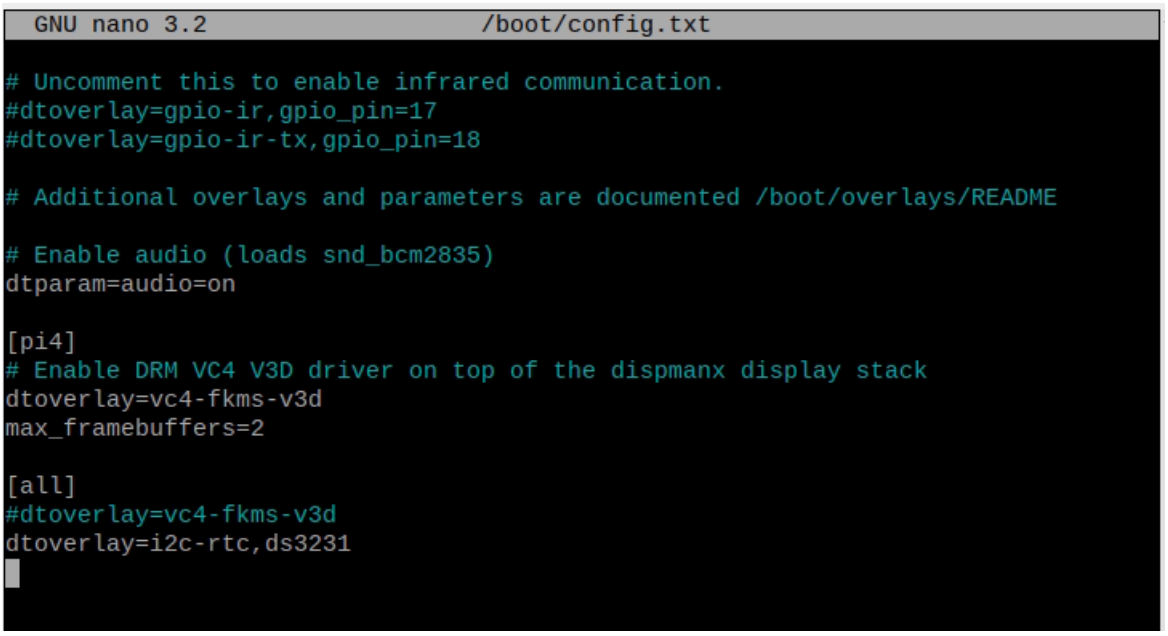

Figura 4.25: Configuración del reloj DS3231.

De esta forma luego de reiniciar con “sudo reboot”, se debería apreciar que el puerto en el que está conectado el reloj fue reemplazado por las letras “UU”

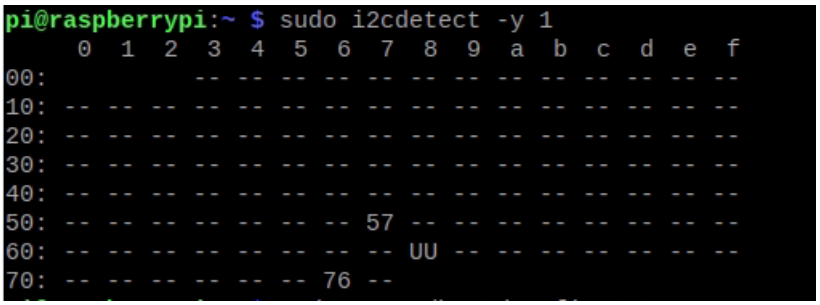

Figura 4.26: Detección del puerto del reloj.

Se continua deshabilitando los relojes que puedan interferir con el que se está instalando a través de los comandos:
“sudo apt-get -y remove fake-hwclock”
“sudo update-rc.d -f fake-hwclock remove”
“sudo systemctl disable fake-hwclock”

Después se habilita el reloj que se usará con el comando “sudo nano /lib/udev/hwclock-set” donde se comentarán las siguientes instrucciones:
“if [-e /run/systemd/system] ; then”
“ exit 0”
“fi”
“/sbin/hwclock –rtc=$dev –systz –badyear”
“/sbin/hwclock –rtc=$dev –systz”

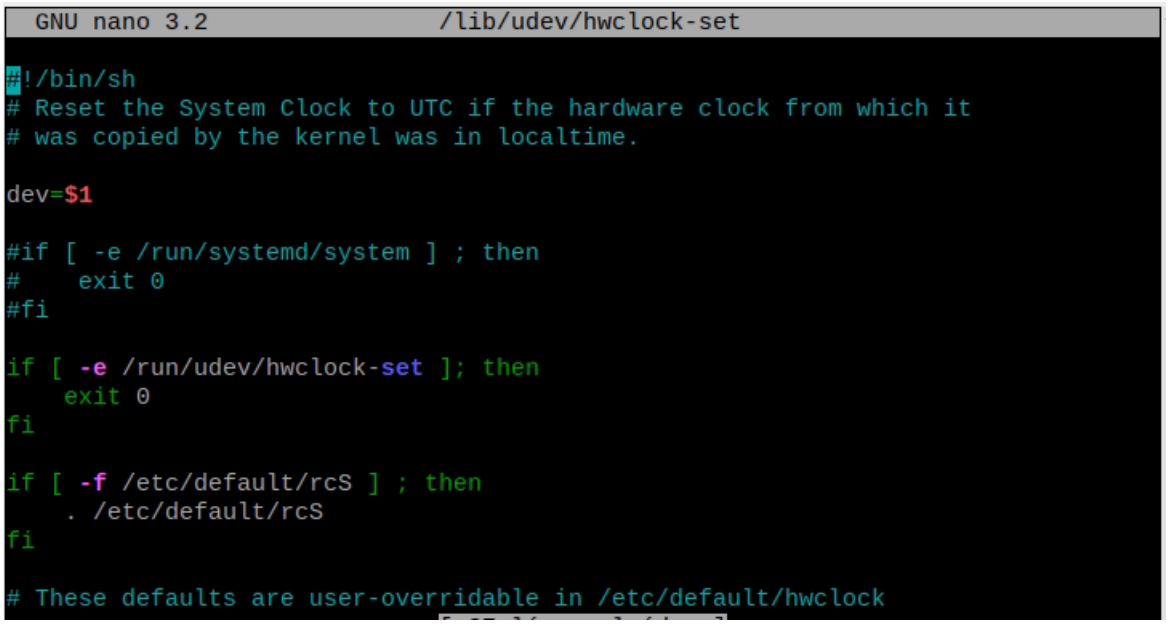

Figura 4.27: Comentarios al habilitar el reloj.

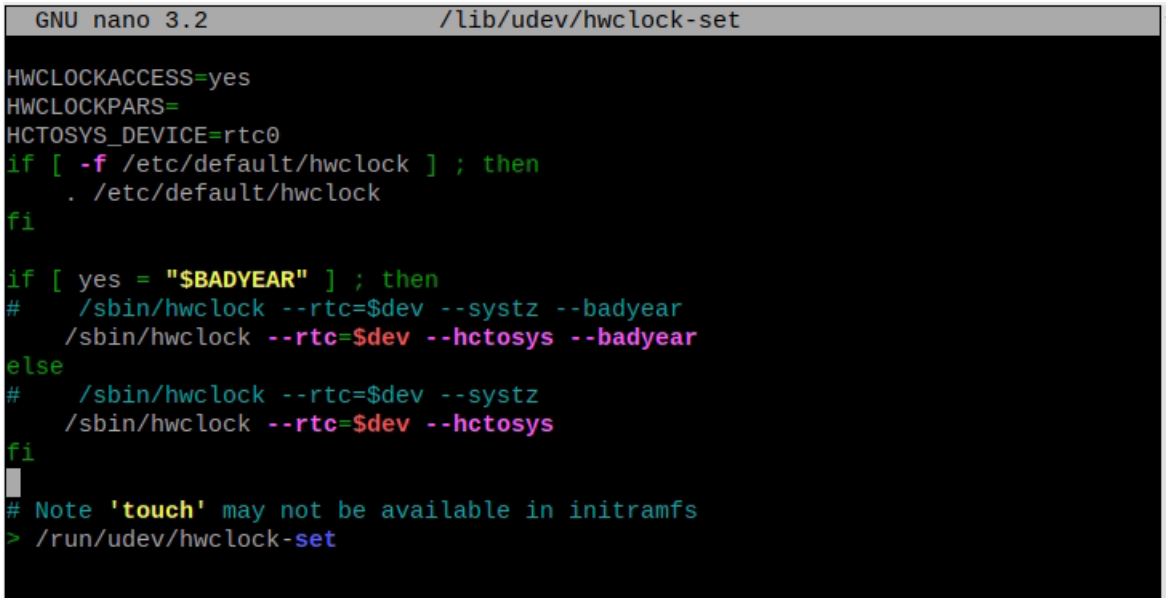

Figura 4.28: Comentarios al habilitar el reloj.

Finalmente, se sincroniza el reloj con el tiempo real. Para esto se utiliza el comando "sudo hwclock -r" con el que se verá que hora tiene actualmente el reloj. Luego se conecta la Raspberry a internet y con "sudo hwclock -w' se modifica la hora a la correcta. Con esto, pese a que no se tenga conexión a internet, el reloj mantendrá la red sincronizada y cada vez que se conecte a internet, se sincronizará automáticamente acercándose al tiempo correcto si por alguna razón se ha desviado un poco.

Capítulo 5

Microvibraciones Ambientales

Debido al corto tiempo de monitoreo, en el que, como se especificó anteriormente, hubo varios problemas de suministro eléctrico a los sensores debido al cargador de batería; se ha podido registrar una cantidad limitada de registros de microvibraciones ambientales con los geófonos presentes en cada equipo. De esta forma se analiza las obtenidas:

Tabla 5.1: Micro vibraciones Registradas

Equipos	Fecha de Adquisición	Periodo de Adquisición [UTC]
Todos	20-10-2021	16:40-23:59
Todos	21-10-2021	0:00-23:59
Todos	22-10-2021	0:00-23:59
Todos	23-10-2021	0:00-5:00
R4E16, R81A0, R5642, RACCE	10-11-2021	14:30-2359
R4E16, R81A0, R5642, RACCE	11-11-2021	0:00-23:59
R4E16, R81A0, R5642, RACCE	12-11-2021	0:00-23:59
R4E16, R81A0, R5642, RACCE	13-11-2021	0:00-17:30
Todos	19-11-2021	13:30-23:59

Finalmente, desde que se logra corregir los problemas de suministros de energía arreglando los problemas del cargador, desde el 29 de noviembre de 2021 se pueden registrar datos continuamente.

5.1. Registros

Para obtener los datos de los equipos de los registros especificados en la tabla 5.1 se utiliza el programa FileZilla. Una vez descargados desde la Raspberry al computador personal, se programa en MATLAB un código para leer los archivos miniseed y separarlos en ventanas de 30 minutos en archivos ".txt", los que serán analizadas en la sección 5.3.2.
Este y códigos directamente relacionados a su funcionamiento de encuentran en el anexo A.

Para obtener los registros de los sensores de forma automática en la Raspberry Pi a partir

del 29 de noviembre, se utiliza un software de código abierto llamado Obspy[13] que permite ser utilizado en Python para leer los archivos "mseed" que arroja cada instrumento.
Este código permite pasar todos los datos a un archivo ".txt" para luego programar, a partir de Octave, un código capaz de procesar estos registros separados por tiempo y realizar un análisis FDD completo cada una hora.
Además, permite registrar instantáneamente la humedad y temperatura del lugar junto a las 3 primeras frecuencias naturales de la estructura para ser comparadas más adelante.
Este y códigos directamente relacionados a su funcionamiento se encuentran en el anexo A.

Tal como se establece en la tabla 5.1 se obtienen registros de micro vibraciones ambientales recolectados para los tiempos y equipos especificados. Los resultados de una ventana del día 20 de octubre de 2021 entre 17:00 y 17:30 se muestran en la figura 5.1:

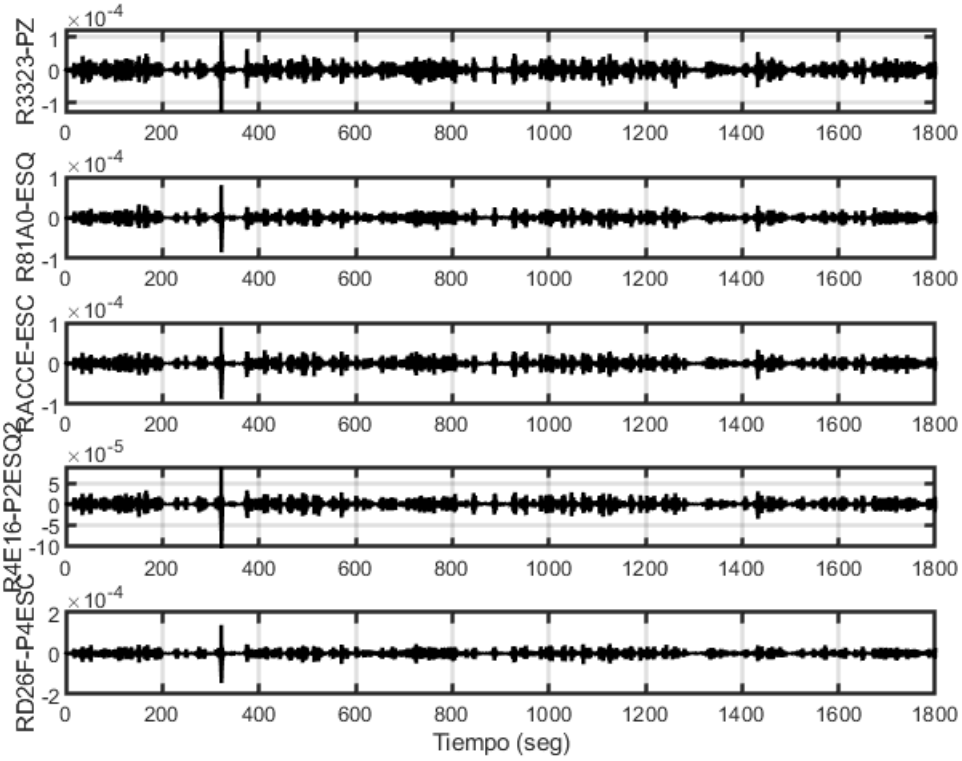

Figura 5.1: Registro de sensores el día 20-10-2021 de 17:00 a 17:30

A modo de ejemplo de lo registrado, los registros del día 21-10-21 se pueden encontrar en el anexo B.

5.2. Condiciones Ambientales

Para medir como varían las propiedades de la estructura en el tiempo, se decide medir de igual forma las condiciones ambientales a las que queda expuesta la estructura el día de los registros, de esta forma, antes de instalar el sensor de temperatura y humedad, se utilizan los datos de la Dirección Meteorológica de Chile, en la estación 330007[14].

La estación se encuentra en el interior del aeródromo rodelillo de Valparaíso a una distancia de 13,92 km de la torre, tal como se puede apreciar en la figura 5.2:

Figura 5.2: Ubicación de la estación 30007 y distancia de la Torre Peñuelas

En la tabla se muestra como ejemplo las temperaturas y la humedad relativa medida en la estación meteorológica para el día 21-10-21 a distintas horas (UTC).

Tabla 5.2: Condiciones Ambientales 21 de Octubre de 2021

Día	21-10-2021	
Hora [UTC]	T° [°C]	Humedad [%]
0:00-0:30	11,6	83
0:30-1:00	11,6	83
1:00-1:30	11,8	80
1:30-2:00	11,8	80
2:00-2:30	12	80
2:30-3:00	12	80
3:00-3:30	11,6	80
3:30-4:00	11,6	80
4:00-4:30	10,4	83
4:30-5:00	10,4	83
5:00-5:30	9	91
5:30-6:00	9	91
6:00-6:30	8,5	91
6:30-7:00	8,5	91
7:00-7:30	8,6	91
7:30-8:00	8,6	91
8:00-8:30	9,2	87
8:30-9:00	9,2	87
9:00-9:30	9,5	85
9:30-10:00	9,5	85
10:00-10:30	9,8	84
10:30-11:00	9,8	84
11:00-11:30	10,6	81
11:30-12:00	10,6	81

Tabla 5.3: Condiciones Ambientales 21 de Octubre de 2021

Día	21-10-2021	
Hora [UTC]	T°[°C]	Humedad [%]
12:00-12:30	12	78
12:30-13:00	12	78
13:00-13:30	14,4	67
13:30-14:00	14,4	67
14:00-14:30	16,2	61
14:30-15:00	16,2	61
15:00-15:30	16	63
15:30-16:00	16	63
16:00-16:30	16,8	61
16:30-17:00	16,8	61
17:00-17:30	17,7	59
17:30-18:00	17,7	59
18:00-18:30	18,5	57
18:30-19:00	18,5	57
19:00-19:30	18,8	56
19:30-20:00	18,8	56
20:00-20:30	18,5	54
20:30-21:00	18,5	54
21:00-21:30	16,9	57
21:30-22:00	16,9	57
22:00-22:30	15,3	67
22:30-23:00	15,3	67
23:00-23:30	12,9	78
23:30-0:00	12,9	78

Este proceso se realizó de manera manual extrayendo los datos que se obtienen de la pagina web de la Dirección Meteorológica de Chile, en su apartado de servicios climáticos. Luego, en la sección Productos>Diarios>Estaciones Automáticas>Datos Horarios Estación EMA. Se introduce el número de la estación y el día que se quiere obtener los datos.
Una vez obtenidos se pasan a formato excel de manera manual tal como se apreción entre la tabla 5.2.

A partir del 29 de Noviembre, como se ha comentado anteriormente, se instala un sensor BME280 con el que se programa a través de Octave el registro de la información para obtener la medición más precisa de las condiciones ambientales al interior de la estructura en el momento exacto en el que se realiza el análisis de la estructura.

5.3. Análisis de los Registros de Microvibraciones

5.3.1. Introducción Teórica

En esta sección se especifican los métodos que se utilizarán para obtener las propiedades de la torre a partir de los datos obtenidos.

5.3.1.1. Descomposición en el Dominio de la Frecuencia (FDD)

La descomposición en el dominio de la frecuencia es un método de análisis modal en el que los parámetros dinámicos de la estructura son obtenidos a partir de señales de respuesta, como el espectro de densidad de potencia y espectros de potencia cruzado obtenido por sensores en distintos puntos de una estructura de varios grados de libertad.

El método consiste en la manipulación de la respuesta tipo de una estructura de N grados de libertad, que se puede expresar como:

$$x(t) = [\phi] * y(t) \tag{5.1}$$

Donde:
x(t)=Vector de desplazamientos reales
ϕ=Matriz de forma modal
y(t)=Vector de desplazamientos modales

Si se le aplica la Transformada de Fourier a ambos lados de la ecuación, se obtiene la frecuencia a ambos lados, quedando de igual manera la matriz de forma modal, lo que se puede expresar de la siguiente forma:

$$X(j * w) = [\phi] * Y(j * w) \tag{5.2}$$

Si se le calcula la transpuesta queda de la siguiente manera (con H=hermitiano):

$$X(j * w)^H = Y(j * w)^H * [\phi]^T \tag{5.3}$$

De esta forma, al multiplicar por la izquierda la ecuación 6.2 por la ecuación 6.3, se obtiene:

$$X(j * w) * X(j * w)^H = [\phi] * Y(j * w) * Y(j * w)^H * [\phi]^T \tag{5.4}$$

Donde la parte izquierda de la ecuación corresponde a la matriz de espectro de respuesta de salida y la parte derecha corresponde a las formas modales por la matriz de respuestas modales, multiplicado por las formas modales traspuestas.

De esta forma, si los modos son independientes, al obtener el valor esperado de la ecuación 6.4, la parte derecha se diagonaliza:

$$E[X(j*w)*X(j*w)^H] = [\phi]*E[Y(j*w)*Y(j*w)^H]*[\phi]^T \tag{5.5}$$

$$[S_{x_i x_j}(j*w)] = [\phi]*S_{y_i y_j}(j*w)*[\phi]^T \tag{5.6}$$

Donde la parte derecha, al suponer una acción aleatoria del tipo de ruido blanco, se calcula cada espectro de potencia como $S_{yi}(w) = |H_i^m(j*w)|^2 * S_{0-i}^m$.
Así, reemplazando, obtenemos la matiz de respuesta como:

$$\left[S_{x_i x_j}(j*w)\right] = \left[\phi\right] * \begin{bmatrix} | & \dots & 0 \\ \dots & |H_i^m(j*w)|^2 * S_{0-i}^m & \dots \\ 0 & \dots & | \end{bmatrix} * \left[\phi\right]^T \tag{5.7}$$

Por lo tanto los pasos del método son:

1. Construir la matriz de potencia de las respuestas medidas.
2. Realizar una descomposición de valores singulares o equivalentemente determinar el problema de valores y vectores propios.
3. Graficar los valores singulares en función de la frecuencia (figura 5.3).
4. A partir de los peaks del grafico de valores singulares, determinar las frecuencias naturales amortiguadas. En caso que existan modos de frecuencias cercanas, se podrá visualizar dos modos de frecuencias cercanas en los peaks de dos o varios valores singulares.
5. Las formas modales son obtenidas de la parte real de los vectores propios asociados a cada peak identificado de cada valor singular.
6. La parte imaginaria de los vectores propios asociados a las frecuencias fundamentales se asocian a la tasa de amortiguamiento del modo.

5.3.2. Análisis por FDD Manual

El método FDD explicado en 5.3.1.1 fue codificado en el programa MATLAB
En un principio, los datos registrados en el capítulo 5 fueron separados cada media hora a través del código del anexo A.1, para luego ser analizados con el código descrito en el párrafo anterior, de donde se obtienen los siguientes resultados:

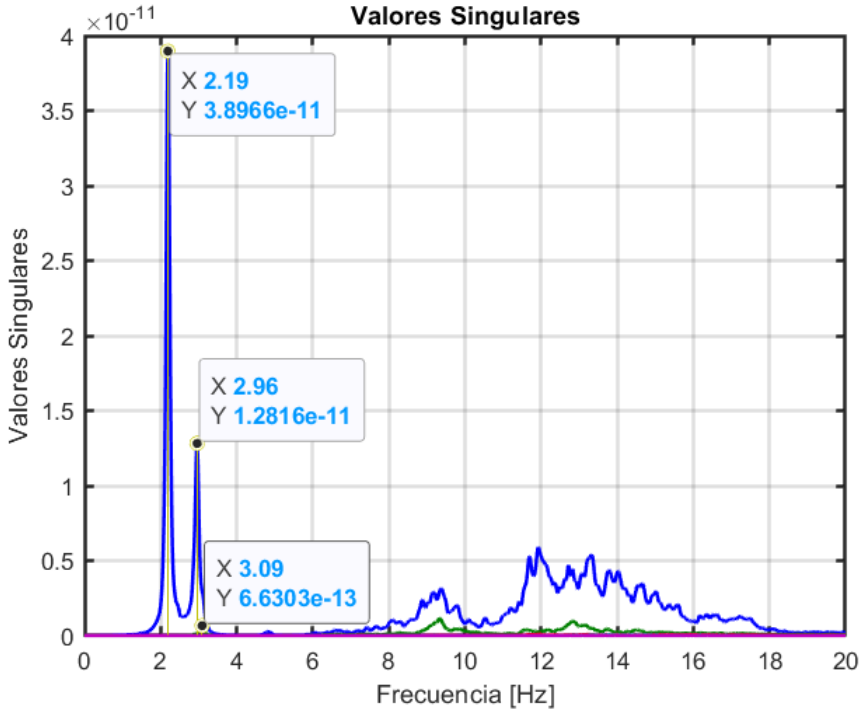

Figura 5.3: Análisis FDD de las estacion entre las 0:30 y 1:00 UTC el día 21-10-2021

De esta figura podemos obtener las 3 primeras frecuencias naturales de la estructura, que se pueden obtener de manera más sencilla al aplicar la función logaritmica, tal como se muestra a continuación:

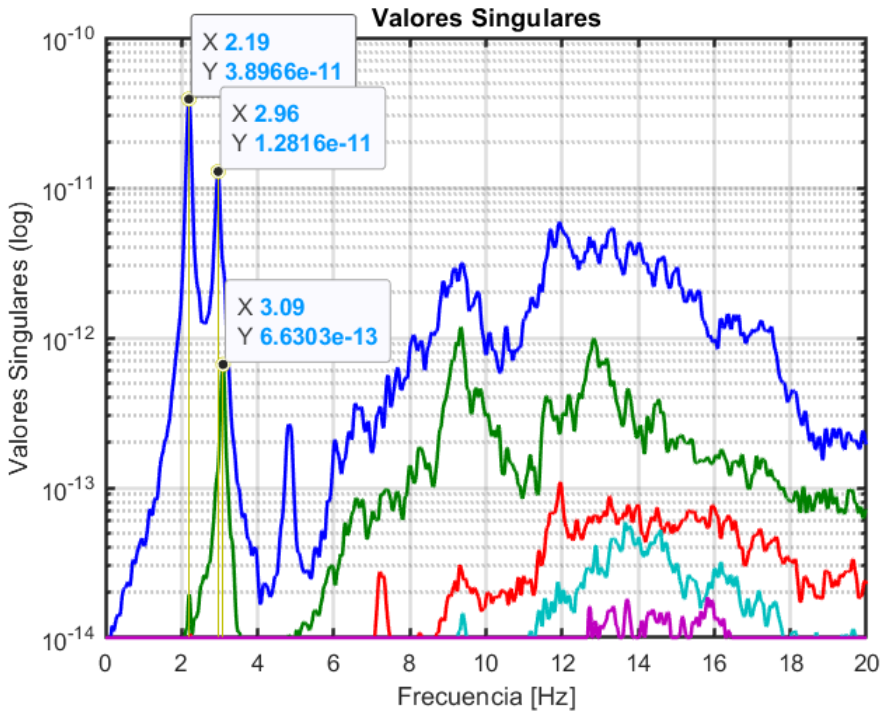

Figura 5.4: Análisis FDD de las estacion entre las 0:30 y 1:00 UTC el día 21-10-2021

A modo de ejemplo, los análisis de los registros asociados al día 21-10-21 se pueden encontrar en los anexos en las figuras pertenecientes al anexo C.

Tal como se apreció en las figuras 5.3 y 5.4, las 3 primeras frecuencias naturalesde la torre el día 21 de octubre de 2021, entre las horas 0:30 y 1:00 UTC, fueron de 2.19, 2.96 y 3.09.

Asi, juntando esta información con los datos obtenidos de la estación meteorológica, y clasificando los resultados por hora para el día 21-10-21, se obtiene las siguiente tabla resumen:

Tabla 5.4: Resultados Análisis FDD Manual el día 21 de octubre de 2021

Día	21-10-2021				
Hora [UTC]	$f_1[Hz]$	$f_2[Hz]$	$f_3[Hz]$	T° [°C]	Humedad Relativa[%]
0:30-1:00	2,19	2,96	3,08	11,6	83
1:00-1:30	2,2	2,97	3,08	11,8	80
1:30-2:00	2,19	2,98	3,08	11,8	80
2:00-2:30	2,2	2,98	3,1	12	80
2:30-3:00	2,2	2,98	3,08	12	80
3:00-3:30	2,21	2,98	3,09	11,6	80
4:00-4:30	2,2	2,98	3,11	10,4	83
4:30-5:00	2,19	2,99	3,09	10,4	83
5:00-5:30	2,2	2,99	3,1	9	91
5:30-6:00	2,21	2,99	3,1	9	91
6:00-6:30	2,21	2,99	3,1	8,5	91
11:00-11:30	2,22	2,99	3,11	10,6	81
11:30-12:00	2,21	2,99	3,1	10,6	81
12:00-12:30	2,21	2,98	3,11	12	78
12:30-13:00	2,2	2,98	3,1	12	78
13:00-13:30	2,2	2,98	3,09	14,4	67
13:30-14:00	2,2	2,97	3,1	14,4	67
14:00-14:30	2,18	2,97	3,07	16,2	61
14:30-15:00	2,17	2,96	3,07	16,2	61
15:00-15:30	2,17	2,94	3,07	16	63
15:30-16:00	2,16	2,93	3,05	16	63
16:00-16:30	2,15	2,91	3,03	16,8	61
16:30-17:00	2,14	2,92	3,01	16,8	61
17:00-17:30	2,12	2,9	3	17,7	59
17:30-18:00	2,11	2,89	2,99	17,7	59
18:00-18:30	2,1	2,89	3,01	18,5	57
18:30-19:00	2,13	2,87	3,02	18,5	57
19:00-19:30	2,12	2,9	2,99	18,8	56
20:30-21:00	2,11	2,86	3	18,5	54
21:00-21:30	2,12	2,9	3,03	16,9	57
21:30-22:00	2,12	2,91	3	16,9	57
22:00-22:30	2,13	2,9	3	15,3	67
22:30-23:00	2,14	2,89	3	15,3	67
23:00-23:30	2,15	2,91	3,03	12,9	78
Promedio	2,169	2,945	3,059		
Desviación	0,038	0,042	0,043		

Así, realizando una comparación entre la humedad, la temperatura y como varían las 3 primeras frecuencias de la estructura, se pueden apreciar los datos de la tabla 5.4 en la figura 5.5:

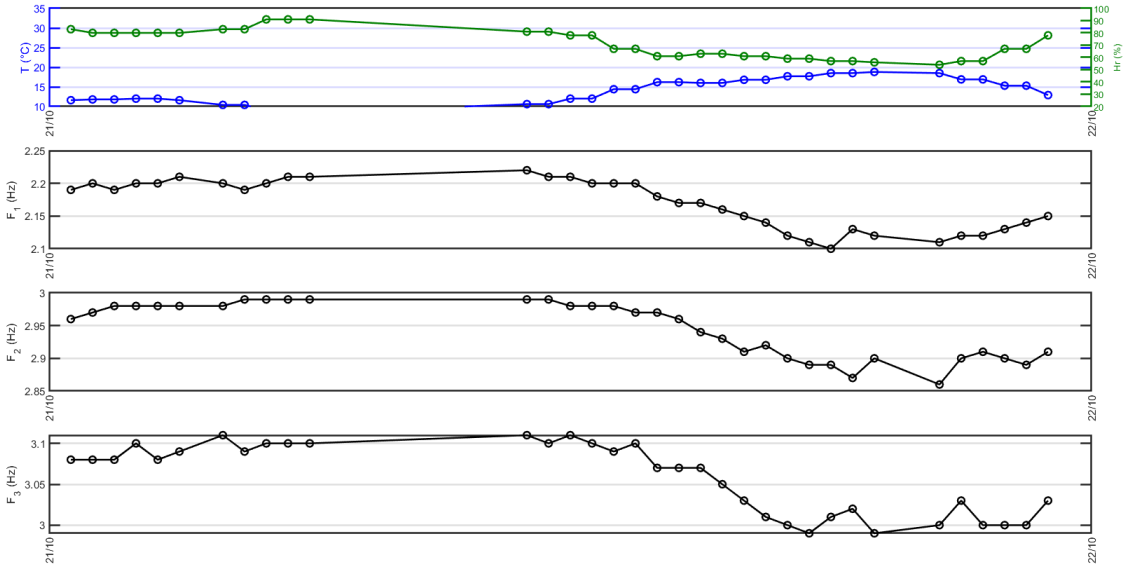

Figura 5.5: Comparación principal de frecuencias fundamentales vs condiciones ambientales día 21-10-2021

5.3.3. Análisis por FDD Automático

Tal como se comentó en la sección 5.1, a partir del 29 de noviembre, que fue el día en el que se solucionaron los problemas eléctricos que afectaban a las mediciones continuas de los sensores, se instaló un programa en Octave que permite realizar un análisis FDD en la misma Raspberry Pi y además registrar los datos de temperatura y humedad obtenidos desde el sensor BME280.

Con esta rutina (Anexo A.11), se decide realizar el análisis cada una hora. Para esto se utiliza nuevamente el crontab de la Raspberry para que el procesamiento se lleve a cabo cada una una hora. El procesamiento consiste en emplear ObsPy para extraer registros de 30 minutos (Anexo A.4) . Luego con la rutina programada en Octave del método FDD se determinan las primeras tres frecuencias fundamentales de la estructura. Las frecuencias y los datos de temperatura y humedad relativa son almacenados en un archivo, para ser posteriormente procesados con el el objetivo de analizar la variación de las frecuencias naturales con los parámetros ambientales.

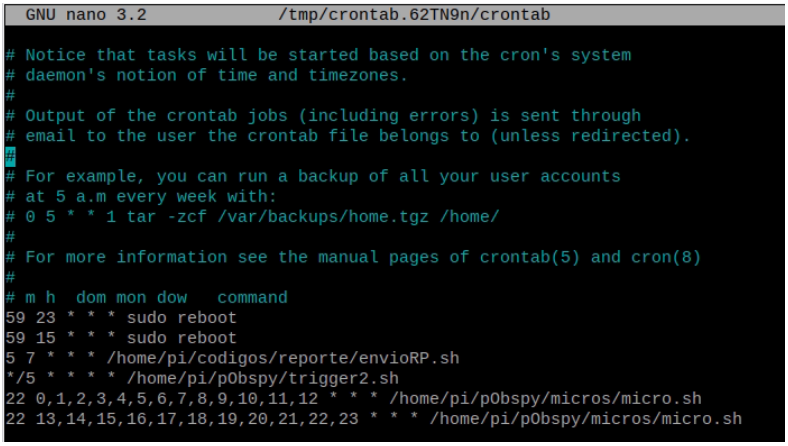

Figura 5.6: Programación en contrab de la realización del análisis FDD cada hora

Para verificar que el programa procesa de manera correcta el análisis FDD con la metodología automática, se compara de forma manual con lo registrado por el programa. Para esto se obtiene, al igual que en la sección anterior, los datos de un día completo a través de FileZilla separados cada media hora según los horarios a los que se programó para realizarse el análisis.

Así, se analiza el día 1 de diciembre de 2021 obteniendo los siguientes resultados:

Tabla 5.5: Comparación Análisis FDD Manual vs Automático.

Hora [UTC]	Peak 1	Peak 2	Peak 3	Peak 1 Código	Peak 2 Código	Peak 3 Código	Diferencia peak 1	Diferencia peak 2	Diferencia peak 3
1:22	2,10	2,86	2,98	2,10	2,86	2,98	0,00	0,00	0,00
2:22	2,12	2,87	3,00	2,12	2,88	3,00	0,00	-0,01	0,00
3:22	2,13	2,88	3,02	2,13	2,88	3,01	0,00	0,00	0,01
4:22	2,13	2,89	3,00	2,13	2,89	3,01	0,00	0,00	-0,01
5:22	2,13	2,90	3,02	2,14	2,90	3,02	-0,01	0,00	0,00
6:22	2,14	2,89	3,01	2,15	2,88	3,02	-0,01	0,01	-0,01
7:22	2,15	2,90	3,02	2,14	2,91	3,02	0,01	-0,01	0,00
8:22	2,15	2,89	2,99	2,14	2,89	3,00	0,01	0,00	-0,01
9:22	2,15	2,90	3,03	2,15	2,89	3,03	0,00	0,01	0,00
11:22	2,16	2,91	3,05	2,16	2,93	3,05	0,00	-0,02	0,00
12:22	2,15	2,90	3,03	2,16	2,91	3,05	-0,01	-0,01	-0,02
13:22	2,15	2,88	3,05	2,16	2,91	3,04	-0,01	-0,03	0,01
14:22	2,15	2,90	3,04	2,15	2,90	3,04	0,00	0,00	0,00
15:22	2,13	2,89	3,03	2,13	2,89	3,05	0,00	0,00	-0,02
16:22	2,12	2,86	2,99	2,12	2,86	3,00	0,00	0,00	-0,01
17:22	2,10	2,86	2,97	2,09	2,87	2,97	0,01	-0,01	0,00
18:22	2,07	2,82	2,97	2,08	2,80	2,97	-0,01	0,02	0,00
19:22	2,07	2,81	2,94	2,07	2,81	2,94	0,00	0,00	0,00
20:22	2,07	2,80	2,94	2,07	2,81	2,94	0,00	-0,01	0,00
21:22	2,07	2,82	2,94	2,07	2,82	2,94	0,00	0,00	0,00
22:22	2,07	2,81	2,94	2,07	2,81	2,94	0,00	0,00	0,00
23:22	2,08	2,81	2,94	2,07	2,8	2,92	0,01	0,01	0,02

Tabla 5.6: Estadísticas de comparación proceso manual vs computarizado

Diferencia promedio	Desviación estándar	Máxima diferencia	moda
0,0015	0,009	0,03	0

De donde se puede apreciar que la diferencia en la mayor cantidad de las veces 0, y en las que no, la diferencia es mínima, lo que puede deberse principalmente a la forma de capturar el peak en el gráfico de valores singulares.

Gráficamente se puede hacer la comparación de ambos resultados, cuando al colocarlos en un gráfico, ambos resultados debieran ser iguales y ajustarse a la recta en 45º de las figuras 5.7, 5.8 y 5.9 a continuación:

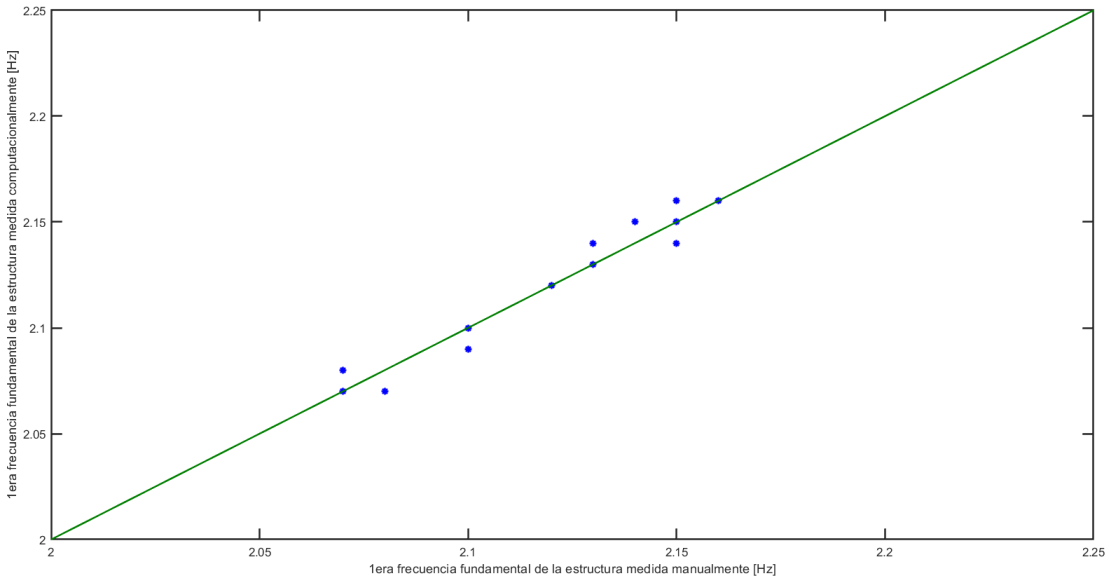

Figura 5.7: Comparación resultados 1era frecuencia manualmente vs computacionalmente

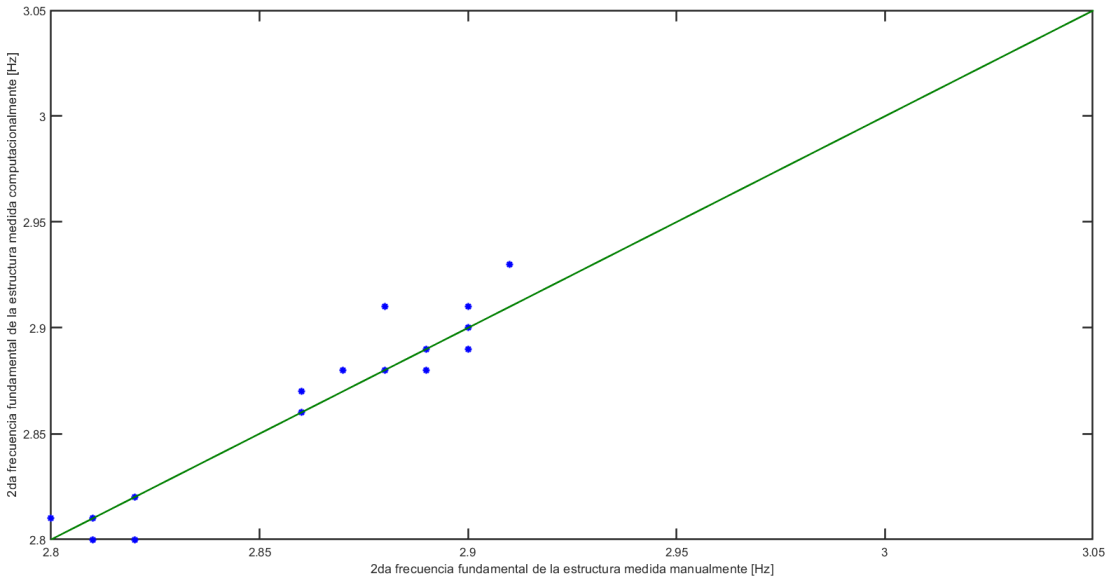

Figura 5.8: Comparación resultados 2da frecuencia manualmente vs computacionalmente

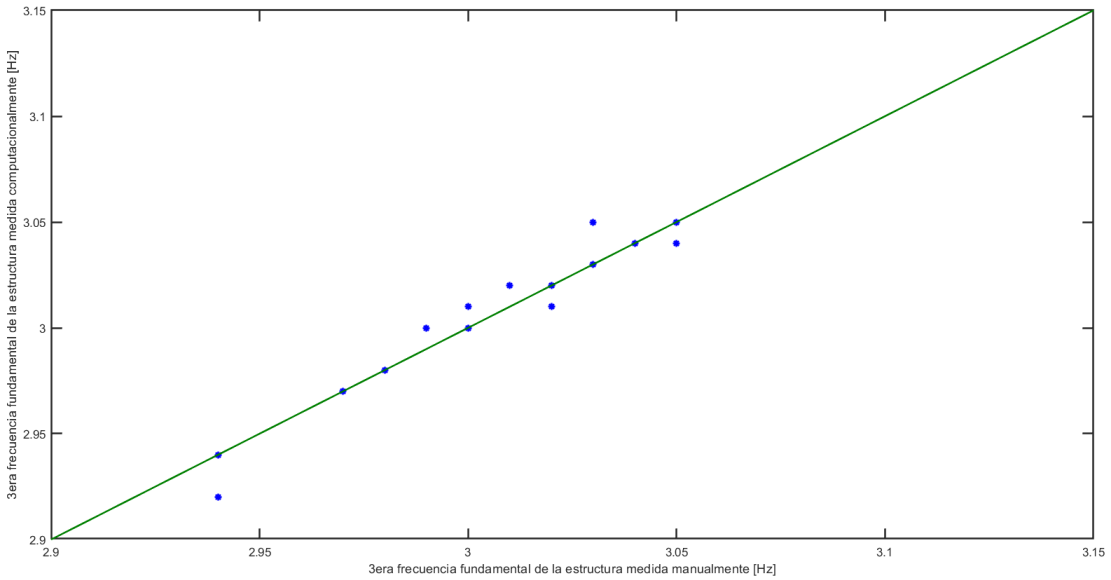

Figura 5.9: Comparación resultados 3era frecuencia manualmente vs computacionalmente

Tabla 5.7: Comparación entre Temperatura Obtenida a Través de Estación Metereológica y Sensor.

Hora [UTC]	Temperatura Estación [°C]	Humedad Estación [%]	Temperatura Sensor [°C]	Humedad Sensor [%]	Gradiente de Temperatura	Gradiente de Humedad
1:22	11,3	84,0	16,6	51,6	5,30	-32,4
2:22	10,1	90,0	16,6	51,1	6,50	-38,9
3:22	9,9	89,0	16,5	51,8	6,60	-37,2
4:22	9,7	92,0	16,7	51,2	7,00	-40,8
5:22	9,3	100,0	16,0	53,5	6,70	-46,5
6:22	11,1	93,0	16,2	52,1	5,10	-40,9
7:22	11,5	84,0	15,7	53,3	4,20	-30,7
8:22	11,2	84,0	15,6	53,1	4,40	-30,9
9:22	11,1	84,0	15,5	53,6	4,40	-30,4
11:22	11,3	83,0	14,9	54,0	3,60	-29,0
12:22	11,8	81,0	15,1	55,0	3,30	-26,0
13:22	12,4	77,0	15,4	54,1	3,00	-22,9
14:22	13,6	73,0	15,9	51,8	2,30	-21,2
15:22	16,7	60,0	16,8	50,3	0,10	-9,7
16:22	16,8	59,0	17,2	48,9	0,40	-10,1
17:22	19,5	51,0	17,7	49,1	-1,80	-1,9
18:22	19,0	51,0	18,1	48,0	-0,90	-3,0
19:22	18,8	52,0	18,5	47,5	-0,30	-4,5
20:22	18,5	55,0	19,2	45,9	0,70	-9,1
21:22	18,9	53,0	20,0	44,2	1,10	-8,8
22:22	17,3	58,0	23,4	37,5	6,10	-20,5
23:22	14,4	69,0	18,0	48,2	3,60	-20,8

De la tabla anterior se puede observar que la estación meteorológica obtiene la temperatura y humedad desde el exterior, mientras que el sensor instalado en la Raspberry Pi está sometido a condiciones al interior de la estructura. Sin embargo, al ser ambas consistentes con su ubicación, se puede observar una tendencia a la linealidad de ambos registros tal como se aprecia en las figuras 5.10 y 5.11:

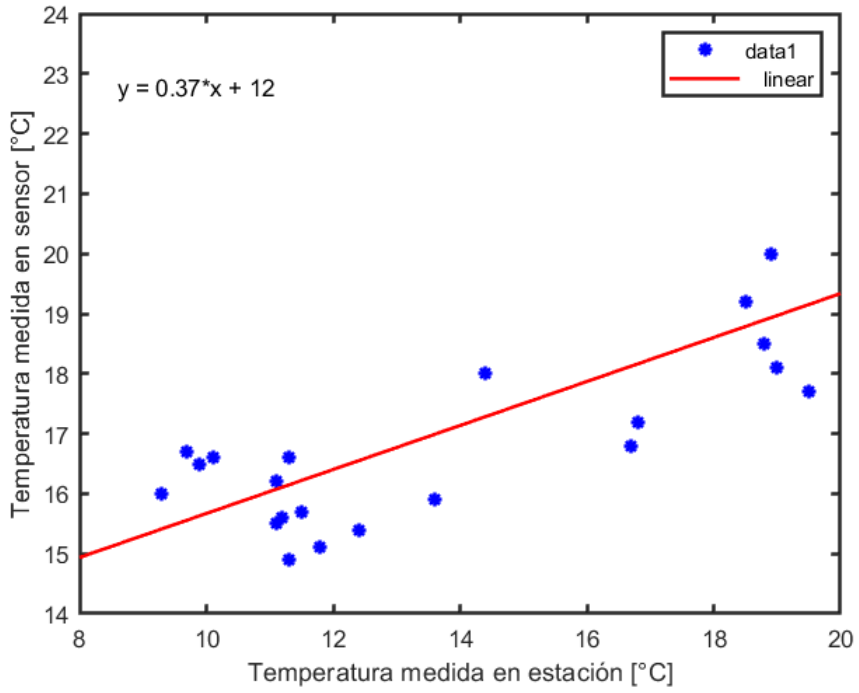

Figura 5.10: Comparación resultados 2da frecuencia manualmente vs computacionalmente

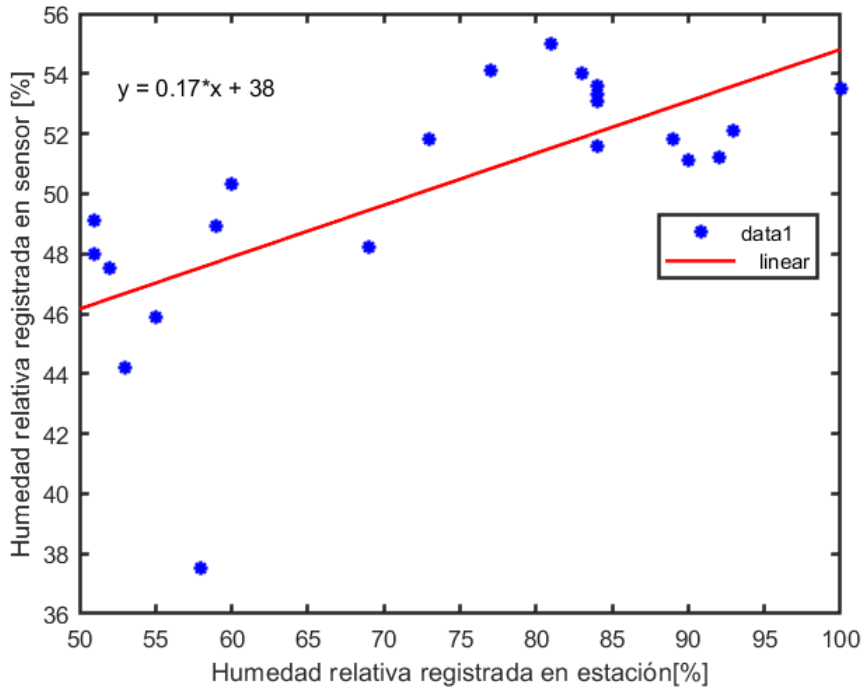

Figura 5.11: Comparación resultados 3era frecuencia manualmente vs computacionalmente

Así, se obtienen los resultados de las frecuencias principales en una comparación con la

variación de temperatura y humedad que se puede ver en la figura 5.12 comprendiendo las fechas entre el 30 de noviembre de 2021 y 29 de diciembre de 2021:

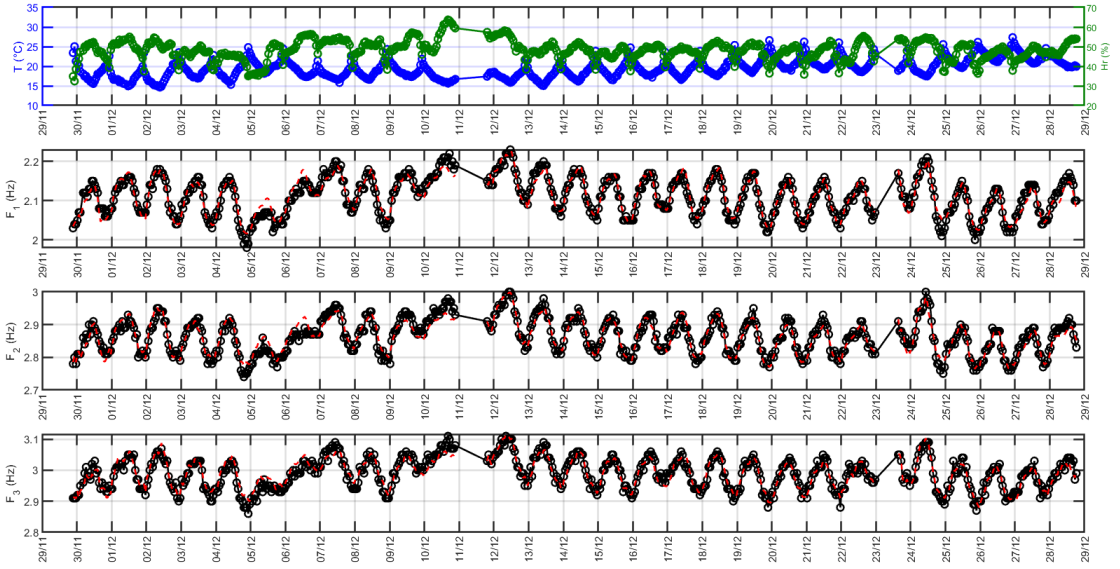

Figura 5.12: Comparación principal de peaks vs condiciones ambientales

De los datos de la figura 5.12 se desprende que la máxima variación ocurrida en un día fue el día 24 de diciembre de 2021, cuando las 3 frecuencias fundamentales alcanzaron su máxima variación correspondiente a un 9 %, 8 % y un 6 % respectivamente.

Esto último se explica bajo el concepto que el día 24 de diciembre fue el día en el que más varió su temperatura y humedad relativa de forma conjunta, mientras que en días en los que solo varió una variable de forma abrupta, si bien las frecuencias fundamentales variaron, no lo hicieron de forma tan marcada como el 24 de diciembre donde tanto la humedad relativa como la temperatura variaron un 32 % entre su máxima y su mínima.

5.3.4. Análisis de Resultados

Debido a la diferencia en las condiciones ambientales se decide trabajar y proceder en el análisis de resultados con las obtenidas por el método automático ya que este posee una mayor cantidad de datos y se decide no mezclar los datos tomados en el exterior con los obtenidos en el interior de la estructura debido a las diferencias en las condiciones ambientales determinadas en la estación meteorológica cercana y las medidas al interior de la estructura.

Se trabajaron 2 tipos de análisis, el primero un ajuste polinómico donde se estudian la tendencia del comportamiento a través de ecuaciones polinómicas que recreen su comportamiento y la segunda se trabaja con el "modelo espacio estado", un modelo matemático dinámico que en el que a partir de datos de entrada y salida, se calcula a través de recursiones y tomando e cuenta los datos pasados y próximos, la función que mejor represente el caso de forma estimada.

5.3.4.1. Método de Ajustes Polinómicos

En primer lugar, entendiendo que las frecuencias naturales de la estructura no son tan sensitivas a los peaks de temperatura y humedad relativa, se utiliza un filtro para evitar temperatura y/o humedades relativas que presenten condiciones muy dispares entre un momento y el siguiente debido a errores de medición, en el que para la temperatura del momento actual pasa a depender también con un ponderador de las anteriores, donde las 2 horas anteriores ponderan 1/4 y la medida por el sensor pondera 1/2.
Este filtro se utiliza a través de la función "filtfilt" que produce este comportamiento en el que el dato medido pasa a depender de los anteriores.

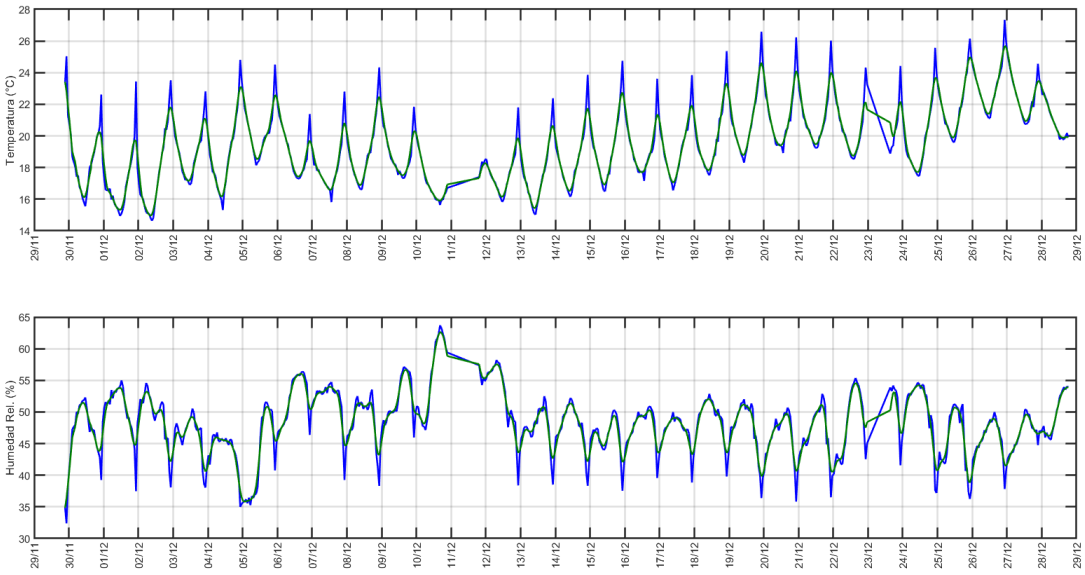

Figura 5.13: Condiciones ambientales filtradas.

En la figura 5.13 se puede apreciar las condiciones filtradas en verde con respecto a las medidas en color azul.

De esta forma, en la figura 5.14 se comienza comparando los resultados de humedad relativa y temperatura contra las frecuencias obtenidas de la estructura y se evalúa si existe una tendencia para cada una de las variables en forma independiente.

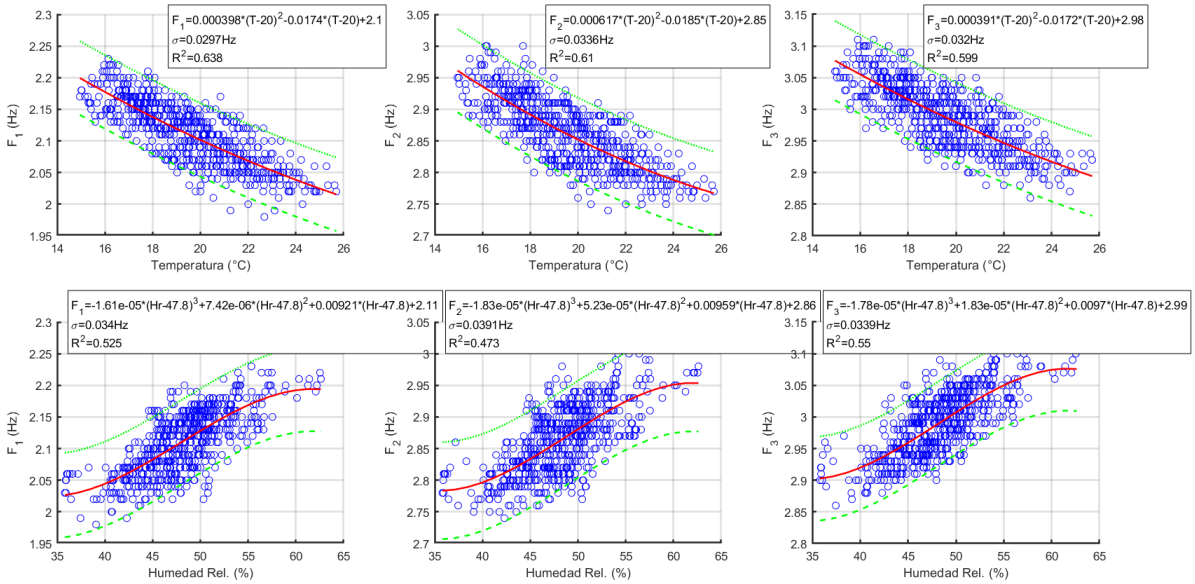

Figura 5.14: Gráficos de tendencia de las frecuencias en función de la temperatura y la humedad relativa del aire.

Es importante notar que las funciones fueron centradas a 20 °C y 47,8 % de humedad tal como demuestra cada una, para que de esta forma la constante cobre sentido físico al ser la frecuencia estimada a esas condiciones.

Con esto también se puede observar que las demás constantes y la desviación estándar son parecidas para las primeras tres frecuencias naturales de la estructura. De esta forma se puede concluir que las condiciones ambientales afectan en forma similar a las frecuencias fundamentales, indicando que la distribución de rigidez y masa permanece constante, pero existiría una variación de la rigidez (uniforme a lo largo de la estructura) por efectos ambientales.

Además, es importante destacar en la imagen que se acota esta tendencia por arriba y por abajo (líneas verdes de la figura 5.14) al multiplicar 1.96 veces la desviación estándar y así obtener el rango para un 95 % de confianza (suponiendo una distribución normal) en la obtención de las frecuencias.
En la figura 5.15 se muestra la distribución de los errores donde se ve su comportamiento gaussiano y con valores cercanos e iguales a cero en el intervalo de confianza, de donde se obtienen los ajustes individuales:

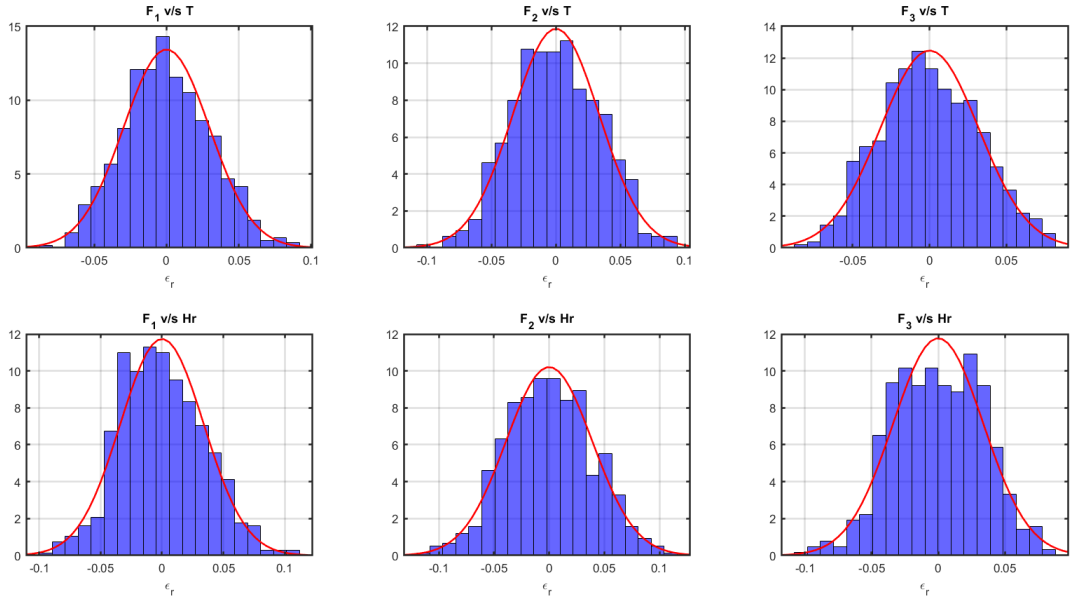

Figura 5.15: Análisis de dispersión en líneas de tendencia.

Una vez que se aprecia un carácter descendente en el caso de la frecuencia con respecto a la temperatura y uno con forma de s con respecto a la humedad relativa. Se puede observar que el factor de correlación sigue siendo bastante lejano a 1 que sería en el caso de que se pudiera predecir completamente su comportamiento a través de la ecuación.

La figura 5.15 indica que la temperatura es un mejor indicador independiente para estimar las frecuencias fundamentales de la estructura al poseer una menor dispersión. Por otra parte se puede observar que tanto la temperatura y la humedad relativa juegan un rol relavante. Por otro lado, se puede observar que la temperatura y humedad relativa tienen un cierto grado de correlación.

Se puede analizar de otra forma estos resultados como una comparación entre cada punto obtenido de forma experimental vs cada punto obtenido de forma estimada por las líneas de tendencia tal como se aprecia en la figura 5.16.

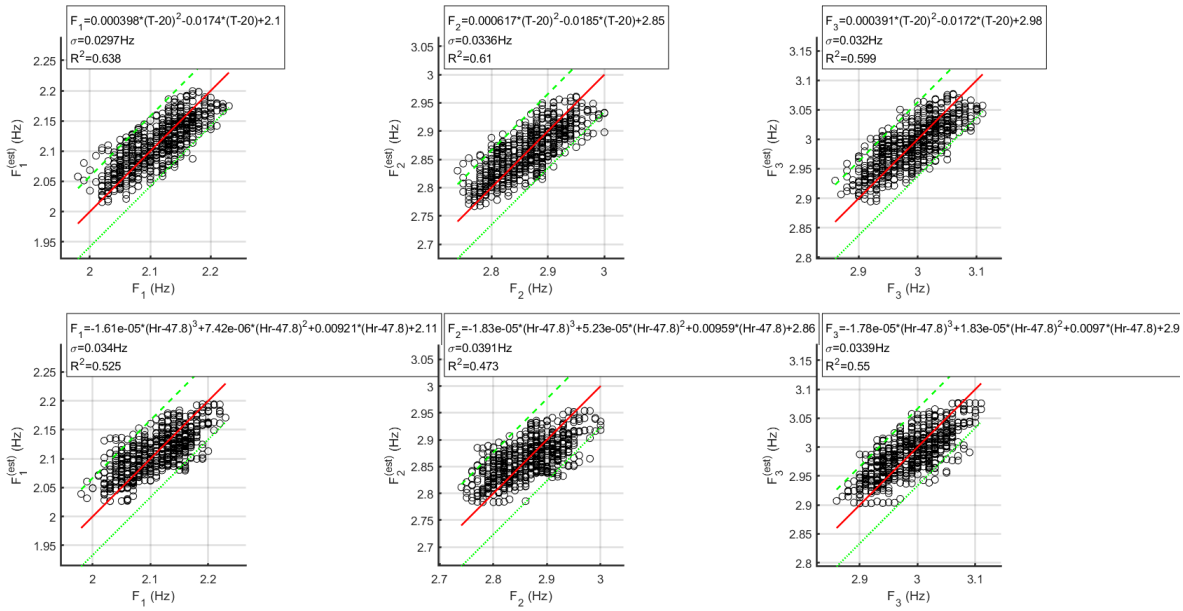

Figura 5.16: Frecuencias estimadas con respecto a las reales.

Notar que en el mejor de los casos, se debiera apreciar que los resultados debieran ser iguales y ser una recta en 45º en la figura 5.16 (x=y). Sin embargo, se puede apreciar fácilmente la dispersión de los datos.

Es por esto que se decide trabajar con un factor de corrección en el que se considera que las frecuencias obtenidas dependen tanto de la temperatura como de la humedad, por lo que no se puede considerar que una sola podrá modelar la curva que represente de mejor manera el comportamiento de la torre.

5.3.4.2. Método de Ajustes Polinómicos con 2 factores

Dado que se observa que las frecuencias naturales de la estructura dependen tanto de la temperatura y humedad relativa, se busca definir una función que dependa de ambos parametros. En este contexto, se puede modelar de alguna manera bajo el formato de

$$F = F(T) * C(H) \tag{5.8}$$

o como

$$F = F(T) + C(H) \tag{5.9}$$

donde $F(T)$ es la frecuencia estimada a partir de la línea de tendencia en función de la temperatura y $C(H)$ es un factor de corrección que dependerá de la humedad relativa del aire.

En este contexto, se considerará correcta la estimación debido a temperatura (que muestra una mayor correlación) y se aplica una función de corrección a la estimación por temperatura debido a la humedad relativa.

Para esto, se coloca todo en función de la humedad relativa como

$$F/F(T) = C(H) \tag{5.10}$$

o como

$$F - F(T) = C(H) \tag{5.11}$$

donde el factor se puede estimar fácilmente al dividir (o restar) cada punto obtenido de forma experimental por cada punto obtenido de forma estimada y luego colocar eso en función de la humedad relativa para obtener la función correctora como se puede ver gráficamente:

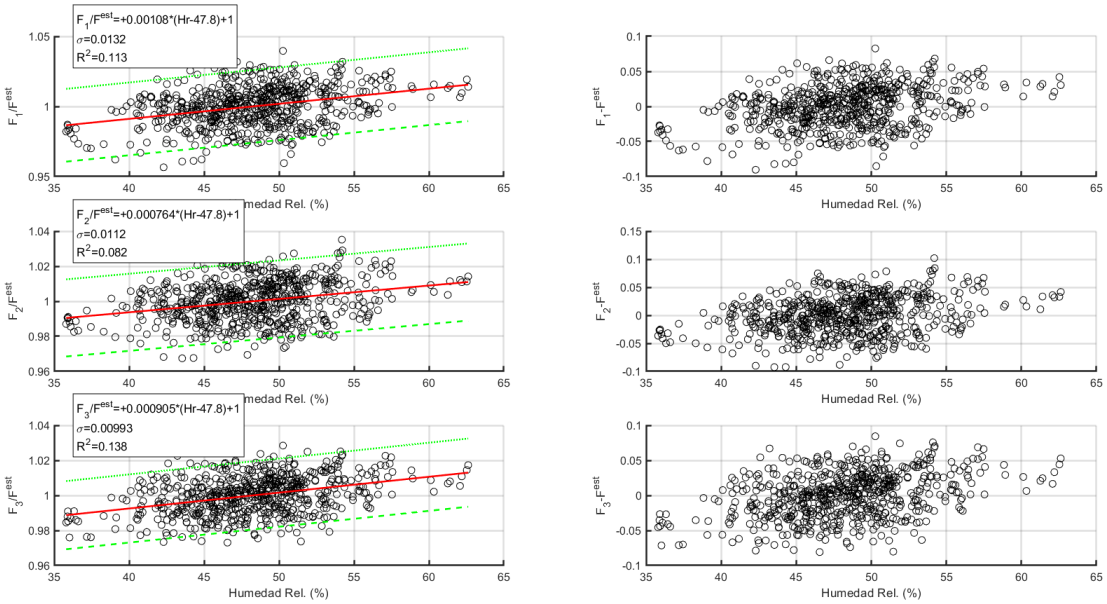

Figura 5.17: Obtención factor de corrección.

Así, se obtiene una función correctora que depende de la humedad y que puede ponderar a la función que estima la frecuencia a partir de líneas de tendencia que responden a la temperatura.
Se aplica esta función correctora a las estimaciones iniciales y se crea un nuevo modelo que estima las frecuencias, pero esta vez en función de la temperatura y la humedad juntas.
Gráficamente se puede ver el análisis de la siguiente forma:

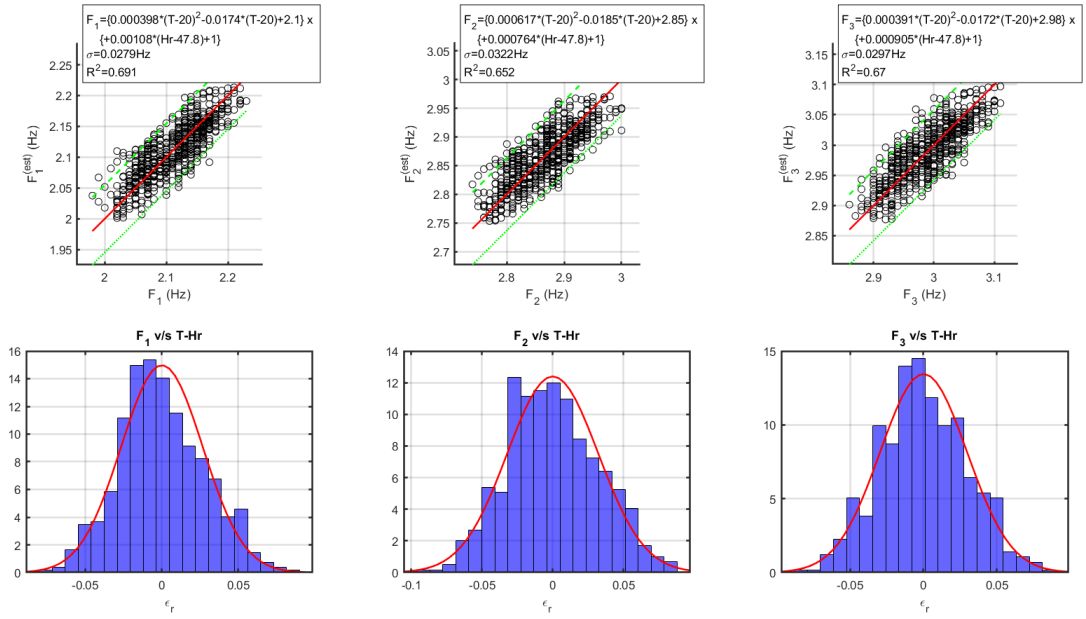

Figura 5.18: Análisis con factor de corrección por humedad.

Se observa que se logra disminuir la desviación estándar y aumentar el factor de correlación, sin embargo, no se ve que sea de gran manera con respecto a la estimación más sencilla que solo corresponde a la temperatura.

Ante esto se decide estudiar el factor de corrección en base a una humedad normalizada tomada como la resta entre la humedad relativa del aire y la humedad estimada. Esto es similar a la idea anterior y se puede ver de manera gráfica a continuación:

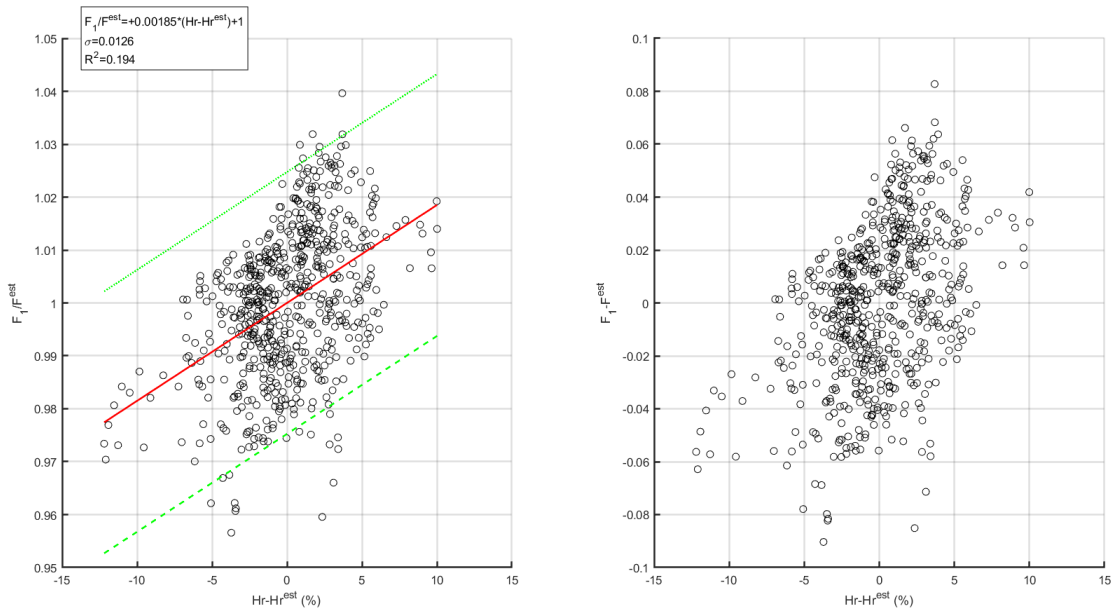

Figura 5.19: Obtención factor de corrección con humedad normalizada.

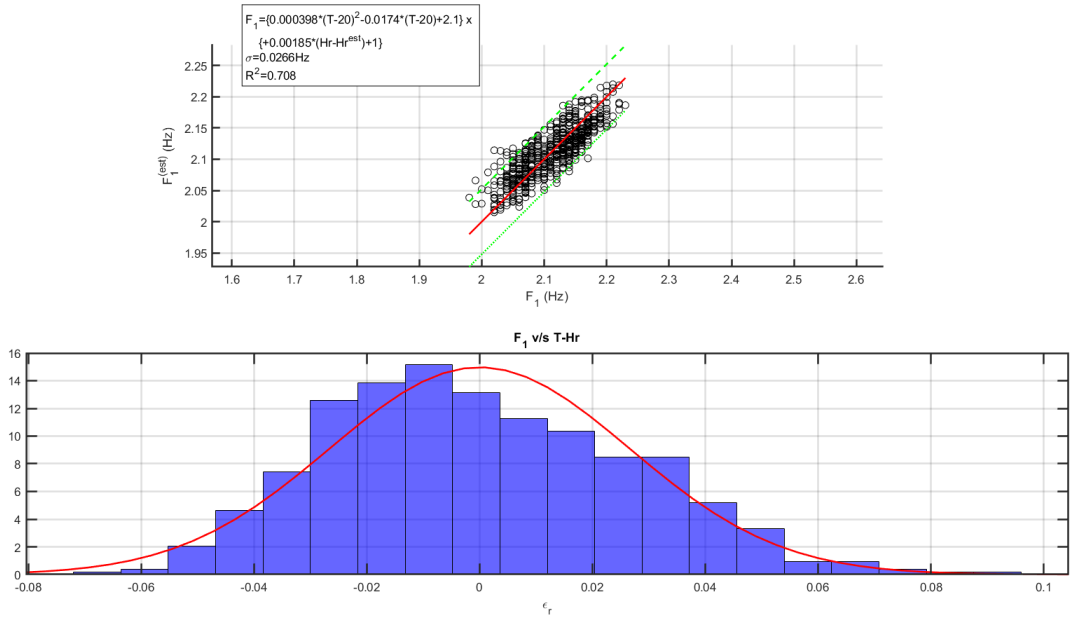

Figura 5.20: Análisis con factor de corrección por humedad normalizada.

5.3.4.3. Modelo Espacio Estado

El modelo espacio estado, como bien se introdujo anteriormente es un un modelo matemático dinámico que en el que a partir de datos de entrada y salida se calcula a través de recursiones y tomando e cuenta los datos pasados y próximos, la función que mejor represente el caso de forma estimada, que se puede trabajar fácilmente a través de programas como MATLAB con la función "n4sid"[2].

Con este procedimiento se puede modelar la respuesta de manera gráfica tal como se ha hecho hasta el momento:

[2] Para más información de cómo funciona el modelo y como trabajar con el: https://www.mathworks.com/help/ident/ref/n4sid.html

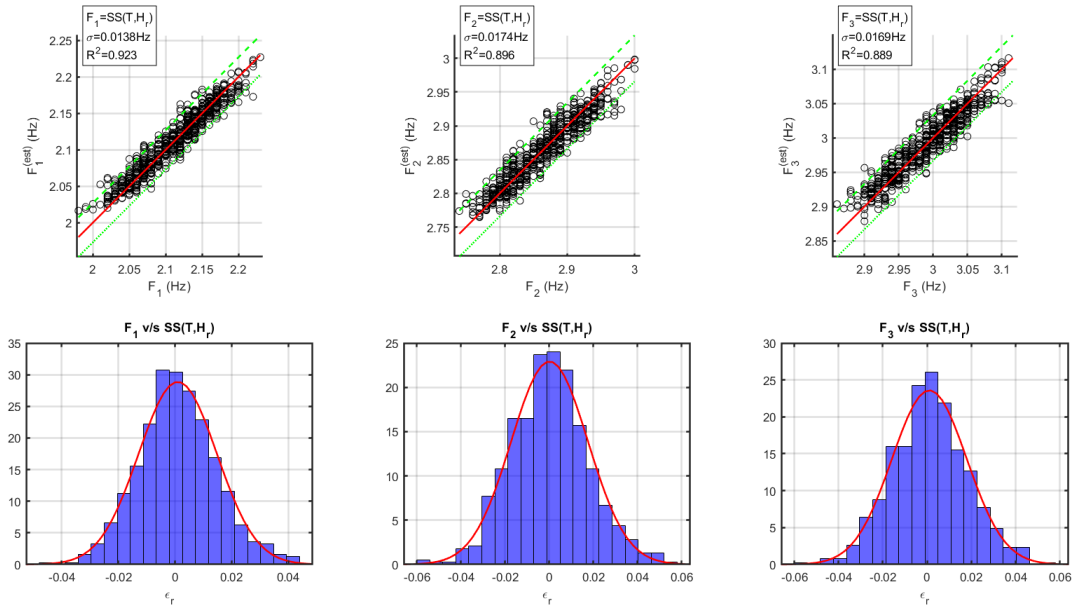

Figura 5.21: Análisis a través del modelo espacio estado.

La línea roja que se muestra en la figura 5.21 presenta la comparación de la estimación.

Con este modelo se aprecia un salto considerable en el factor de correlación y una gran disminución en la desviación estándar, con lo que se espera poder predecir los cambios de frecuencia de la torre a partir de las condiciones ambientales.

5.3.4.4. Resumen de los Análisis

Para visualizar todo lo anterior, se estudia el comportamiento de todos los modelos estudiados al graficar la predicción de cada uno con respecto a los datos recolectados en terreno ante las condiciones ambientales:

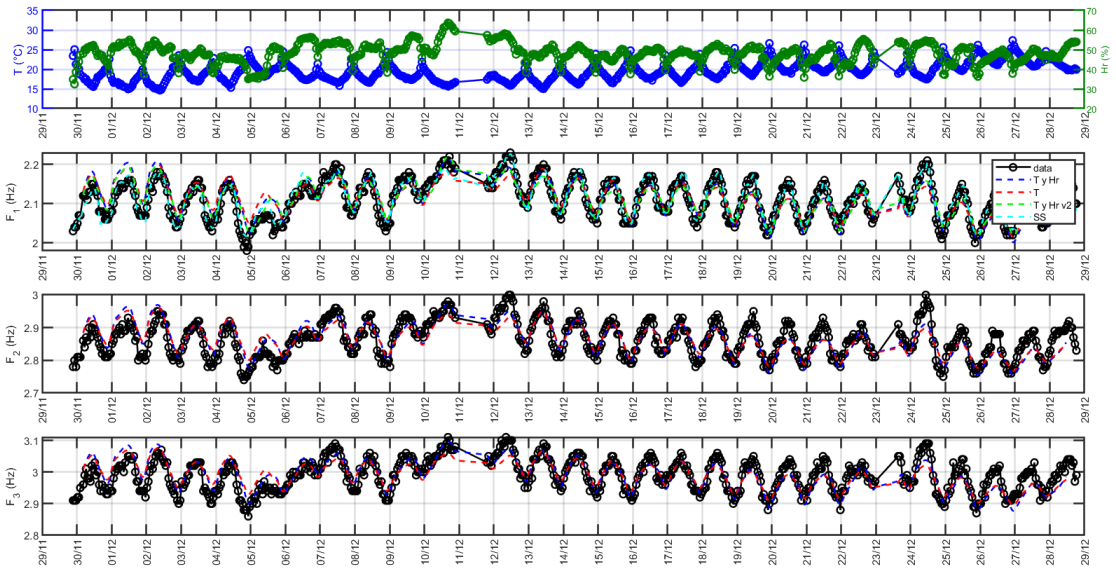

Figura 5.22: Análisis modelos de predicción de frecuencia.

Donde cada línea corresponde a un modelo:

Tabla 5.8: Modelos de predicción frecuencia vs condiciones ambientales.

Color Línea	Modelo
Roja	Frecuencia en función de la temperatura
Azul	Frecuencia en función de la temperatura corregida por humedad normalizada
Verde	Frecuencia en función de la temperatura corregida por humedad
Celeste	Modelo espacio estado
Negra	Mediciones experimentales

Finalmente, con modelo predictivo obtenido se puede realizar un análisis pseudo estático de la estructura en el que se observa la sensibilidad de una de las variables al ser la otra constante. Esto permite construir una gráfica que estudie el comportamiento de la frecuencia a partir de ambas variables, situación que se puede ver en la figura 5.22:

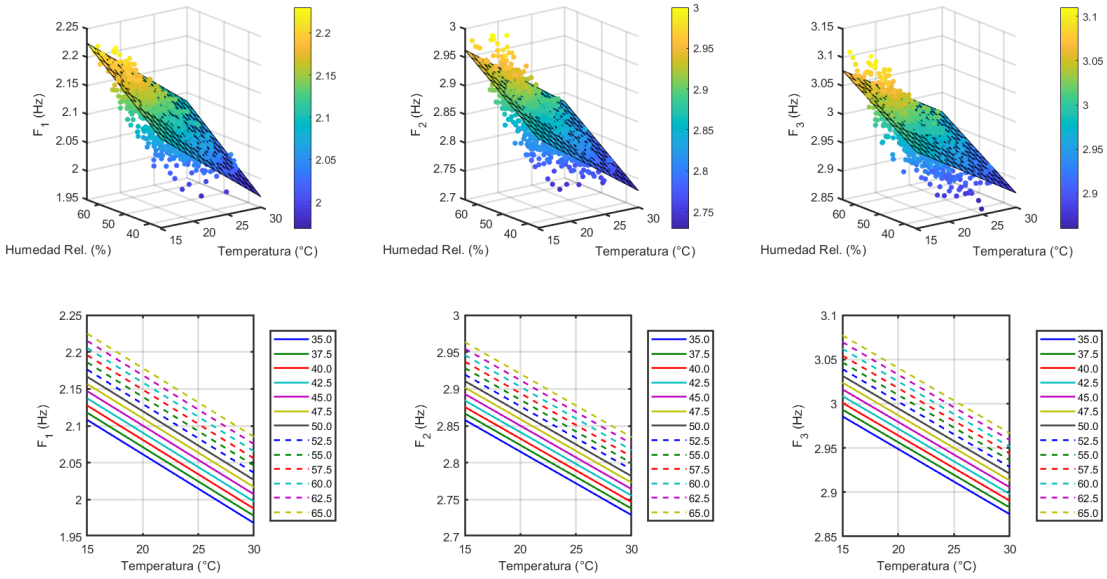

Figura 5.23: Análisis pseudo estático.

Con esto se observa que ante una variable constante, la otra se comportará de manera lineal en el material.

Todo el código que se utilizó en este análisis de variación de frecuencias puede ser encontrado en el Anexo A.14.

Capítulo 6

Conclusiones

Este trabajo de investigación nace a partir del mismo nombre de la Torre Peñuelas que la califica de experimental. Esta condición se le otorga ya que es el edificio más alto de Latinoamérica en ser de madera como material principal y busca comprobar la posibilidad de construir más estructuras de este tipo y en altura para el futuro.

Es por esto que los resultados de este trabajo son bastante importantes ya que en primer lugar se deja operativo un sistema de instrumentación y monitoreo sísmico en la torre para comprobar a futuro como es la respuesta de la estructura ante un evento sísmico.

Por otro lado, la madera al ser un material sacado de un ser vivo que se encuentra en condiciones muy distintas a las que es implementado, puede ver cómo cambian sus propiedades ante nuevas condiciones. Es por esto que este trabajo es muy importante a la hora de ver la respuesta de un edificio como este ante un sismo, ya que el cambio de las frecuencias fundamentales de una estructura se suele asociar a que hubo daño y comportamiento inelástico en ella, sin embargo, como se pudo comprobar en la memoria realizada, estos cambios de frecuencia en la estructura pueden tener un componente ambiental que se debe desagregar para analizar si verdaderamente el edificio se vio más dañado de lo que debía ante un movimiento o si esto proviene simplemente de la naturaleza propia del material.

Del análisis de resultados se puede observar que el modelo espacio estado es el más preciso al obtener un coeficiente de correlación cercano a 0,9 para todas las frecuencias. Sin embargo, su utilización requiere de apoyo para trabajar rápidamente con sus datos. Es por esto que después de observar la figura 5.22 uno distingue rápidamente que, si bien la línea celeste del modelo espacio estado es la que más se ajusta a los resultados mostrados en negro, la función propuesta dependiente únicamente de la temperatura no es una mala estimación para un inicio ya que no se desvía tanto de lo obtenido y es mucho más fácil de interpretar.

Los demás modelos estudiados como el factor de corrección y la humedad normalizada, si bien mejoran el coeficiente de correlación, no se distinguen notablemente de la correlación existente únicamente de la temperatura del día, complejizando notablemente la estimación en una primera instancia.

Es importante destacar que el ambiente en el que se encuentra construido la Torre Experimental Peñuelas es un ambiente costero y se encuentra en una reserva nacional, por lo que

se registraron cambios en el clima que propiciaron una buena base de datos ante cambios de temperatura y humedad relativa. Sin embargo, también es valioso recalcar que la mayor parte de los datos estudiados se encuentran entre el 29 de noviembre y el 29 de diciembre, fechas considerables a la hora de estudiar la madera debido a su época del año en la que se finaliza la primavera y se empieza el verano por lo que se invita a pensar que las temperaturas y humedades se mantendrán bien cercana a sus valores típicos de la época.

Debido a lo anterior, se puede esperar que si bien lo realizado en esta memoria ayuda a realizar una estimación certera de cómo se comportará la estructura ante las condiciones ambientales, este estudio puede continuarse con datos tomados en otoño e invierno para ver como se comporta con temperaturas usuales más bajas y humedades más altas, y de esta forma calibrar mejor el modelo o ver si aun así predice con exactitud la frecuencia del edificio.

Una variante importante que mencionar sobre los datos con los que se trabajó para la investigación, es que se usó únicamente el geófono de los equipos ya que es el que poseía la sensibilidad para captar las micro vibraciones. Esto condiciona ya que el geófono de los equipos no se encuentra en las 3 direcciones principales como si lo hacen los acelerómetros de los equipos, por lo que las principales frecuencias detectadas en los análisis realizados necesariamente se identifican con aquellas cuyo modo asociado tiene algún componente de su movimiento en la vertical.
A pesar de ello, aún es posible identificar las frecuencias naturales de la estructura a partir de mediciones de microvibraciones con geofonos orientados en dirección vertical.

Esto a su vez impidió que se realizaran algunos otros métodos de análisis como el Método del Sub-Espacio Estocástico[3] (SSI), que no arrojó datos de manera clara para reconocer las frecuencias principales.

[3] Para mayor información del funcionamiento de este método se puede revisar Hernández, F. (2009). .ªNÁLISIS DE MÉTODOS DE IDENTIFICACIÓN DE VARIACIÓN DE PROPIEDADES DINÁMICAS". Universidad de Chile

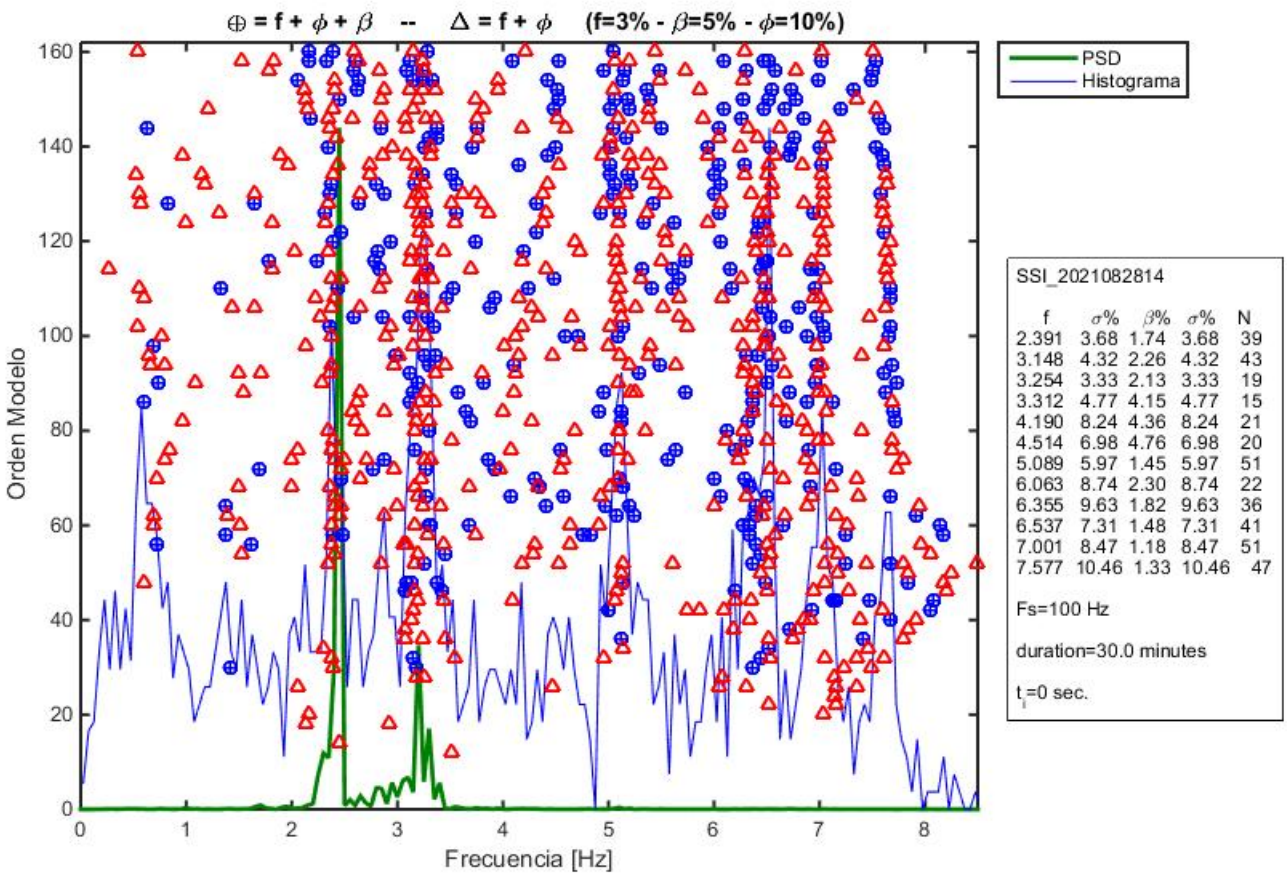

Figura 6.1: Resultados Análisis SSI

Aquí mostró su superioridad el método FDD que fue capaz a partir de los datos de los geófonos verticales encontrar las frecuencias principales de la estructura.

Finalmente, es notable que en el caso en el que se registra un peak más bajo de una de las frecuencias fundamentales y cuando registra el más alto de esa misma frecuencia, puede haber una variación de un 12 %, por lo que el aspecto medioambiental si se debe considerar a la hora de establecer si hubo daño permanente o simplemente hubo cambio debido a la naturaleza del material.

Sobre los Raspberry Shake utilizados, se obtuvo una muy buena respuesta y se logró instrumentar el edificio con un costo cercano al 5-7 % de lo que se necesitaría con los sensores usuales para este tipo de estudios, por lo que abre las puertas a una mayor instrumentación de bajo costo a otros edificios en Chile.

No obstante, la instalación se encontró con varios problemas a la hora de instrumentar, que si bien no se debían a los equipos en sí, deben tenerse muy en cuenta para las futuras instalaciones. El temporizador digital fue una gran solución que permitió solucionar los problemas de continuidad de energía, pero este problema surgió a raíz de que no se proveía la suficiente energía desde el cargador a la batería una vez que esta se cargaba, por lo que

se empezaba a descargar y se agotaba la energía. Por esto, el temporizador solucionó este problema al reiniciar el cargador cada vez que sucedía este problema, sin embargo, se sugiere revisar en futuras instalaciones que el cargador pueda suministrar de la energía necesaria a la batería en todo momento.

Bibliografía

[1] CORMA(-). ¿Por qué madera?. Madera21. https://www.madera21.cl/2757-2/

[2] Centro de Innovación en Madera (2019). Torre experimental Peñuelas. https://madera.uc.cl/es/investigacion/proyectos-de-investigacion/249-torre-penuelas

[3] Kinemetrics(2021). ETNA 2. https://kinemetrics.com/wp-content/uploads/2017/04/datasheet-etna2-accelerograph-kinemetrics.pdf

[4] Bosch(2018). BME280 - Data Sheet. https://cdn.sparkfun.com/assets/e/7/3/b/1/BME280_Datasheet.pdf

[5] Raspberry Shake(2017). Specifications for: Raspberry Shake RS4D. https://manual.raspberryshake.org/specifications.htmltechspecs

[6] Ada, L. (2012). Adding a Real Time Clock to Raspberry Pi. https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi/set-rtc-time

[7] Dev Yadav, R. (2020). How to Install NTP Server and Client(s) on Raspberry Pi with Ubuntu Mate. https://rishabhdevyadav.medium.com/how-to-install-ntp-server-and-client-s-on-ubuntu-18-04-lts-f0562e41d0e1

[8] Aguilar, A. (2012). Estudio Experimental del Comportamiento Dinámico de un Edificio de Viña del Mar Dañado Durante el Terremoto de Febrero de 2010. Universidad de Chile.

[9] Díaz, P. (2018). Análisis de Estructuras Aisladas con el Método de Múltiples Sub-Estructuras. Universidad de Chile.

[10] Belmar, L. (2019). Dispositivo de Disipación de Energía con Sistema de Cables y Poleas. Universidad de Chile.

[11] Hernández, F. (2009). Análisis de Métodos de Identificación de Variación de Propiedades Dinámicas. Universidad de Chile.

[12] Ada, L. (2018). I2C Sensors Devices. https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/i2c-sensors-and-devices

[13] Obspy (Nº de versión 1.2.0). (2022). Python.

[14] Dirección Meteorológica de Chile.(-). Datos de una estación automática al minuto. Meteochile. https://climatologia.meteochile.gob.cl/application/requerimiento/producto/RE5027

Anexos

Anexo A

A.1. Extracción manual

Código programado en matlab que permite pasar un archivo mseed a uno .txt y separar el contenido según la cantidad de tiempo deseada.

```
clear all
[t1,a1]=leemseed2('AM.R4E16.00.EHZ.D.2021.335');
[t2,a2]=leemseed2('AM.R81A0.00.EHZ.D.2021.335');
[t3,a3]=leemseed2('AM.R5642.00.EHZ.D.2021.335');
[t4,a4]=leemseed2('AM.RACCE.00.EHZ.D.2021.335');
[t5,a5]=leemseed2('AM.RD26F.00.EHZ.D.2021.335');
[t6,a6]=leemseed2('AM.RD26F.00.EHZ.D.2021.335');
%a1=a1/3.996500E+08; %m/s

for i=1:48
    x(i)=addtodate(datenum('30-Nov-2021 23:52:00'),30*(i-1),'minute');
    k=max(find(abs(t1-x(i))<0.00001));
    l=max(find(abs(t2-x(i))<0.00001));
    m=max(find(abs(t3-x(i))<0.00001));
    n=max(find(abs(t4-x(i))<0.00001));
    o=max(find(abs(t5-x(i))<0.00001));
    v=max(find(abs(t6-x(i))<0.00001));

    y(i)=addtodate(datenum('1-Dic-2021 00:22:00'),30*(i-1),'minute');
    p=max(find(abs(t1-y(i))<0.00001));
    q=max(find(abs(t2-y(i))<0.00001));
    r=max(find(abs(t3-y(i))<0.00001));
    s=max(find(abs(t4-y(i))<0.00001));
    u=max(find(abs(t5-y(i))<0.00001));
    w=max(find(abs(t6-y(i))<0.00001));

    f=datestr(x(i),'AAAA-mm-DD HH:MM:SS');
    g=split(f);
    h=split(g(2),":");
    i=string(strcat(h(1),h(2),h(3)));
    miarchivo=fopen(strcat('R4E16utc20211201',i,'.txt'),'w'); %abre el archivo txt
    miarchivo2=fopen(strcat('R81A0utc20211201',i,'.txt'),'w');
    miarchivo3=fopen(strcat('R5642utc20211201',i,'.txt'),'w');
    miarchivo4=fopen(strcat('RACCEutc20211201',i,'.txt'),'w');
    miarchivo5=fopen(strcat('RD26Futc20211201',i,'.txt'),'w');
    miarchivo6=fopen(strcat('R3323utc20211201',i,'.txt'),'w');
```

```
37
38    Escritura1=[a1(k:p)];
39    Escritura2=[a2(l:q)];
40    Escritura3=[a3(m:r)];
41    Escritura4=[a4(n:s)];
42    Escritura5=[a5(o:u)];
43    Escritura6=[a6(v:w)];
44
45    fprintf(miarchivo, '%i \n', Escritura1'); %escribe la fecha y los puntos singulares
46    fprintf(miarchivo2, '%i \n', Escritura2');
47    fprintf(miarchivo3, '%i \n', Escritura3');
48    fprintf(miarchivo4, '%i \n', Escritura4');
49    fprintf(miarchivo5, '%i \n', Escritura5');
50    fprintf(miarchivo6, '%i \n', Escritura6');
51
52    fclose(miarchivo);
53    fclose(miarchivo2);
54    fclose(miarchivo3);
55    fclose(miarchivo4);
56    fclose(miarchivo5);
57    fclose(miarchivo6);
58
59end
```

A.2. leemseed2

Código programado en Matlab que permite leer los archivos mseed y pasar los datos registrados a vectores de tiempo y aceleración trabajables en software de programación.

```
1  function [t,a]=leemseed2(archivo)
2% function [t,a]=LeeMseed2(archivo)
3%
4% Lee Mseed y entrega todo el registro concatenado
5%
6% archivo --> Directorio y nombre del archivo seed
7% a(:,1)--> tiempo
8% a(:,2)--> aceleracion (cuentas)
9%
10% Basada en rdmeed.m
11%  Author: François Beauducel <beauducel@ipgp.fr>
12%    Institut de Physique du Globe de Paris
13%  Created: 2010-09-17
14%  Updated: 2014-06-29
15
16
17data=rdmseed(archivo);
18t=cat(1,data.t);
19a=cat(1,data.d);
20
21
22
23function varargout = rdmseed(varargin)
24%RDMSEED Read miniSEED format file.
25%  X = RDMSEED(F) reads file F and returns a M-by-1 structure X containing
```

```
% M blocks ("data records") of a miniSEED file with headers, blockettes,
% and data in dedicated fields, in particular, for each data block X(i):
%           t: time vector (DATENUM format)
%           d: data vector (double)
%   BLOCKETTES: existing blockettes (substructures)
%
% Known blockettes are 100, 500, 1000, 1001 and 2000. Others will be
% ignored with a warning message.
%
% X = RDMSEED(F,ENCODINGFORMAT,WORDORDER,RECORDLENGTH), when file F does
% not include the Blockette 1000 (like Seismic Handler outputs), specifies:
%   - ENCODINGFORMAT: FDSN code (see below); default is 10 = Steim-1;
%   - WORDORDER: 1 = big-endian (default), 0 = little-endian;
%   - RECORDLENGTH: must be a power of 2, at least 256 (default is 4096).
% If the file contains Blockette 1000 (which is mandatory in the SEED
% convention...), these 3 arguments are ignored except with 'force' option.
%
% X = RDMSEED without input argument opens user interface to select the
% file from disk.
%
% [X,I] = RDMSEED(...) returns a N-by-1 structure I with N the detected
% number of different channels, and the following fields:
%    ChannelFullName: channel name,
%       XBlockIndex: channel's vector index into X,
%        ClockDrift: vector of time interval errors, in seconds,
%                    between each data block (relative to sampling
%                    period). This can be compared to "Max Clock Drift"
%                    value of a Blockette 52.
%                       = 0 in perfect case
%                       < 0 tends to overlapping
%                       > 0 tends to gapping
%  OverlapBlockIndex: index of blocks (into X) having a significant
%                    overlap with previous block (less than 0.5
%                    sampling period).
%       OverlapTime: time vector of overlapped blocks (DATENUM format).
%      GapBlockIndex: index of blocks (into X) having a significant gap
%                    with next block (more than 0.5 sampling period).
%           GapTime: time vector of gapped blocks (DATENUM format).
%
% RDMSEED(...) without output arguments plots the imported signal by
% concatenating all the data records, in one single plot if single channel
% is detected, or subplots for multiplexed file (limited to 10 channels).
% Gaps are shown with red stars, overlaps with green circles.
%
% [...] = RDMSEED(F,...,'be') forces big-endian reading (overwrites the
% automatic detection of endianness coding, which fails in some cases).
%
% [...] = RDMSEED(F,...,'notc') disable time correction.
%
% [...] = RDMSEED(F,...,'plot') forces the plot with output arguments.
%
% [...] = RDMSEED(F,...,'v') uses verbose mode (displays additional
% information and warnings when necessary). Use 'vv' for extras, 'vvv'
% for debuging.
%
% Some instructions for usage of the returned structure:
```

```
%
% - to get concatenated time and data vectors from a single-channel file:
%    X = rdmseed(f,'plot');
%    t = cat(1,X.t);
%    d = cat(1,X.d);
%
% - to get the list of channels in a multiplexed file:
%    [X,I] = rdmseed(f);
%    char(I.ChannelFullName)
%
% - to extract the station component n from a multiplexed file:
%    [X,I] = rdmseed(f);
%    k = I(n).XBlockIndex;
%    plot(cat(1,X(k).t),cat(1,X(k).d))
%    datetick('x')
%    title(I(n).ChannelFullName)
%
% Known encoding formats are the following FDSN codes:
%     0: ASCII
%     1: 16-bit integer
%     2: 24-bit integer
%     3: 32-bit integer
%     4: IEEE float32
%     5: IEEE float64
%    10: Steim-1
%    11: Steim-2
%    12: GEOSCOPE 24-bit (untested)
%    13: GEOSCOPE 16/3-bit gain ranged
%    14: GEOSCOPE 16/4-bit gain ranged
%    19: Steim-3 (alpha and untested)
%
% See also MKMSEED to export data in miniSEED format.
%
%
% Author: François Beauducel <beauducel@ipgp.fr>
%   Institut de Physique du Globe de Paris
% Created: 2010-09-17
% Updated: 2014-06-29
%
% Acknowledgments:
%   Ljupco Jordanovski, Jean-Marie Saurel, Mohamed Boubacar, Jonathan Berger,
%   Shahid Ullah, Wayne Crawford, Constanza Pardo, Sylvie Barbier,
%   Robert Chase, Arnaud Lemarchand.
%
% References:
%   IRIS (2010), SEED Reference Manual: SEED Format Version 2.4, May 2010,
%     IFDSN/IRIS/USGS, http://www.iris.edu
%   Trabant C. (2010), libmseed: the Mini-SEED library, IRIS DMC.
%   Steim J.M. (1994), 'Steim' Compression, Quanterra Inc.
%
% History:
%   [2014-06-29]
%     - 24-bit uncompressed format tested (bug correction), thanks to
%       Arnaud Lemarchand.
%   [2014-05-31]
%     - applies the time correction to StartTime and X.t (if needed).
```

```
%       - new option 'notc' to disable time correction.
%       - Geoscope 16/4 format passed real data archive tests.
%       - fixes a problem when plotting multiplexed channels (thanks to
%         Robert Chase).
%    [2014-03-14]
%       - Improved endianness automatic detection (see comments).
%       - Accepts mixed little/big endian encoding in a single file.
%       - minor fixes.
%    [2013-10-25]
%       - Due to obsolete syntax of bitcmp(0,N) in R2013b, replaces all
%         by: 2^N-1 (which is much faster...)
%    [2013-02-15]
%       - Tests also DayOfYear in header to determine automatically
%         little-endian coding of the file.
%       - Adds option 'be' to force big-endian reading (overwrites
%         automatic detection).
%    [2012-12-21]
%       - Adds a verbose mode
%    [2012-04-21]
%       - Correct bug with Steim + little-endian coding
%         (thanks to Shahid Ullah)
%    [2012-03-21]
%       - Adds IDs for warning messages
%    [2011-11-10]
%       - Correct bug with multiple channel name length (thanks to
%         Jonathan Berger)
%    [2011-10-27]
%       - Add LocationIdentifier to X.ChannelFullName
%    [2011-10-24]
%       - Validation of IEEE double encoding (with PQL)
%       - Import/plot data even with file integrity problem (like PQL)
%    [2011-07-21]
%       - Validation of ASCII encoding format (logs)
%       - Blockettes are now stored in substructures below a single
%         field X.BLOCKETTES
%       - Add import of blockettes 500 and 2000
%       - Accept multi-channel files with various data coding
%    [2010-10-16]
%       - Alpha-version of Steim-3 decoding...
%       - Extend output parameters with channel detection
%       - Add gaps and overlaps on plots
%       - Add possibility to force the plot
%    [2010-10-02]
%       - Add the input formats for GEOSCOPE multiplexed old data files
%       - Additional output argument with gap and overlap analysis
%       - Create a plot when no output argument are specified
%       - Optimize script coding (30 times faster STEIM decoding!)
%    [2010-09-28]
%       - Correction of a problem with STEIM-1 nibble 3 decoding (one
%         32-bit difference)
%       - Add reading of files without blockette 1000 with additional
%         input arguments (like Seismic Handler output files).
%       - Uses warning() function instead of fprintf().
%
%  Copyright (c) 2014, François Beauducel, covered by BSD License.
%  All rights reserved.
```

```
%
%  Redistribution and use in source and binary forms, with or without
%  modification, are permitted provided that the following conditions are
%  met:
%
%     * Redistributions of source code must retain the above copyright
%       notice, this list of conditions and the following disclaimer.
%     * Redistributions in binary form must reproduce the above copyright
%       notice, this list of conditions and the following disclaimer in
%       the documentation and/or other materials provided with the distribution
%
%  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    ↪ IS"
%  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
%  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
    ↪ PURPOSE
%  ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS
    ↪ BE
%  LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
%  CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
%  SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
%  INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
%  CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
%  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
%  POSSIBILITY OF SUCH DAMAGE.

if nargin > 6
  error('Too many input arguments.')
end

% global variables shared with sub-functions
global f fid offset le ef wo rl forcebe verbose notc force

% default input arguments
makeplot = 0;  % make plot flag
verbose = 0;   % verbose flag/level
forcebe = 0;   % force big-endian
ef = 10;    % encoding format default
wo = 1;        % word order default
rl = 2^12;     % record length default
force = 0;     % force input argument over blockette 1000 (UNDOCUMENTED)
notc = 0;      % force no time correction (over ActivityFlags)

if nargin < 1
  [filename,pathname] = uigetfile('*','Please select a miniSEED file...');
  f = fullfile(pathname,filename);
else
  f = varargin{1};
end

if ~ischar(f) || ~exist(f,'file')
  error('File %s does not exist.',f);
end

if nargin > 1
  verbose = any(strcmpi(varargin,'v')) + 2*any(strcmpi(varargin,'vv')) ...
```

```
247             + 3*any(strcmpi(varargin,'vvv'));
248   makeplot = any(strcmpi(varargin,'plot'));
249   forcebe = any(strcmpi(varargin,'be'));
250   notc = any(strcmpi(varargin,'notc'));
251   force = any(strcmpi(varargin,'force'));
252 end
253 nargs = (makeplot>0) + (verbose>0) + (forcebe>0) + (notc>0) + (force>0);
254
255
256 if nargin > (1 + nargs)
257   ef = varargin{2};
258   if ~isnumeric(ef) || ~any(ef==[0:5,10:19,30:33])
259     error('Argument ENCODINGFORMAT must be a valid FDSN code value.');
260   end
261 end
262
263 if nargin > (2 + nargs)
264   wo = varargin{3};
265   if ~isnumeric(wo) || (wo ~= 0 && wo ~= 1)
266     error('Argument WORDORDER must be 0 or 1.');
267   end
268 end
269
270 if nargin > (3 + nargs)
271   rl = varargin{4};
272   if ~isnumeric(rl) || rl < 256 || rem(log(rl)/log(2),1) ~= 0
273     error('Argument RECORDLENGTH must be a power of 2 and greater or equal to 256.');
274   end
275 end
276
277 if nargout == 0
278   makeplot = 1;
279 end
280
281 % sensible limits for multiplexed files
282 max_channels = 20;    % absolute max number of channels to plot
283 max_channel_label = 6;  % max. number of channels for y-labels
284
285 % file is opened in Big-Endian encoding (this is encouraged by SEED)
286 fid = fopen(f,'rb','ieee-be');
287 le = 0;
288
289 % --- tests if the header is mini-SEED
290 % the 7th character must be one of the "data header/quality indicator", usually 'D'
291 header = fread(fid,20,'*char');
292 if ~ismember(header(7),'DRMQ')
293   if ismember(header(7),'VAST')
294     s = ' (seems to be a SEED Volume)';
295   else
296     s = '';
297   end
298   error('File is not in mini-SEED format%s. Cannot read it.',s);
299 end
300
301 i = 1;
302 offset = 0;
```

```
303
304 while offset >= 0
305   X(i) = read_data_record;
306   i = i + 1;
307 end
308
309 fclose(fid);
310
311 if nargout > 0
312   varargout{1} = X;
313 end
314
315 % --- analyses data
316 if makeplot || nargout > 1
317
318   % test if the file is multiplexed or a single channel
319   un = unique(cellstr(char(X.ChannelFullName)));
320   nc = numel(un);
321   for i = 1:nc
322     k = find(strcmp(cellstr(char(X.ChannelFullName)),un{i}));
323     I(i).ChannelFullName = X(k(1)).ChannelFullName;
324     I(i).XBlockIndex = k;
325     I(i).ClockDrift = ([diff(cat(1,X(k).RecordStartTimeMATLAB));NaN]*86400 - cat(1,X(k).
      ↪ NumberSamples)./cat(1,X(k).SampleRate))./cat(1,X(k).NumberSamples);
326     I(i).OverlapBlockIndex = k(find(I(i).ClockDrift.*cat(1,X(k).NumberSamples).*cat(1,X(k).SampleRate)
      ↪  < -.5) + 1);
327     I(i).OverlapTime = cat(1,X(I(i).OverlapBlockIndex).RecordStartTimeMATLAB);
328     I(i).GapBlockIndex = k(find(I(i).ClockDrift.*cat(1,X(k).NumberSamples).*cat(1,X(k).SampleRate) >
      ↪ .5) + 1);
329     I(i).GapTime = cat(1,X(I(i).GapBlockIndex).RecordStartTimeMATLAB);
330   end
331 end
332 if nargout > 1
333   varargout{2} = I;
334 end
335
336 % --- plots the data
337 if makeplot
338
339   figure
340
341   xlim = [min(cat(1,X.t)),max(cat(1,X.t))];
342
343   % test if all data records have the same length
344   rl = unique(cat(1,X.DataRecordSize));
345   if numel(rl) == 1
346     rl_text = sprintf('%d bytes',rl);
347   else
348     rl_text = sprintf('%d-%d bytes',min(rl),max(rl));
349   end
350
351   % test if all data records have the same sampling rate
352   sr = unique(cat(1,X.SampleRate));
353   if numel(sr) == 1
354     sr_text = sprintf('%g Hz',sr);
355   else
```

```
356     sr_text = sprintf('%d # samp. rates',numel(sr));
357   end
358
359   % test if all data records have the same encoding format
360   ef = unique(cellstr(cat(1,X.EncodingFormatName)));
361   if numel(ef) == 1
362     ef_text = sprintf('%s',ef{:});
363   else
364     ef_text = sprintf('%d different encod. formats',numel(ef));
365   end
366
367   if nc == 1
368     plot(cat(1,X.t),cat(1,X.d))
369     hold on
370     for i = 1:length(I.GapBlockIndex)
371       plot(I.GapTime(i),X(I.GapBlockIndex(i)).d(1),'*r')
372     end
373     for i = 1:length(I.OverlapBlockIndex)
374       plot(I.OverlapTime(i),X(I.OverlapBlockIndex(i)).d(1),'og')
375     end
376     hold off
377     set(gca,'XLim',xlim)
378     datetick('x','keeplimits')
379     grid on
380     xlabel(sprintf('Time\n(%s to %s)',datestr(xlim(1)),datestr(xlim(2))))
381     ylabel('Counts')
382     title(sprintf('mini-SEED file "%s"\n%s (%d rec. @ %s - %g samp. @ %s - %s)', ...
383       f,un{1},length(X),rl_text,numel(cat(1,X.d)),sr_text,ef_text),'Interpreter','none')
384   else
385     % plot is done only for real data channels...
386     if nc > max_channels
387       warning('Plot has been limited to %d channels (over %d). See help to manage multiplexed file.', ...
388         max_channels,nc);
389       nc = max_channels;
390     end
391     for i = 1:nc
392       subplot(nc*2,1,i*2 + (-1:0))
393       k = I(i).XBlockIndex;
394       if ~any(strcmp('ASCII',cellstr(cat(1,X(k).EncodingFormatName))))
395         plot(cat(1,X(k).t),cat(1,X(k).d))
396         hold on
397         for ii = 1:length(I(i).GapBlockIndex)
398           plot(I(i).GapTime(ii),X(I(i).GapBlockIndex(ii)).d(1),'*r')
399         end
400         for ii = 1:length(I(i).OverlapBlockIndex)
401           plot(I(i).OverlapTime(ii),X(I(i).OverlapBlockIndex(ii)).d(1),'og')
402         end
403         hold off
404       end
405       set(gca,'XLim',xlim,'FontSize',8)
406       h = ylabel(un{i},'Interpreter','none');
407       if nc > max_channel_label
408         set(gca,'YTick',[])
409         set(h,'Rotation',0,'HorizontalAlignment','right','FontSize',8)
410       end
411       datetick('x','keeplimits')
```

```
      set(gca,'XTickLabel',[])
      grid on
      if i == 1
        title(sprintf('mini−SEED file "%s"\n%d channels (%d rec. @ %s − %g data − %s − %s)', ...
          f,length(un),length(X),rl_text,numel(cat(1,X(k).d)),sr_text,ef_text),'Interpreter','none')
      end
      if i == nc
        datetick('x','keeplimits')
        xlabel(sprintf('Time\n(%s to %s)',datestr(xlim(1)),datestr(xlim(2))))
      end
    end
    v = version;
    if str2double(v(1))>=7
      linkaxes(findobj(gcf,'type','axes'),'x')
    end
  end
end


%
    ↪ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ↪
function D = read_data_record
% read_data_record uses global variables f, fid, offset, le, ef, wo, rl,
%  and verbose. It reads a data record and returns a structure D.

global f fid offset le ef wo rl verbose notc force

fseek(fid,offset,'bof');

% −−− read fixed section of Data Header (48 bytes)
D.SequenceNumber         = fread(fid,6,'∗char')';
D.DataQualityIndicator   = fread(fid,1,'∗char');
D.ReservedByte           = fread(fid,1,'∗char');
D.StationIdentifierCode  = fread(fid,5,'∗char')';
D.LocationIdentifier     = fread(fid,2,'∗char')';
D.ChannelIdentifier      = fread(fid,3,'∗char')';
D.NetworkCode            = fread(fid,2,'∗char')';
D.ChannelFullName = sprintf('%s:%s:%s:%s',deblank(D.NetworkCode), ...
  deblank(D.StationIdentifierCode),deblank(D.LocationIdentifier), ...
  deblank(D.ChannelIdentifier));

% Start Time decoding
[D.RecordStartTime,swapflag] = readbtime;
if swapflag
  if le
    machinefmt = 'ieee−be';
    le = 0;
  else
    machinefmt = 'ieee−le';
    le = 1;
  end
  position = ftell(fid);
  fclose(fid);
  fid = fopen(f,'rb',machinefmt);
  fseek(fid,position,'bof');
```

```
466   if verbose > 0
467      warning('RDMSEED:DataIntegrity', ...
468         'Sequence # %s: need to switch file encoding to %s...\n', ...
469         D.SequenceNumber,machinefmt);
470   end
471end
472
473D.NumberSamples           = fread(fid,1,'uint16');
474
475% Sample Rate decoding
476SampleRateFactor        = fread(fid,1,'int16');
477SampleRateMultiplier    = fread(fid,1,'int16');
478if SampleRateFactor > 0
479   if SampleRateMultiplier >= 0
480      D.SampleRate = SampleRateFactor*SampleRateMultiplier;
481   else
482      D.SampleRate = -1*SampleRateFactor/SampleRateMultiplier;
483   end
484else
485   if SampleRateMultiplier >= 0
486      D.SampleRate = -1*SampleRateMultiplier/SampleRateFactor;
487   else
488      D.SampleRate = 1/(SampleRateFactor*SampleRateMultiplier);
489   end
490end
491
492D.ActivityFlags          = fread(fid,1,'uint8');
493D.IOFlags                = fread(fid,1,'uint8');
494D.DataQualityFlags       = fread(fid,1,'uint8');
495D.NumberBlockettesFollow = fread(fid,1,'uint8');
496D.TimeCorrection         = fread(fid,1,'int32');   % Time correction in 0.0001 s
497D.OffsetBeginData        = fread(fid,1,'uint16');
498D.OffsetFirstBlockette   = fread(fid,1,'uint16');
499
500% --- read the blockettes
501OffsetNextBlockette = D.OffsetFirstBlockette;
502
503D.BLOCKETTES = [];
504b2000 = 0;  % Number of Blockette 2000
505
506for i = 1:D.NumberBlockettesFollow
507   fseek(fid,offset + OffsetNextBlockette,'bof');
508   BlocketteType = fread(fid,1,'uint16');
509
510   switch BlocketteType
511
512      case 1000
513         % BLOCKETTE 1000 = Data Only SEED (8 bytes)
514         OffsetNextBlockette = fread(fid,1,'uint16');
515         D.BLOCKETTES.B1000.EncodingFormat = fread(fid,1,'uint8');
516         D.BLOCKETTES.B1000.WordOrder = fread(fid,1,'uint8');
517         D.BLOCKETTES.B1000.DataRecordLength = fread(fid,1,'uint8');
518         D.BLOCKETTES.B1000.Reserved = fread(fid,1,'uint8');
519
520      case 1001
521         % BLOCKETTE 1001 = Data Extension (8 bytes)
```

```
522         OffsetNextBlockette = fread(fid,1,'uint16');
523         D.BLOCKETTES.B1001.TimingQuality = fread(fid,1,'uint8');
524         D.BLOCKETTES.B1001.Micro_sec = fread(fid,1,'int8');
525         D.BLOCKETTES.B1001.Reserved = fread(fid,1,'uint8');
526         D.BLOCKETTES.B1001.FrameCount = fread(fid,1,'uint8');
527
528      case 100
529         % BLOCKETTE 100 = Sample Rate (12 bytes)
530         OffsetNextBlockette = fread(fid,1,'uint16');
531         D.BLOCKETTES.B100.ActualSampleRate = fread(fid,1,'float32');
532         D.BLOCKETTES.B100.Flags = fread(fid,1,'uint8');
533         D.BLOCKETTES.B100.Reserved = fread(fid,1,'uint8');
534
535      case 500
536         % BLOCKETTE 500 = Timing (200 bytes)
537         OffsetNextBlockette = fread(fid,1,'uint16');
538         D.BLOCKETTES.B500.VCOCorrection = fread(fid,1,'float32');
539         D.BLOCKETTES.B500.TimeOfException = readbtime;
540         D.BLOCKETTES.B500.MicroSec = fread(fid,1,'int8');
541         D.BLOCKETTES.B500.ReceptionQuality = fread(fid,1,'uint8');
542         D.BLOCKETTES.B500.ExceptionCount = fread(fid,1,'uint16');
543         D.BLOCKETTES.B500.ExceptionType = fread(fid,16,'*char')';
544         D.BLOCKETTES.B500.ClockModel = fread(fid,32,'*char')';
545         D.BLOCKETTES.B500.ClockStatus = fread(fid,128,'*char')';
546
547      case 2000
548         % BLOCKETTE 2000 = Opaque Data (variable length)
549         b2000 = b2000 + 1;
550         OffsetNextBlockette = fread(fid,1,'uint16');
551         BlocketteLength = fread(fid,1,'uint16');
552         OffsetOpaqueData = fread(fid,1,'uint16');
553         D.BLOCKETTES.B2000(b2000).RecordNumber = fread(fid,1,'uint32');
554         D.BLOCKETTES.B2000(b2000).DataWordOrder = fread(fid,1,'uint8');
555         D.BLOCKETTES.B2000(b2000).Flags = fread(fid,1,'uint8');
556         NumberHeaderFields = fread(fid,1,'uint8');
557         HeaderFields = splitfield(fread(fid,OffsetOpaqueData-15,'*char')','~');
558         D.BLOCKETTES.B2000(b2000).HeaderFields = HeaderFields(1:NumberHeaderFields);
559         % Opaque data are stored as a single char string, but must be
560         % decoded using appropriate format (e.g., Quanterra Q330)
561         D.BLOCKETTES.B2000(b2000).OpaqueData = fread(fid,BlocketteLength-OffsetOpaqueData,'*
    ↪ char')';
562
563      otherwise
564         OffsetNextBlockette = fread(fid,1,'uint16');
565
566         if verbose > 0
567            warning('RDMSEED:UnknownBlockette', ...
568               'Unknown Blockette number %d (%s)!\n', ...
569               BlocketteType,D.ChannelFullName);
570         end
571   end
572end
573
574% --- read the data stream
575fseek(fid,offset + D.OffsetBeginData,'bof');
576
```

```
if ~force && isfield(D.BLOCKETTES,'B1000')
  EncodingFormat = D.BLOCKETTES.B1000.EncodingFormat;
  WordOrder = D.BLOCKETTES.B1000.WordOrder;
  D.DataRecordSize = 2^D.BLOCKETTES.B1000.DataRecordLength;
else
  EncodingFormat = ef;
  WordOrder = wo;
  D.DataRecordSize = rl;
end

uncoded = 0;

D.d = NaN;
D.t = NaN;

switch EncodingFormat

  case 0
    % --- decoding format: ASCII text
    D.EncodingFormatName = {'ASCII'};
    D.d = fread(fid,D.DataRecordSize - D.OffsetBeginData,'*char')';

  case 1
    % --- decoding format: 16-bit integers
    D.EncodingFormatName = {'INT16'};
    dd = fread(fid,ceil((D.DataRecordSize - D.OffsetBeginData)/2),'*int16');
    if xor(~WordOrder,le)
      dd = swapbytes(dd);
    end
    D.d = dd(1:D.NumberSamples);

  case 2
    % --- decoding format: 24-bit integers
    D.EncodingFormatName = {'INT24'};
    dd = fread(fid,ceil((D.DataRecordSize - D.OffsetBeginData)/3),'bit24=>int32');
    if xor(~WordOrder,le)
      dd = swapbytes(dd);
    end
    D.d = dd(1:D.NumberSamples);

  case 3
    % --- decoding format: 32-bit integers
    D.EncodingFormatName = {'INT32'};
    dd = fread(fid,ceil((D.DataRecordSize - D.OffsetBeginData)/4),'*int32');
    if xor(~WordOrder,le)
      dd = swapbytes(dd);
    end
    D.d = dd(1:D.NumberSamples);

  case 4
    % --- decoding format: IEEE floating point
    D.EncodingFormatName = {'FLOAT32'};
    dd = fread(fid,ceil((D.DataRecordSize - D.OffsetBeginData)/4),'*float');
    if xor(~WordOrder,le)
      dd = swapbytes(dd);
    end
```

```
633     D.d = dd(1:D.NumberSamples);
634
635   case 5
636     % --- decoding format: IEEE double precision floating point
637     D.EncodingFormatName = {'FLOAT64'};
638     dd = fread(fid,ceil((D.DataRecordSize - D.OffsetBeginData)/8),'*double');
639     if xor(~WordOrder,le)
640       dd = swapbytes(dd);
641     end
642     D.d = dd(1:D.NumberSamples);
643
644   case {10,11,19}
645     % --- decoding formats: STEIM-1 and STEIM-2 compression
646     %     (c) Joseph M. Steim, Quanterra Inc., 1994
647     steim = find(EncodingFormat==[10,11,19]);
648     D.EncodingFormatName = {sprintf('STEIM%d',steim)};
649
650     % Steim compression decoding strategy optimized for Matlab
651     % -- by F. Beauducel, October 2010 --
652     %
653     %  1. loads all data into a single 16xM uint32 array
654     %  2. gets all nibbles from the first row splitted into 2-bit values
655     %  3. for each possible nibble value, selects (find) and decodes
656     %     (bitsplit) all the corresponding words, and stores results
657     %     in a 4xN (STEIM1) or 7xN (STEIM2) array previously filled with
658     %     NaN's. For STEIM2 with nibbles 2 or 3, decodes also dnib values
659     %     (first 2-bit of the word)
660     %  5. reduces this array with non-NaN values only
661     %  6. integrates with cumsum
662     %
663     % This method is about 30 times faster than a 'C-like' loops coding...
664
665     frame32 = fread(fid,[16,(D.DataRecordSize - D.OffsetBeginData)/64],'*uint32');
666     if xor(~WordOrder,le)
667       frame32 = swapbytes(frame32);
668     end
669
670     % specific processes for STEIM-3
671     if steim == 3
672       % first bit = 1 means second differences
673       SecondDiff = bitshift(frame32(1,:),-31);
674       % checks for "squeezed flag"... and replaces frame32(1,:)
675       squeezed = bitand(bitshift(frame32(1,:),-24),127);
676       k = find(bitget(squeezed,7));
677       if ~isempty(k)
678         moredata24 = bitand(frame32(1,k),16777215);
679         k = find(squeezed == 80);  % upper nibble 8-bit = 0x50
680         if ~isempty(k)
681           frame32(1,k) = hex2dec('15555555');
682         end
683         k = find(squeezed == 96);  % upper nibble 8-bit = 0x60
684         if ~isempty(k)
685           frame32(1,k) = hex2dec('2aaaaaaa');
686         end
687         k = find(squeezed == 112); % upper nibble 8-bit = 0x70
688         if ~isempty(k)
```

```
689             frame32(1,k) = hex2dec('3ffffff');
690           end
691         end
692       end
693
694       % nibbles is an array of the same size as frame32...
695       nibbles = bitand(bitshift(repmat(frame32(1,:),16,1),repmat(-30:2:0,size(frame32,2),1)'),3);
696       x0 = bitsign(frame32(2,1),32);    % forward integration constant
697       xn = bitsign(frame32(3,1),32);    % reverse integration constant
698
699       switch steim
700
701       case 1
702         % STEIM-1: 3 cases following the nibbles
703         ddd = NaN*ones(4,numel(frame32));    % initiates array with NaN
704         k = find(nibbles == 1);        % nibble = 1 : four 8-bit differences
705         if ~isempty(k)
706           ddd(1:4,k) = bitsplit(frame32(k),32,8);
707         end
708         k = find(nibbles == 2);        % nibble = 2 : two 16-bit differences
709         if ~isempty(k)
710           ddd(1:2,k) = bitsplit(frame32(k),32,16);
711         end
712         k = find(nibbles == 3);        % nibble = 3 : one 32-bit difference
713         if ~isempty(k)
714           ddd(1,k) = bitsign(frame32(k),32);
715         end
716
717       case 2
718         % STEIM-2: 7 cases following the nibbles and dnib
719         ddd = NaN*ones(7,numel(frame32));    % initiates array with NaN
720         k = find(nibbles == 1);        % nibble = 1 : four 8-bit differences
721         if ~isempty(k)
722           ddd(1:4,k) = bitsplit(frame32(k),32,8);
723         end
724         k = find(nibbles == 2);        % nibble = 2 : must look in dnib
725         if ~isempty(k)
726           dnib = bitshift(frame32(k),-30);
727           kk = k(dnib == 1);       % dnib = 1 : one 30-bit difference
728           if ~isempty(kk)
729             ddd(1,kk) = bitsign(frame32(kk),30);
730           end
731           kk = k(dnib == 2);       % dnib = 2 : two 15-bit differences
732           if ~isempty(kk)
733             ddd(1:2,kk) = bitsplit(frame32(kk),30,15);
734           end
735           kk = k(dnib == 3);       % dnib = 3 : three 10-bit differences
736           if ~isempty(kk)
737             ddd(1:3,kk) = bitsplit(frame32(kk),30,10);
738           end
739         end
740         k = find(nibbles == 3);          % nibble = 3 : must look in dnib
741         if ~isempty(k)
742           dnib = bitshift(frame32(k),-30);
743           kk = k(dnib == 0);       % dnib = 0 : five 6-bit difference
744           if ~isempty(kk)
```

```
745           ddd(1:5,kk) = bitsplit(frame32(kk),30,6);
746         end
747         kk = k(dnib == 1);       % dnib = 1 : six 5-bit differences
748         if ~isempty(kk)
749           ddd(1:6,kk) = bitsplit(frame32(kk),30,5);
750         end
751         kk = k(dnib == 2);       % dnib = 2 : seven 4-bit differences (28 bits!)
752         if ~isempty(kk)
753           ddd(1:7,kk) = bitsplit(frame32(kk),28,4);
754         end
755       end
756
757     case 3   % *** STEIM-3 DECODING IS ALPHA AND UNTESTED ***
758       % STEIM-3: 7 cases following the nibbles
759       ddd = NaN*ones(9,numel(frame32));   % initiates array with NaN
760       k = find(nibbles == 0);          % nibble = 0 : two 16-bit differences
761       if ~isempty(k)
762         ddd(1:2,k) = bitsplit(frame32(k),32,16);
763       end
764       k = find(nibbles == 1);          % nibble = 1 : four 8-bit differences
765       if ~isempty(k)
766         ddd(1:4,k) = bitsplit(frame32(k),32,8);
767       end
768       k = find(nibbles == 2);          % nibble = 2 : must look even dnib
769       if ~isempty(k)
770         dnib2 = bitshift(frame32(k(2:2:end)),-30);
771         w60 = bitand(frame32(k(2:2:end)),1073741823) ...
772           + bitshift(bitand(frame32(k(1:2:end)),1073741823),30);   % concatenates two 30-bit words
773         kk = find(dnib2 == 0);     % dnib = 0: five 12-bit differences (60 bits)
774         if ~isempty(kk)
775           ddd(1:5,k(2*kk)) = bitsplit(w60,60,12);
776         end
777         kk = find(dnib2 == 1);     % dnib = 1: three 20-bit differences (60 bits)
778         if ~isempty(kk)
779           ddd(1:3,k(2*kk)) = bitsplit(w60,60,20);
780         end
781       end
782       k = find(nibbles == 3);          % nibble = 3 : must look 3rd bit
783       if ~isempty(k)
784         dnib = bitshift(frame32(k),-27);
785         kk = k(dnib == 24);      % dnib = 11000 : nine 3-bit differences (27 bits)
786         if ~isempty(kk)
787           ddd(1:9,kk) = bitsplit(frame32(kk),27,3);
788         end
789         kk = k(dnib == 25);      % dnib = 11001 : Not A Difference
790         if ~isempty(kk)
791           ddd(1,kk) = bitsign(frame32(kk),27);
792         end
793         kk = k(dnib > 27);       % dnib = 111.. : 29-bit sample (29 bits)
794         if ~isempty(kk)
795           ddd(1,kk) = bitsign(frame32(kk),29);
796         end
797       end
798     end
799
800     % Little-endian coding: needs to swap bytes
```

```
801     if ~WordOrder
802       ddd = flipud(ddd);
803     end
804     dd = ddd(~isnan(ddd));      % reduces initial array ddd: dd is non-NaN values of ddd
805
806     % controls the number of samples
807     if numel(dd) ~= D.NumberSamples
808       if verbose > 1
809         warning('RDMSEED:DataIntegrity','Problem in %s sequence # %s [%s]: number of samples in
    ↪ header (%d) does not equal data (%d).\n',D.EncodingFormatName{:},D.SequenceNumber,D.
    ↪ RecordStartTimeISO,D.NumberSamples,numel(dd));
810       end
811       if numel(dd) < D.NumberSamples
812         D.NumberSamples = numel(dd);
813       end
814     end
815
816     % rebuilds the data vector by integrating the differences
817     D.d = cumsum([x0;dd(2:D.NumberSamples)]);
818
819     % controls data integrity...
820     if D.d(end) ~= xn
821       warning('RDMSEED:DataIntegrity','Problem in %s sequence # %s [%s]: data integrity check failed
    ↪ , last_data=%d, Xn=%d.\n',D.EncodingFormatName{:},D.SequenceNumber,D.
    ↪ RecordStartTimeISO,D.d(end),xn);
822     end
823
824     % for debug purpose...
825     if verbose > 2
826       D.dd = dd;
827       D.nibbles = nibbles;
828       D.x0 = x0;
829       D.xn = xn;
830     end
831
832   case 12
833     % --- decoding format: GEOSCOPE multiplexed 24-bit integer
834     D.EncodingFormatName = {'GEOSCOPE24'};
835     dd = fread(fid,(D.DataRecordSize - D.OffsetBeginData)/3,'bit24=>double');
836     if xor(~WordOrder,le)
837       dd = swapbytes(dd);
838     end
839     D.d = dd(1:D.NumberSamples);
840
841   case {13,14}
842     % --- decoding format: GEOSCOPE multiplexed 16/3 and 16/4 bit gain ranged
843     %  (13): 16/3-bit (bit 15 is unused)
844     %  (14): 16/4-bit
845     %  bits 15-12 = 3 or 4-bit gain exponent (positive)
846     %  bits 11-0 = 12-bit mantissa (positive)
847     %  => data = (mantissa - 2048) / 2^gain
848     geoscope = 7 + 8*(EncodingFormat==14); % mask for gain exponent
849     D.EncodingFormatName = {sprintf('GEOSCOPE16-%d',EncodingFormat-10)};
850     dd = fread(fid,(D.DataRecordSize - D.OffsetBeginData)/2,'*uint16');
851     if xor(~WordOrder,le)
852       dd = swapbytes(dd);
```

```
853     end
854     dd = (double(bitand(dd,2^12-1))-2^11)./2.^double(bitand(bitshift(dd,-12),geoscope));
855     D.d = dd(1:D.NumberSamples);
856
857   case 15
858     % --- decoding format: US National Network compression
859     D.EncodingFormatName = {'USNN'};
860     uncoded = 1;
861
862   case 16
863     % --- decoding format: CDSN 16-bit gain ranged
864     D.EncodingFormatName = {'CDSN'};
865     uncoded = 1;
866
867   case 17
868     % --- decoding format: Graefenberg 16-bit gain ranged
869     D.EncodingFormatName = {'GRAEFENBERG'};
870     uncoded = 1;
871
872   case 18
873     % --- decoding format: IPG - Strasbourg 16-bit gain ranged
874     D.EncodingFormatName = {'IPGS'};
875     uncoded = 1;
876
877   case 30
878     % --- decoding format: SRO format
879     D.EncodingFormatName = {'SRO'};
880     uncoded = 1;
881
882   case 31
883     % --- decoding format: HGLP format
884     D.EncodingFormatName = {'HGLP'};
885     uncoded = 1;
886
887   case 32
888     % --- decoding format: DWWSSN gain ranged format
889     D.EncodingFormatName = {'DWWSSN'};
890     uncoded = 1;
891
892   case 33
893     % --- decoding format: RSTN 16-bit gain ranged
894     D.EncodingFormatName = {'RSTN'};
895     uncoded = 1;
896
897   otherwise
898     D.EncodingFormatName = {sprintf('** Unknown (%d) **',EncodingFormat)};
899     uncoded = 1;
900
901 end
902
903 if uncoded
904   error('Sorry, the encoding format "%s" is not yet implemented.',D.EncodingFormatName);
905 end
906
907 % Applies time correction (if needed)
908 D.RecordStartTimeMATLAB = datenum(double([D.RecordStartTime(1),0,D.RecordStartTime(2:5)])) ...
```

```
  + (~notc & bitand(D.ActivityFlags,2) == 0)*D.TimeCorrection/1e4/86400;
tv = datevec(D.RecordStartTimeMATLAB);
doy = datenum(tv(1:3)) - datenum(tv(1),1,0);
D.RecordStartTime = [tv(1),doy,tv(4:5),round(tv(6)*1e4)/1e4];
D.RecordStartTimeISO = sprintf('%4d-%03d %02d:%02d:%07.4f',D.RecordStartTime);

D.t = D.RecordStartTimeMATLAB;

% makes the time vector and applies time correction (if needed)
if EncodingFormat > 0
  D.t = D.t + (0:(D.NumberSamples-1))'/(D.SampleRate*86400);
end

offset = ftell(fid);
fread(fid,1,'char'); % this is to force EOF=1 on last record.
if feof(fid)
  offset = -1;
end



%
    ↪ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ↪
function c = splitfield(s,d)
% splitfield(S) splits string S of D-character separated field names
C = textscan(s,'%s','Delimiter',d);
c = C{1};


%
    ↪ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ↪
function [d,swapflag] = readbtime
% readbtime reads BTIME structure from current opened file and returns
%  D = [YEAR,DAY,HOUR,MINUTE,SECONDS]

global fid forcebe

Year      = fread(fid,1,'*uint16');
DayOfYear   = fread(fid,1,'*uint16');
Hours     = fread(fid,1,'uint8');
Minutes      = fread(fid,1,'uint8');
Seconds      = fread(fid,1,'uint8');
fseek(fid,1,0);    % skip 1 byte (unused)
Seconds0001 = fread(fid,1,'*uint16');

% Automatic detection of little/big-endian encoding
% -- by F. Beauducel, March 2014 --
%
% If the 2-byte day is >= 512, the file is not opened in the correct
% endianness. If the day is 1 or 256, there is a possible byte-swap and we
% need to check also the year; but we need to consider what is a valid year:
% - years from 1801 to 2047 are OK (swapbytes >= 2312)
% - years from 2048 to 2055 are OK (swapbytes <= 1800)
% - year 2056 is ambiguous (swapbytes = 2056)
```

```
% − years from 2057 to 2311 are OK (swapbytes >= 2312)
% − year 1799 is ambiguous (swapbytes = 1799)
% − year 1800 is suspicious (swapbytes = 2055)
%
% Thus, the only cases for which we are 'sure' there is a byte−swap, are:
% − day >= 512
% − (day == 1 or day == 256) and (year < 1799 or year > 2311)
%
% Note: in IRIS libmseed, the test is only year>2050 or year<1920.
if ~forcebe && (DayOfYear >= 512 || (ismember(DayOfYear,[1,256]) && (Year > 2311 || Year < 1799)))
  swapflag = 1;
  Year = swapbytes(Year);
  DayOfYear = swapbytes(DayOfYear);
  Seconds0001 = swapbytes(Seconds0001);
else
  swapflag = 0;
end
d = [double(Year),double(DayOfYear),Hours,Minutes,Seconds + double(Seconds0001)/1e4];


%
    ↪ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ↪
function d = bitsplit(x,b,n)
% bitsplit(X,B,N) splits the B−bit number X into signed N−bit array
%  X must be unsigned integer class
%  N ranges from 1 to B
%  B is a multiple of N

sign = repmat((b:−n:n)',1,size(x,1));
x = repmat(x',b/n,1);
d = double(bitand(bitshift(x,flipud(sign−b)),2^n−1)) ...
  − double(bitget(x,sign))*2^n;



%
    ↪ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ↪
function d = bitsign(x,n)
% bitsign(X,N) returns signed double value from unsigned N−bit number X.
% This is equivalent to bitsplit(X,N,N), but the formula is simplified so
% it is much more efficient

d = double(bitand(x,2^n−1)) − double(bitget(x,n)).*2^n;
```

A.3. Codigo principal automático

Código al interior de la Raspberry pi 3 que permite analizar los archivos extraidos con el método FDD para luego registrar los resultados en un archivo .txt

```

clear all
close all
```

```
clc
sigrafica=1;
sipsd=0;



R4E16=detrend(load('R4E16utc20211021210000g.txt'))/3.996500E+08; %m/s;;
R81A0=detrend(load('R81A0utc20211021210000g.txt'))/3.996500E+08; %m/s;;
R3323=detrend(load('R3323utc20211021210000g.txt'))/3.996500E+08; %m/s;;
RACCE=detrend(load('RACCEutc20211021210000g.txt'))/3.996500E+08; %m/s;;
RD26F=detrend(load('RD26Futc20211021210000g.txt'))/3.996500E+08; %m/s;;


Fs=100;

t=(0:1:length(RACCE)-1)*1/Fs;
t2=(0:1:length(R4E16)-1)*1/Fs;

if sigrafica
subplot(5,1,1)
plot(t,R3323)
grid on
ylabel('R3323-PZ')

subplot(5,1,2)
plot(t2,R81A0)
grid on
ylabel('R81A0-ESQ')

subplot(5,1,3)
plot(t,RACCE)
grid on
ylabel('RACCE-ESC')


subplot(5,1,4)
plot(t2,R4E16)
grid on
ylabel('R4E16-P2ESQ2')


subplot(5,1,5)
plot(t2,RD26F)
grid on
ylabel('RD26F-P4ESC')
xlabel('Tiempo (seg)')

end

freq_vector=0:0.001:10;

window=hann(30*Fs);

if sipsd
PR4E16=mipsd(detrend(R4E16),Fs,freq_vector,window);
```

```
PR81A0=mipsd(detrend(R81A0),Fs,freq_vector,window);
PR3323=mipsd(detrend(R3323),Fs,freq_vector,window);
PRACCE=mipsd(detrend(RACCE),Fs,freq_vector,window);
PRD26F=mipsd(detrend(RD26F),Fs,freq_vector,window);

if sigrafica
figure
subplot(5,1,1)
plot(freq_vector,PR3323)
grid on
ylabel('R3323-PZ')

subplot(5,1,2)
plot(freq_vector,PR81A0)
grid on
ylabel('R81A0-ESQ')

subplot(5,1,3)
plot(freq_vector,PRACCE)
grid on
ylabel('RACCE-P2-ESC')

subplot(5,1,4)
plot(freq_vector,PR4E16)
grid on
ylabel('R4E16-ESQ2')


subplot(5,1,5)
plot(freq_vector,PRD26F)
grid on
ylabel('RD26F-P4ESC')
xlabel('Frequency (Hz)')

end
end

%% analisis con FDD (Frequency Domain Decomposition)

si_fdd=1;


if si_fdd
   %plotdef(1.5,1.5)
   timebreak=[];
   WINDOW=hann(20*Fs);
   NOVERLAP='';
   freq_vector=0:0.01:20;
   graph_fdd=1;
%     [F,vs,vectores,Psd]= fdd3(a,Fs,WINDOW,NOVERLAP,freq_vector,graph_fdd);
%     freq_fdd=[2.18,1;2.97,1;3.15,1;4.86,1;6.03,1;7.37,1;9.50,1;8.88,1;10.5,1;13.19,1;15.22,1;16.33,1]; %[
    ↪ freq_ident,vsnominal]
   freq_fdd=[2.13,1;2.89,1;3.02,2]; %[freq_ident,vsnominal]
   Nantes=3;

   nvals='';
```

```
valminimo=10^-14;

a=[R81A0(1:length(t2)),RACCE(1:length(t2)),R4E16(1:length(t2)),RD26F(1:length(t2))];
[F_enc_fdd,Fi_enc_fdd,F,vs,vectores,Psd]= fdd5(a,Fs,WINDOW,NOVERLAP,freq_vector,sigrafica,
  ↪ freq_fdd,Nantes,timebreak,nvals,0,valminimo);

end

%carpeta='2100-2130';

%OK = guarda_figuras(carpeta);
%OK= convierte_figuras_fig_a_jpg_o_tiff_o_svg(carpeta,1,0,1);

abc= dir('*.txt'); %toma el nombre de los archivos que se analizan

tstr=abc.name; %toma el nombre de uno de esos archivos

tsrt2=tstr(9:22); %toma la fecha en formato string

tstr3=[str2num(tsrt2(1:4)), str2num(tsrt2(7:8)), str2num(tsrt2(5:6)), str2num(tsrt2(
9:10)), str2num(tsrt2(11:12)), str2num(tsrt2(13:14))]; %ordena la fecha y la pasa a numeros

t=datenum(tstr3) %coloca la fecha como un numero

disp(F_enc_fdd) %muestra las frecuencias principales

miarchivo=fopen('archivo1.txt','w'); %abre el archivo txt

Escritura=[t, F_enc_fdd(1), F_enc_fdd(2), F_enc_fdd(3)];

fprintf(miarchivo, '%i %i %i %i \n', Escritura') %escribe la fecha y los puntos singulares

fclose(miarchivo)
```

A.4. Extracción de datos a través de Obspy

Código al interior de la Raspberry Pi 3 programado en python, que permite a través de Obspy, extraer los datos de los sensores de una determinada ventana de tiempo y pasarlos a un archivo .txt

```
from obspy.core import UTCDateTime
from obspy.clients.earthworm import Client
import time

def archiva4(ip,est,t,dt1,dt2):
        host=ip
        client=Client(host,16032)
        while True:
                st=client.get_waveforms('AM',est,'00','EHZ',dt1,dt2)
                r=st[0].data # EHZ
                nst=st[0].stats.npts
                dura=int((dt2-dt1)*100)
                if (nst>=dura-1):
                        break
```

```
15
16      tt=t.datetime.strftime(" %Y %m %d %H %M %S")
17      s1=est+'utc'+tt+'g.txt'
18      f=open(s1,'w')
19      for i in range(len(r)):
20          s=('{0:20.4f} \n'.format(r[i]))
21          f.write(s)
22      f.close()
23      print('numero de puntos ',len(r))
24# main
25
26ip1='192.168.0.189'
27est1='RD26F'   # // QDR 4P
28ip2='192.168.0.191'
29est2='R5642'
30ip3='192.168.0.110'
31est3='RACCE'
32ip4='192.168.0.169'
33est4='R4E16'
34ip5='192.168.0.192'
35est5='R81A0'
36
37ip6='192.168.0.104'
38est6='R3323'  # // QDR Pozo
39
40pre=60*30
41pos=0
42
43#dt=UTCDateTime("2021-10-22 01:00:00") # hora UTC de micro
44#print(dt)
45dt=UTCDateTime() # hora UTC de micro
46print(dt)
47
48dt1=dt-pre
49dt2=dt+pos
50
51print('Retirando de ',est1)
52archiva4(ip1,est1,dt,dt1,dt2)
53print('Retirando de ',est2)
54archiva4(ip2,est2,dt,dt1,dt2)
55print('Retirando de ',est3)
56archiva4(ip3,est3,dt,dt1,dt2)
57print('Retirando de ',est4)
58archiva4(ip4,est4,dt,dt1,dt2)
59print('Retirando de ',est5)
60archiva4(ip5,est5,dt,dt1,dt2)
61print('Retirando de ',est6)
62archiva4(ip6,est6,dt,dt1,dt2)
```

A.5. mipsd

Código programado en matlab que permite obtener el espectro de potencia a partir de los datos registrados de la estructura, ocupando como función principal la transformada de fourier discreta.

```
1
2 function [Pxx,freq_vector]=mipsd(x,Fs,freq_vector,window,noverlap)
3 %[Pxx,freq_vector]=mipsd(x,Fs,freq_vector,window,noverlap)
4
5 x2=columna(x);
6
7 Nc=size(x2,2);
8
9 if nargin < 5 || isempty(noverlap)  % no hay overlap
10    noverlap=floor(length(window)/2);
11 end
12
13 if nargin < 3
14    error('No se introduce un vector de frecuencia');  % no se ingresa un vector de frecuencia.
15 end
16
17 clear x
18
19 for j=1:Nc
20    x=x2(:,j);
21
22    n = length(x);       % Number of data points
23    nwind = length(window);
24    if n < nwind     % zero-pad x (and y) if length less than the window length
25       x(nwind)=0;
26       n=nwind;
27
28    end
29    x = x(:);      % Make sure x and y are column vectors
30
31
32    k = fix((n-noverlap)/(nwind-noverlap));   % Number of windows
33    index = 1:nwind;
34
35
36    Pxx2 = zeros(length(freq_vector),1);  % Dual sequence case.
37
38    for i=1:k
39       xw = window.*detrend(x(index),0);
40       Xx=columna(transformada_de_fourier_discreta(xw,freq_vector,Fs));
41       Xx2 = abs(Xx).^2;
42       Pxx2 = Pxx2 + Xx2;
43       index = index + (nwind - noverlap);
44    end
45
46    Pxx(:,j)=Pxx2;
47 end
48
49
50 % function codigo=spcvent(index,posbrk)
51 % %    codigo=spcvent(index,posbrk);
52 % % Identifica si la posicion del index toca alguno de los puntos dentro de la ventanas prohibidas.
53 % % codigo=0  HAY INTERSECCION DE LA VENTANA
54 % % codigo=1  NO HAY INTERESECCION
55 % % ps, rbk 06-11-02
56 %
```

```
% inicio=index(1);
% fin=index(end);
% [nbreak,~]=size(posbrk);
%
% codigo=1; %% no ha interseccion
%
% for i=1:nbreak
%     posini=posbrk(i,1);
%     posfin=posbrk(i,2);
%
%      % NEW agrega condicion faltante
%     if ((inicio>=posini)&&(inicio<=posfin)) || ((fin>=posini)&&(fin<=posfin))|| ((fin>=posfin)&&(
    ↪ inicio<=posini) )
%         codigo=0; % [a,b] parte dentro de ventana
%     end
%
% end


function X=transformada_de_fourier_discreta(x,f,Fs)
% determina la transformada de Fourier discreta de la señal x, para los
% valores de frecuenccia especificados en el vector f.
% fhp 09/2007

f1 = f(1);
f2 = f(end);
m = length(f);
w = exp(-1i*2*pi*(f2-f1)/(m*Fs));
a = exp(1i*2*pi*f1/Fs);
X = czt(x,m,w,a);

% recordar sacar valor absoluto afuera.
```

A.6. columna

Función programada en matlab, necesaria para el código de la sección 8.1.5, que transforma los vectores en columnas.

```

function c=columna(a)
%       function c=columna(a)
% coloca el vector como columna la direccion mayor.
% 6-11-01 rbk modifica para numero complejo no da conjugado

[m,n]=size(a);

if m < n
c=a';
else
c=a;
end


return
```

A.7. czt

Código programado en Matlab, necesario para el código planteado en la sección 8.1.5, que consiste en una generalización de la transformada de Fourier, pero a través de un algoritmo más eficiente que las otras funciones de Matlab que realizan este proceso.

```
1
2## Copyright (C) 2004 Daniel Gunyan
3##
4## This program is free software: you can redistribute it and/or modify
5## it under the terms of the GNU General Public License as published by
6## the Free Software Foundation, either version 3 of the License, or
7## (at your option) any later version.
8##
9## This program is distributed in the hope that it will be useful,
10## but WITHOUT ANY WARRANTY; without even the implied warranty of
11## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
12## GNU General Public License for more details.
13##
14## You should have received a copy of the GNU General Public License
15## along with this program; see the file COPYING. If not, see
16## <https://www.gnu.org/licenses/>.
17
18## -*- texinfo -*-
19## @deftypefn  {Function File} {} czt (@var{x})
20## @deftypefnx {Function File} {} czt (@var{x}, @var{m})
21## @deftypefnx {Function File} {} czt (@var{x}, @var{m}, @var{w})
22## @deftypefnx {Function File} {} czt (@var{x}, @var{m}, @var{w}, @var{a})
23## Chirp z-transform.  Compute the frequency response starting at a and
24## stepping by w for m steps.  a is a point in the complex plane, and
25## w is the ratio between points in each step (i.e., radius increases
26## exponentially, and angle increases linearly).
27##
28## To evaluate the frequency response for the range f1 to f2 in a signal
29## with sampling frequency Fs, use the following:
30##
31## @example
32## @group
33## m = 32;                              ## number of points desired
34## w = exp(-j*2*pi*(f2-f1)/((m-1)*Fs));  ## freq. step of f2-f1/m
35## a = exp(j*2*pi*f1/Fs);                ## starting at frequency f1
36## y = czt(x, m, w, a);
37## @end group
38## @end example
39##
40## If you don't specify them, then the parameters default to a Fourier
41## transform:
42##     m=length(x), w=exp(-j*2*pi/m), a=1
43##
44## If x is a matrix, the transform will be performed column-by-column.
45## @end deftypefn
46
47## Algorithm (based on Oppenheim and Schafer, "Discrete-Time Signal
48## Processing", pp. 623-628):
49##   make chirp of length -N+1 to max(N-1,M-1)
50##     chirp => w^([-N+1:max(N-1,M-1)]^2/2)
```

```
##   multiply x by chirped a and by N−elements of chirp, and call it g
##   convolve g with inverse chirp, and call it gg
##     pad ffts so that multiplication works
##     ifft(fft(g)∗fft(1/chirp))
##   multiply gg by M−elements of chirp and call it done

function y = czt(x, m, w, a)

 if nargin < 1 || nargin > 4, print_usage; endif

 [row, col] = size(x);
 if row == 1, x = x(:); col = 1; endif

 if nargin < 2 || isempty(m), m = length(x(:,1)); endif
 if length(m) > 1, error("czt: m must be a single element\n"); endif
 if nargin < 3 || isempty(w), w = exp(−2∗j∗pi/m); endif
 if nargin < 4 || isempty(a), a = 1; endif
 if length(w) > 1, error("czt: w must be a single element\n"); endif
 if length(a) > 1, error("czt: a must be a single element\n"); endif

 ## indexing to make the statements a little more compact
 n = length(x(:,1));
 N = [0:n−1]'+n;
 NM = [−(n−1):(m−1)]'+n;
 M = [0:m−1]'+n;

 nfft = 2^nextpow2(n+m−1); # fft pad
 W2 = w.^(([−(n−1):max(m−1,n−1)]'.^2)/2); # chirp

 for idx = 1:col
   fg = fft(x(:,idx).∗(a.^−(N−n)).∗W2(N), nfft);
   fw = fft(1./W2(NM), nfft);
   gg = ifft(fg.∗fw, nfft);

   y(:,idx) = gg(M).∗W2(M);
 endfor

 if row == 1, y = y.'; endif

endfunction

%!shared x
%! x = [1,2,4,1,2,3,5,2,3,5,6,7,8,4,3,6,3,2,5,1];
%!assert(fft(x),czt(x),10000∗eps);
%!assert(fft(x'),czt(x'),10000∗eps);
%!assert(fft([x',x']),czt([x',x']),10000∗eps);
```

A.8. FDD

Código programado en Matlab que permite realizar el análisis expuesto en la sección 5.3.1.1. y así obtener las frecuencias fundamentales de la estructura en base a los datos registrados de aceleración por los sensores.

```
1
```

```
function [F_enc_fdd,Fi_enc_fdd,F,vs,vectores,Psd]= fdd5(Ac,Fs,window,NOVERLAP,F,graph,freq_fdd,
    ↪ Nantes,timebreak,nvals,eng,valminimo)
% fdd 2009 fhp pero utilizando la transformada discreta de Fourier
% graph: 0 no se despega graficos
%        1 se depliega graficos entre
%
% Aplicacion
%
%     WINDOW=hann(30*Fs);
%     NOVERLAP='';
%     freq_vector=0:0.01:20;
%     graph_fdd=1;
%     [F,vs,vectores,Psd]= fdd3(a,Fs,WINDOW,NOVERLAP,freq_vector,graph_fdd);
%
%
% ejemplo de procesamiento posterior
%
% freq_fdd=[2.18,1;2.97,1;3.17,1;4.9,1;6.01,1;7.37,1;8.83,1;9.49,1;13.14,1;14.16,1];  %[freq_ident,vsnominal]
%
%  Nantes=3;
%     for i=1:size(freq_fdd,1)
%         [~,auxM1]=min(abs(F-freq_fdd(i,1)));
%         Flocal=F(auxM1-Nantes:auxM1+Nantes);
%         vslocal=vs(auxM1-Nantes:auxM1+Nantes,freq_fdd(i,2));
%         [~,auxM2]=max(vslocal);
%         F_enc_fdd(i)=Flocal(auxM2);
%         Fi_enc_fdd(:,i)=real(vectores(:,freq_fdd(i,2),auxM1-Nantes+auxM2-1));
%     end

% Brinker et al 2001 "Modal identification of output-only systems using
% frequency domain decomposition"

Ac=columna(Ac);

if nargin<12 || isempty(valminimo)
   valminimo=10^-8;
end

if nargin<11 || isempty(eng)
   eng=0;
end

if nargin<3 || isempty(window)
   window=2^13;
end

if length(window)==1
   window=hanning(window);
end

if nargin<4 || isempty(NOVERLAP)
   NOVERLAP=length(window)/2;
end

if nargin<6 || isempty(graph)
   graph=0;
```

```
57 end
58
59 if nargin<8 || isempty(Nantes)
60     Nantes=3;
61 end
62
63 LF=length(F);
64 nc=size(Ac,2);
65 Ntotal=size(Ac,1); % largo total de los datos
66
67 P=zeros(nc,nc,LF);
68
69 if nargin>9 && ~isempty(nvals)
70     vs=zeros(LF,nvals);
71 else
72     vs=zeros(LF,nc);
73 end
74
75
76 if nargin>9 && ~isempty(nvals)
77     if nargout>2
78         vectores=zeros(nc,nvals,LF);
79     end
80 else
81     if nargout>2
82         vectores=zeros(nc,nc,LF);
83     end
84 end
85
86 if nargin<9 || isempty(timebreak)
87     posicionbreak=0;
88 else
89     posicionbreak(:,1)=floor(timebreak(:,1)*Fs);
90     posicionbreak(:,2)=ceil(timebreak(:,2)*Fs);
91 end
92 window = window(:);
93 Tvent=length(window)/Fs;
94 Nventana=Tvent*Fs;
95 Psd=zeros(LF,nc);
96
97
98 % se verifican las ventanas de datos a considerar
99 if nargin<5 || isempty(timebreak)
100     posvent(1,1)=1;
101     posvent(1,2)=Ntotal;
102 else
103     for i=1:size(posicionbreak,1)+1
104         if i==1
105             posvent(i,1)=1;
106         else
107             posvent(i,1)=posicionbreak(i-1,2);
108         end
109
110         if i~=size(posicionbreak,1)+1
111             posvent(i,2)=posicionbreak(i,1);
112         else
```

```
            posvent(i,2)=Ntotal;
        end
    end
end


cuentaV=0;
venti_i=1;


for i=1:size(posvent,1)
    Nventi=floor((posvent(i,2)-posvent(i,1)-NOVERLAP)/(Nventana-NOVERLAP));

    if Nventi>0
        cuentaV=cuentaV+Nventi;

        for j=1:Nventi
            N1(venti_i)=posvent(i,1)+(j-1)*Nventana-NOVERLAP*(j-1);
            N2(venti_i)=posvent(i,1)+j*Nventana-1-NOVERLAP*(j-1);
            venti_i=venti_i+1;
        end
    end
end

disp(['FDD: NUMERO DE VENTANAS A CORRER ES: ',num2str(cuentaV)])




Ac=columna(Ac);
nx = size(Ac,1);
% nwind = length(window);
% ncol = fix((nx-NOVERLAP)/(nwind-NOVERLAP)); % numero de espectros de potencia a calcular
% colindex = floor(nwind/2) + (0:(ncol-1))*(nwind-NOVERLAP);

% if size(Ac,1)<(nwind+colindex(ncol)-1) % se rellena con ceros
%     Ac(nwind+colindex(ncol)-1,nc) = 0;
% end

%X=transformada_de_fourier_discreta(x,f,Fs);

for i=1:nc
    Pxx=zeros(LF,1);
    for nv =1:cuentaV
        x=Ac(N1(nv):N2(nv),i).*window(:);
        X=transformada_de_fourier_discreta(x,F,Fs);
        Pxx=Pxx(:)+conj(X(:)).*X(:);
    end
    P(i,i,:)=Pxx*1/cuentaV;
    Psd(:,i)=Pxx*1/cuentaV;
    for j=i+1:nc
        Pxy=zeros(LF,1);
        for nv =1:cuentaV
            x=Ac(N1(nv):N2(nv),i).*window(:);
```

```
169             y=Ac(N1(nv):N2(nv),j).*window(:);
170             X=transformada_de_fourier_discreta(x,F,Fs);
171             Y=transformada_de_fourier_discreta(y,F,Fs);
172             Pxy=Pxy(:)+conj(X(:)).*Y(:);
173         end
174         P(i,j,:)=Pxy*1/cuentaV;
175         P(j,i,:)=conj(Pxy)*1/cuentaV;
176     end
177 end
178 
179 
180 for i=1:length(F)
181     % Notar que se hace esta funcion con valores propios.
182 
183     if nargin>9 && ~isempty(nvals)
184         if nargout>2
185             [U,S] = eigs(P(:,:,i),nvals);
186             S=diag(S);
187         else
188             [S] = eigs(P(:,:,i),nvals);
189         end
190     else
191         if nargout>2
192             [U,S] = eig(P(:,:,i),'balance');
193             S=diag(S);
194         else
195             [S] = eig(P(:,:,i),'balance');
196         end
197     end
198 
199     [aux,orden]=sort(S,'descend');
200     aux2=S;
201     vs(i,:)=aux2(orden);
202 
203     if nargout>2
204         vectores(:,:,i)=U(:,orden);  %GDL x vs x F
205     end
206 
207     %     [U1,S1,V1] = svd(P(:,:,i));
208     %     [vs1(i,:),orden]=sort(diag(S1),'descend');
209     %     vectores(:,:,i)=U(orden,:);
210     %     vectores2(:,:,i)=V(orden,:);
211 
212 end
213 
214 
215 for i=1:size(vs,1)
216     for j=1:size(vs,2)
217         if vs(i,j)<valminimo;
218             vs(i,j)=valminimo;
219         end
220     end
221 end
222 
223 if nargin >6 && ~isempty(freq_fdd)
224     %         Nantes=1;
```

```
225   for i=1:size(freq_fdd,1)
226       [~,auxM1]=min(abs(F-freq_fdd(i,1)));
227       Flocal=F(auxM1-Nantes:auxM1+Nantes);
228       vslocal=vs(auxM1-Nantes:auxM1+Nantes,freq_fdd(i,2));
229       [~,auxM2]=max(vslocal);
230       F_enc_fdd(i)=Flocal(auxM2);
231       vs_valor_local(i)=vslocal(auxM2);
232       Fi_enc_fdd(:,i)=real(vectores(:,freq_fdd(i,2),auxM1-Nantes+auxM2-1));
233   end
234 else
235   Fi_enc_fdd=[];
236   F_enc_fdd=[];
237
238 end
239
240
241
242 if graph
243
244   %     plot(F,vs(:,1),F,vs1(:,1))
245   %     xlim([0,50])
246   %     grid on
247   %     title('Valores Singulares')
248   %     xlabel('Frecuencia [Hz]')
249   %     ylabel('Valores Singulares')
250
251
252   fgraf=[F(1),F(end)];
253   figure
254
255   plot(F,vs)
256   xlim(fgraf)
257   grid on
258   if ~eng
259       title('Valores Singulares')
260       xlabel('Frecuencia [Hz]')
261       ylabel('Valores Singulares')
262   else
263       %             title('Singular Values')
264       xlabel('Frequency [Hz]')
265       ylabel('Singular Values')
266   end
267
268   hold on
269   if ~isempty(F_enc_fdd)
270       stem(F_enc_fdd,vs_valor_local)
271   end
272
273   figure
274   semilogy(F,vs)
275   grid on
276   xlim(fgraf)
277   if ~eng
278       title('Valores Singulares')
279       xlabel('Frecuencia [Hz]')
280       ylabel('Valores Singulares (log)')
```

```
281 else
282     xlabel('Frequency [Hz]')
283     ylabel('Singular Values (log)')
284 end
285
286 hold on
287 if ~isempty(F_enc_fdd)
288     stem(F_enc_fdd,vs_valor_local)
289 end
290
291 %     figure
292 %
293 %     plot(F,20*log10(vs))
294 %     grid on
295 %     xlim(fgraf)
296 %
297 %     title('Valores Singulares')
298 %     xlabel('Frecuencia [Hz]')
299 %     ylabel('Valores Singulares (dB)')
300 %     hold on
301 %     stem(F_enc_fdd,20*log10(vs_valor_local))
302
303 figure
304
305 plot(F,Psd)
306 grid on
307 xlim(fgraf)
308 if ~eng
309     title('Espectro de Potencia')
310     xlabel('Frecuencia [Hz]')
311     ylabel('PSD')
312 else
313     xlabel('Frequency [Hz]')
314     ylabel('PSD')
315 end
316
317 end
318
319 disp('FDD Terminado')
```

A.9. hann

Código programado en Matlab, necesario para el código expuesto en 8.1.8. que permite la realización de ventanas siméstricas de hann, utilizadas para evitar discontinuidades que puedan producirse al principio y al final de los bloques analizados.

```
1
2 ## Copyright (C) 2014-2019 Mike Miller
3 ##
4 ## This program is free software: you can redistribute it and/or modify
5 ## it under the terms of the GNU General Public License as published by
6 ## the Free Software Foundation, either version 3 of the License, or
7 ## (at your option) any later version.
8 ##
```

```
## This program is distributed in the hope that it will be useful,
## but WITHOUT ANY WARRANTY; without even the implied warranty of
## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
## GNU General Public License for more details.
##
## You should have received a copy of the GNU General Public License
## along with this program; see the file COPYING. If not, see
## <https://www.gnu.org/licenses/>.

## -*- texinfo -*-
## @deftypefn  {Function File} {} hann (@var{m})
## @deftypefnx {Function File} {} hann (@var{m}, "periodic")
## @deftypefnx {Function File} {} hann (@var{m}, "symmetric")
## Return the filter coefficients of a Hanning window of length @var{m}.
##
## If the optional argument @code{"periodic"} is given, the periodic form
## of the window is returned.  This is equivalent to the window of length
## @var{m}+1 with the last coefficient removed.  The optional argument
## @code{"symmetric"} is equivalent to not specifying a second argument.
##
## This function exists for @sc{matlab} compatibility only, and is equivalent
## to @code{hanning (@var{m})}.
##
## @seealso{hanning}
## @end deftypefn

function w = hann (varargin)

 if (nargin < 1 || nargin > 2)
   print_usage ();
 endif

 w = hanning (varargin{:});

endfunction

%!assert (hann (1), 1);
%!assert (hann (2), zeros (2, 1));
%!assert (hann (16), flipud (hann (16)), 10*eps);
%!assert (hann (15), flipud (hann (15)), 10*eps);
%!test
%! N = 15;
%! A = hann (N);
%! assert (A(ceil (N/2)), 1);

%!assert (hann (15), hann (15, "symmetric"));
%!assert (hann (16)(1:15), hann (15, "periodic"));
%!test
%! N = 16;
%! A = hann (N, "periodic");
%! assert (A (N/2 + 1), 1);

%% Test input validation
%!error hann ()
%!error hann (0.5)
%!error hann (-1)
```

```
%!error hann (1, "invalid")
```

A.10. Transformada de Fourier discreta

Código programado en Matlab que como su nombre lo dice, devuelve la transformada de fourier de la señal, pero utilizando el código expuesto en 8.1.7. para mejorar su eficiencia con respecto a las funciones ya existentes en matlab.

```
function X=transformada_de_fourier_discreta(x,f,Fs)
% determina la transformada de Fourier discreta de la señal x, para los
% valores de frecuenccia especificados en el vector f.
% fhp 09/2007

f1 = f(1);
f2 = f(end);
m = length(f);
w = exp(-j*2*pi*(f2-f1)/(m*Fs));
a = exp(j*2*pi*f1/Fs);
X = czt(x,m,w,a)*(1/(Fs*sqrt(2*pi)));
```

A.11. shmpen

Código programado en Octave que se utiliza en la Raspberry Pi 3 para analizar los datos de los últimos 30 minutos, cada una hora y que registra finalmente los resultados del análisis FDD y las condiciones ambientales registradas en los sensores en un archivo .txt

```
file=dir('*g.txt');
s1=file(1).name;
i=regexp(s1,'g');
fecha=s1(9:i-1);

[t,u] =system("python3 pentem2.py");
temp=str2num(u);

tsrt2=fecha; %toma la fecha en formato string

tstr3=[str2num(tsrt2(1:4)), str2num(tsrt2(7:8)), str2num(tsrt2(5:6)), str2num(tsrt2(
9:10)), str2num(tsrt2(11:12)), str2num(tsrt2(13:14))]; %ordena la fecha y la pasa a numeros

tactual=datenum(tstr3) %coloca la fecha como un numero

estaciones=['RD26F';'R5642';'RACCE';'R81A0';'R4E16';'R3323'];
s1=strcat(estaciones(1,:),'utc',fecha,'g.txt');
r1=load(s1);
s1=strcat(estaciones(2,:),'utc',fecha,'g.txt');
r2=load(s1);
s1=strcat(estaciones(3,:),'utc',fecha,'g.txt');
r3=load(s1);
s1=strcat(estaciones(4,:),'utc',fecha,'g.txt');
r4=load(s1);
```

```
s1=strcat(estaciones(5,:),'utc',fecha,'g.txt');
r5=load(s1);
s1=strcat(estaciones(6,:),'utc',fecha,'g.txt');
r6=load(s1);

npt=min([length(r1) length(r2) length(r3) length(r4) length(r5) length(r6)]);
A=[r1(1:npt) r2(1:npt) r3(1:npt) r4(1:npt) r5(1:npt) r6(1:npt)];

%npt=min([length(r2) length(r3) length(r4) length(r5) length(r6)]);
%A=[r2(1:npt) r3(1:npt) r4(1:npt) r5(1:npt) r6(1:npt)];

estaciones=['RD26F','R5642';'RACCE';'R81A0';'R4E16';'R3323']; % mientras RD26F caido

sieliminatxt=1;

if sieliminatxt
 for i=1:length(file)
   delete(file(i).name);
 endfor;
endif;

%  gruarda matriz A con geofonos en cuentas y del mismo largo en /home/pi/data/
Fs=100;
#clear file i r1 r2 r3 s1 r4 r5 r6
#save('-v7',strcat('/home/pi/data/pm',fecha,'.mat'))

sigrafica=0;
if sigrafica
% grafica matriz A
graficosLib
plot(A)
title(fecha)
legend('RD26F','R5642','RACCE','R81A0','R4E16','R3323');

% gruarda grafico matriz A en png en /home/pi/data/fig/
temp=['/home/pi/data/fig/pm' fecha 'g.png'];
print(temp,'-dpng');
end

close all
% AQUI deberia venir PSD, SSI o otro
% bye

si_fdd=1;
sigrafica=0;

mi_freq_fdd0=[2.03,2.78,2.95]; %[freq_ident,vsnominal]

if exist('resumen_micro.txt','file')
   resu1=load('resumen_micro.txt','-ascii');
   fultima=resu1(end,7:9);
   dif_f=abs(fultima-mi_freq_fdd0)./mi_freq_fdd0;
   for i=1:3
       if dif_f(i)<0.3
           mi_freq_fdd0(i)=fultima(i);
       end
```

```
82    end
83end
84
85if si_fdd
86    %plotdef(1.5,1.5)
87    timebreak=[];
88    WINDOW=hann(20*Fs);
89    NOVERLAP='';
90    freq_vector=0:0.01:20;
91    graph_fdd=1;
92%       [F,vs,vectores,Psd]= fdd3(a,Fs,WINDOW,NOVERLAP,freq_vector,graph_fdd);
93%       freq_fdd=[2.18,1;2.97,1;3.15,1;4.86,1;6.03,1;7.37,1;9.50,1;8.88,1;10.5,1;13.19,1;15.22,1;16.33,1];  %[
      ↪ freq_ident,vsnominal]
94    freq_fdd=[mi_freq_fdd0(1),1;mi_freq_fdd0(2),1;mi_freq_fdd0(3),2];  %[freq_ident,vsnominal]
95    Nantes=10;
96
97    nvals='';
98    valminimo=10^-14;
99
100   a=A(:,1:(end-1));
101   [F_enc_fdd,Fi_enc_fdd,F,vs,vectores,Psd]= fdd5(a,Fs,WINDOW,NOVERLAP,freq_vector,sigrafica,
      ↪ freq_fdd,Nantes,timebreak,nvals,0,valminimo);
102
103end
104
105
106
107disp(tactual)
108disp(F_enc_fdd)
109
110resu=[str2num(tsrt2(1:4)), str2num(tsrt2(7:8)), str2num(tsrt2(5:6)), str2num(tsrt2(
1119:10)), str2num(tsrt2(11:12)), str2num(tsrt2(13:14)),F_enc_fdd, temp(1), temp(2)];
112
113disp(resu)
114
115if ~exist('resumen_micro.txt','file')
116  save('resumen_micro.txt','resu','-ascii')
117else
118  resu1=load('resumen_micro.txt','-ascii');
119  resu=[resu1;resu];
120  delete('resumen_micro.txt')
121  save('resumen_micro.txt','resu','-ascii')
122end
123
124%carpeta='2100-2130';
125
126%OK = guarda_figuras(carpeta);
127%OK= convierte_figuras_fig_a_jpg_o_tiff_o_svg(carpeta,1,0,1);
128
129% test
130
131e1=fopen('resultadostest.txt','a');
132sf=num2str(F_enc_fdd,"%7.3f");
133u1=num2str(temp,"%5.1f");
134temporal=sprintf("%s  %s  %s\r\n",fecha,sf,u1);
135fputs(e1,temporal);
```

```
fclose(e1);

% end test

quit
```

A.12. graficosLib

Código que contiene las librerias para poder graficar los resultados.

```
%graficosLib
graphics_toolkit('fltk')

\
```

A.13. Extracción Condiciones Ambientales

Código programado en python que permite obtener y registrar los datos medioambientales obtenidos por el sensor bme280.

```

import time
import board
from adafruit_bme280 import basic as adafruit_bme280


def temperatura():

# Create sensor object, using the board's default I2C bus.
    i2c = board.I2C()  # uses board.SCL and board.SDA
    bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c,address=0x76)
    t=bme280.temperature
    h=bme280.relative_humidity

    return t,h

t,h=temperatura()
print(t,' ',h)
```

A.14. Análisis de variación de frecuencias ante condiciones ambientales

Código programado en Matlab que permite realizar todo el análisis estadístico expuesto en la sección 5.3.4. a partir de los datos registrados con el código expuesto en 8.1.11.

```

close all
clc
clearvars
```

```
plotdef(1.5,1.5)


Tm=20;
Hrm=47.8;
nbins=20;
orden1=18;
orden2=18;
orden3=18;

diainicial='29-11-2021';
diafinal='29-12-2021';
% diainicial='13-12-2021';
% diafinal='23-12-2021';


ddiainicial=datenum(diainicial,'dd-mm-yyyy');
ddiafinal=datenum(diafinal,'dd-mm-yyyy');

difdia1=datevec(ddiafinal-ddiainicial);
difdia=difdia1(3);

misdias(1,:)=datevec(ddiainicial);
xauxlabel{1}=datestr(datevec(ddiainicial),'dd/mm');

for i=1:difdia
    misdias(i+1,:)=datevec(ddiainicial)+[0,0,i,0,0,0];
    xauxlabel{i+1}=datestr(datevec(ddiainicial)+[0,0,i,0,0,0],'dd/mm');
end



datos=load('resumen_micro.txt');

ano=datos(:,1);
dia=datos(:,2);
mes=datos(:,3);
hora=datos(:,4);
minuto=datos(:,5);
segundo=datos(:,6);
f1=datos(:,7);
f2=datos(:,8);
f3=datos(:,9);
temp=datos(:,10);
hum=datos(:,11);

DN = datenum(ano,mes,dia,hora,minuto,segundo);


figure(44)

[AXe,H1,~]=plotyy(DN,temp,DN,hum);

% set(get(AXe(1),'children'),'LineWidth',2.5,'Color','b','LineStyle','-','LineWidth',1.5,'Marker','o')
% set(get(AXe(2),'children'),'LineWidth',2.5,'Color','g','LineStyle','-','LineWidth',1.5,'Marker','o')
```

```
61
62 set(get(AXe(1),'children'),'LineWidth',2.5,'LineStyle','−','LineWidth',1.5,'Marker','o')
63 set(get(AXe(2),'children'),'LineWidth',2.5,'LineStyle','−','LineWidth',1.5,'Marker','o')
64
65
66 set(AXe(1),'xlim',[ddiainicial,ddiafinal],'ylim',[10,35],'Ytick',[10:5:35],'fontsize',7) %%%% cambiar limites
67 set(AXe(2),'xlim',[ddiainicial,ddiafinal],'ylim',[20,70],'Ytick',[20:10:70],'fontsize',7)
68 %
69 set(get(AXe(1),'Ylabel'),'String','T (°C)','fontsize',7);
70 set(get(AXe(2),'Ylabel'),'String','Hr ( %)','fontsize',7);
71
72 set(gca,'Xtick',datenum(misdias),'Xticklabel',xauxlabel,'XTickLabelRotation',90,'Fontsize',9)
73 grid on
74
75
76
77
78 figure(1)
79
80 subplot(4,1,1)
81
82 [AXe,H1,~]=plotyy(DN,temp,DN,hum);
83
84 % set(get(AXe(1),'children'),'LineWidth',2.5,'Color','b','LineStyle','−','LineWidth',1.5,'Marker','o')
85 % set(get(AXe(2),'children'),'LineWidth',2.5,'Color','g','LineStyle','−','LineWidth',1.5,'Marker','o')
86
87 set(get(AXe(1),'children'),'LineWidth',2.5,'LineStyle','−','LineWidth',1.5,'Marker','o')
88 set(get(AXe(2),'children'),'LineWidth',2.5,'LineStyle','−','LineWidth',1.5,'Marker','o')
89
90
91 set(AXe(1),'xlim',[ddiainicial,ddiafinal],'ylim',[10,35],'Ytick',[10:5:35],'fontsize',7) %%%% cambiar limites
92 set(AXe(2),'xlim',[ddiainicial,ddiafinal],'ylim',[20,70],'Ytick',[20:10:70],'fontsize',7)
93 %
94 set(get(AXe(1),'Ylabel'),'String','T (°C)','fontsize',7);
95 set(get(AXe(2),'Ylabel'),'String','Hr ( %)','fontsize',7);
96
97 set(gca,'Xtick',datenum(misdias),'Xticklabel',xauxlabel,'XTickLabelRotation',90,'Fontsize',9)
98 grid on
99
100
101
102 subplot(4,1,2)
103 plot(DN,f1,'k−o')
104 grid on
105 xlim([ddiainicial,ddiafinal])
106 set(gca,'Xtick',datenum(misdias),'Xticklabel',xauxlabel,'XTickLabelRotation',90,'Fontsize',9)
107 % datetick('x','dd−mmm−yyyy')
108 ylabel('F_1 (Hz)')
109
110 subplot(4,1,3)
111 plot(DN,f2,'k−o')
112 grid on
113 xlim([ddiainicial,ddiafinal])
114 set(gca,'Xtick',datenum(misdias),'Xticklabel',xauxlabel,'XTickLabelRotation',90,'Fontsize',9)
115 ylabel('F_2 (Hz)')
116
```

```
subplot(4,1,4)
plot(DN,f3,'k-o')
grid on
% datetick('x','dd-mmm-yyyy')
xlim([ddiainicial,ddiafinal])
set(gca,'Xtick',datenum(misdias),'Xticklabel',xauxlabel,'XTickLabelRotation',90,'Fontsize',9)
ylabel('F_3 (Hz)')



% modelo Espacio - Estado o ARX

miDN=diff(DN);
auxExp=find(miDN>miDN(1)*3);
Nexp=length(auxExp)+1;


ctex=[];
for i=1:Nexp
    if i==1
        ctex=[ctex,'data(1:',num2str(auxExp(1),'%1.0f'),')'];
        midata{1}=['data(1:',num2str(auxExp(1),'%1.0f'),')'];
        tomados{1}=['(1:',num2str(auxExp(1),'%1.0f'),')'];
    elseif i<Nexp
        ctex=[ctex,',data(',num2str(auxExp(i-1)+1,'%1.0f'),':',num2str(auxExp(i),'%1.0f'),')'];
        midata{i}=['data(',num2str(auxExp(i-1)+1,'%1.0f'),':',num2str(auxExp(i),'%1.0f'),')'];
        tomados{i}=['(',num2str(auxExp(i-1)+1,'%1.0f'),':',num2str(auxExp(i),'%1.0f'),')'];
    else
        ctex=[ctex,',data(',num2str(auxExp(i-1)+1,'%1.0f'),':',num2str(length(DN),'%1.0f'),')'];
        midata{Nexp}=['data(',num2str(auxExp(i-1)+1,'%1.0f'),':',num2str(length(DN),'%1.0f'),')'];
        tomados{Nexp}=['(',num2str(auxExp(i-1)+1,'%1.0f'),':',num2str(length(DN),'%1.0f'),')'];
    end
end



figure(201)
subplot(4,1,1)
[AXe,H1,~]=plotyy(DN,temp,DN,hum);
set(get(AXe(1),'children'),'LineWidth',2.5,'LineStyle','-','LineWidth',1.5,'Marker','o')
set(get(AXe(2),'children'),'LineWidth',2.5,'LineStyle','-','LineWidth',1.5,'Marker','o')
set(AXe(1),'xlim',[ddiainicial,ddiafinal],'ylim',[10,35],'Ytick',[10:5:35],'fontsize',7)  %%%% cambiar limites
set(AXe(2),'xlim',[ddiainicial,ddiafinal],'ylim',[20,70],'Ytick',[20:10:70],'fontsize',7)
set(get(AXe(1),'Ylabel'),'String','T (°C)','fontsize',7);
set(get(AXe(2),'Ylabel'),'String','Hr ( %)','fontsize',7);
set(gca,'Xtick',datenum(misdias),'Xticklabel',xauxlabel,'XTickLabelRotation',90,'Fontsize',9)
grid on

u1=[temp(:),hum(:)];
fmedia=mean(f1(:))*1;
y=f1(:)-fmedia;
Ts=1; % hr
data = iddata(y,u,Ts);
eval(['dat=merge(',ctex,')'])
opt = arxOptions('InitialCondition','estimate');
% [sys,ic] = arx(z,[2 2 1])  % la version 2018 o 2014 no incluye condicion
```

```
% inicial para modelos ARX
% [sys_tf,ic] = tfest(data,2,1);
% Ni=size(u,2);
% sys2 = arx(data,[orden,orden*ones(1,Ni),zeros(1,Ni)],opt);
sys1=n4sid(dat,orden1);
sys1.Report.Fit
f1arx=[];

subplot(4,1,2)
plot(DN,f1,'k-o')
hold on
for i=1:Nexp
    [yaux,fit,x0] = compare(sys1,eval(midata{i}));
    f1arxaux = yaux.OutputData+fmedia;
    f1arx=[f1arx(:);f1arxaux(:)];
    plot(DN(eval(tomados{i})),f1arxaux(:),'--r')
end
% disp(x0)
% sys = arx(data,[2 2 1]);
% simOpt = simOptions('InitialCondition',ic);
% opt = simOptions('InitialCondition',x0);
% f1arx = sim(sys1,u,opt);
grid on
xlim([ddiainicial,ddiafinal])
set(gca,'Xtick',datenum(misdias),'Xticklabel',xauxlabel,'XTickLabelRotation',90,'Fontsize',9)
ylabel('F_1 (Hz)')
% ylim([1.9,2.3])


% Grafico 3D de comportamiento pseudo-estático

Tlim=[15:1:30];
Hlim=[35:2.5:65];


A=sys1.A;
B=sys1.B;
C=sys1.C;
D=sys1.D;

mimatriz=(C*(inv(eye(orden1)-A)*B)+D);

for i=1:length(Tlim)
    for j=1:length(Hlim)
        f13D(j,i)=mimatriz*[Tlim(i);Hlim(j)]+fmedia;
    end
end

[XT,YH] = meshgrid(Tlim,Hlim);

figure(1001)
subplot(2,3,1)
% miColor1=(f13D-max(max(f13D)))/(max(max(f13D))-min(min(f13D)));
miColor1=f13D;
surf(XT,YH,f13D,miColor1)
grid on
```

```
xlabel('Temperatura (°C)')
ylabel('Humedad Rel. (%)')
zlabel('F_1 (Hz)')
hold on
% miColor=(f1-max(max(f13D)))/(max(max(f13D))-min(min(f13D)));
miColor=f1;
scatter3(temp,hum,f1,ones(length(temp),1)*20,miColor,'filled')
grid on
colorbar
view([   -36.5000   12.0000])
xlim([15,30])
ylim([35,65])

subplot(2,3,4)

for i=1:length(Hlim)
    plot(XT(i,:),f13D(i,:))
    hold on
    mileg{i}=num2str(YH(i,1),'%1.1f');
end
grid on
xlabel('Temperatura (°C)')
ylabel('F_1 (Hz)')
legend(mileg,'Location','eastoutside')









%

%
% opt = simOptions('InitialCondition','z');
%
% tprueba=0:1:length(tempS)-1;
% uS=[tempS(:),humS(:)];
% f1arxS = sim(sys1,uS,opt)+fmedia;
% figure(1000)
% subplot(3,1,1)
% plot(tprueba,tempS)
% grid on
% subplot(3,1,2)
% plot(tprueba,humS)
% grid on
% subplot(3,1,3)
% plot(tprueba,f1arxS)
% grid on



figure(206)
subplot(2,3,1)
scatter(f1,f1arx,'k')
```

```
grid on
hold on
xL(2)=max(max([f1(:),f1arx(:)]));
xL(1)=min(min([f1(:),f1arx(:)]));
tsstd = std(f1-f1arx);
plot(xL,xL,'r')
plot(xL,xL+1.96*tsstd,'g')
plot(xL,xL-1.96*tsstd,'g')
xlabel('F_1 (Hz)')
ylabel('F_1^{(est)} (Hz)')
R = corrcoef(f1arx(:),f1);
text(0.05,1,{['F_1=SS(T,H_r)'],['\sigma=',num2str(tsstd,' %1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,' %1.3g'
    ↪ )]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
axis equal

mu=mean(f1arx(:)-f1);
subplot(2,3,4)
histogram(f1arx(:)-f1,nbins,'Normalization','pdf')
x1=get(gca,'xlim');
xx=x1(1):(x1(2)-x1(1))/50:x1(2);
hold on
plot(xx,pdf('Normal',xx,mu,tsstd),'r')
xlabel('\epsilon_r')
title('F_1 v/s SS(T,H_r)')
grid on


fmedia=mean(f2(:))*1;
y=f2(:)-fmedia;
Ts=1; % hr
data = iddata(y,u,Ts);
opt = arxOptions('InitialCondition','estimate');
eval(['dat=merge(',ctex,')'])
sys1=n4sid(dat,orden2);
sys1.Report.Fit

figure(201)
subplot(4,1,3)
plot(DN,f2,'k-o')
hold on
f2arx=[];
for i=1:Nexp
    [yaux,fit,x0] = compare(sys1,eval(midata{i}));
    f2arxaux = yaux.OutputData+fmedia;
    f2arx=[f2arx(:);f2arxaux(:)];
    plot(DN(eval(tomados{i})),f2arxaux(:),'--r')
end
grid on
xlim([ddiainicial,ddiafinal])
set(gca,'Xtick',datenum(misdias),'Ticklabel',xauxlabel,'XTickLabelRotation',90,'Fontsize',9)
ylabel('F_2 (Hz)')

% Grafico 3D de comportamiento pseudo-estático

Tlim=[15:1:30];
```

```
Hlim=[35:2.5:65];


A=sys1.A;
B=sys1.B;
C=sys1.C;
D=sys1.D;

mimatriz=(C*(inv(eye(orden1)-A)*B)+D);

for i=1:length(Tlim)
    for j=1:length(Hlim)
        f13D(j,i)=mimatriz*[Tlim(i);Hlim(j)]+fmedia;
    end
end

[XT,YH] = meshgrid(Tlim,Hlim);

figure(1001)
subplot(2,3,2)
miColor1=f13D;
surf(XT,YH,f13D,miColor1)
grid on
xlabel('Temperatura (°C)')
ylabel('Humedad Rel. (%)')
zlabel('F_2 (Hz)')
hold on
miColor=f2;
scatter3(temp,hum,f2,ones(length(temp),1)*20,miColor,'filled')
grid on
colorbar
view([    -36.5000    12.0000])
xlim([15,30])
ylim([35,65])

subplot(2,3,5)

for i=1:length(Hlim)
    plot(XT(i,:),f13D(i,:))
    hold on
    mileg{i}=num2str(YH(i,1),'%1.1f');
end
grid on
xlabel('Temperatura (°C)')
ylabel('F_2 (Hz)')
legend(mileg,'Location','eastoutside')

%%%%%%%%%%%%


figure(206)
subplot(2,3,2)
scatter(f2,f2arx,'k')
grid on
hold on
xL(2)=max(max([f2(:),f2arx(:)]));
```

```
xL(1)=min(min([f2(:),f2arx(:)]));
tsstd = std(f2-f2arx);
plot(xL,xL,'r')
plot(xL,xL+1.96*tsstd,'g')
plot(xL,xL-1.96*tsstd,'g')
xlabel('F_2 (Hz)')
ylabel('F_2^{(est)} (Hz)')
R = corrcoef(f2arx(:),f2);
text(0.05,1,{['F_2=SS(T,H_r)'],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g'
    ↪ )]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
axis equal

mu=mean(f2arx(:)-f2);
subplot(2,3,5)
histogram(f2arx(:)-f2,nbins,'Normalization','pdf')
x1=get(gca,'xlim');
xx=x1(1):(x1(2)-x1(1))/50:x1(2);
hold on
plot(xx,pdf('Normal',xx,mu,tsstd),'r')
xlabel('\epsilon_r')
title('F_2 v/s SS(T,H_r)')
grid on



fmedia=mean(f3(:))*1;
y=f3(:)-fmedia;
Ts=1; % hr
data = iddata(y,u,Ts);
opt = arxOptions('InitialCondition','estimate');
eval(['dat=merge(',ctex,')'])
sys1=n4sid(dat,orden3);
sys1.Report.Fit

figure(201)
subplot(4,1,4)
plot(DN,f3,'k-o')
hold on
f3arx=[];
for i=1:Nexp
    [yaux,fit,x0] = compare(sys1,eval(midata{i}));
    f3arxaux = yaux.OutputData+fmedia;
    f3arx=[f3arx(:);f3arxaux(:)];
    plot(DN(eval(tomados{i})),f3arxaux(:),'--r')
end
grid on
xlim([ddiainicial,ddiafinal])
set(gca,'Xtick',datenum(misdias),'Xticklabel',xauxlabel,'XTickLabelRotation',90,'Fontsize',9)
ylabel('F_3 (Hz)')



% Grafico 3D de comportamiento pseudo-estático

Tlim=[15:1:30];
```

```
Hlim=[35:2.5:65];


A=sys1.A;
B=sys1.B;
C=sys1.C;
D=sys1.D;

mimatriz=(C*(inv(eye(orden1)-A)*B)+D);

for i=1:length(Tlim)
    for j=1:length(Hlim)
        f13D(j,i)=mimatriz*[Tlim(i);Hlim(j)]+fmedia;
    end
end

[XT,YH] = meshgrid(Tlim,Hlim);

figure(1001)
subplot(2,3,3)
miColor1=f13D;
surf(XT,YH,f13D,miColor1)
grid on
xlabel('Temperatura (°C)')
ylabel('Humedad Rel. (%)')
zlabel('F_3 (Hz)')
hold on
miColor=f3;
scatter3(temp,hum,f3,ones(length(temp),1)*20,miColor,'filled')
grid on
colorbar
view([    -36.5000    12.0000])
xlim([15,30])
ylim([35,65])

subplot(2,3,6)

for i=1:length(Hlim)
    plot(XT(i,:),f13D(i,:))
    hold on
    mileg{i}=num2str(YH(i,1),'%1.1f');
end
grid on
xlabel('Temperatura (°C)')
ylabel('F_3 (Hz)')
legend(mileg,'Location','eastoutside')

%%%%%%%%%%%


figure(206)
subplot(2,3,3)
scatter(f3,f3arx,'k')
grid on
hold on
```

```
xL(2)=max(max([f3(:),f3arx(:)]));
xL(1)=min(min([f3(:),f3arx(:)]));
tsstd = std(f3-f3arx);
plot(xL,xL,'r')
plot(xL,xL+1.96*tsstd,'g')
plot(xL,xL-1.96*tsstd,'g')
xlabel('F_3 (Hz)')
ylabel('F_3^{(est)} (Hz)')
R = corrcoef(f3arx(:),f3);
text(0.05,1,{['F_3=SS(T,H_r)'],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g'
    ↪ )]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
axis equal

mu=mean(f3arx(:)-f3);
subplot(2,3,6)
histogram(f3arx(:)-f3,nbins,'Normalization','pdf')
x1=get(gca,'xlim');
xx=x1(1):(x1(2)-x1(1))/50:x1(2);
hold on
plot(xx,pdf('Normal',xx,mu,tsstd),'r')
xlabel('\epsilon_r')
title('F_3 v/s SS(T,H_r)')
grid on


plotdef(1.5,1.5)
Bf=[1/2,1/4,1/4];
Bf=1/4*[1,1,1,1];
% Bf=1;
Af=1;

temp2=temp;
hum2=hum;
f1_2=f1;
f2_2=f2;
f3_2=f3;
DN_2=DN;
temp = filtfilt(Bf,Af,temp);


% Bf=1;
% Af=1;

hum = filtfilt(Bf,Af,hum);


figure(88)
subplot(2,1,1)
plot(DN,temp2,DN,temp)
grid on
xlim([ddiainicial,ddiafinal])
set(gca,'Xtick',datenum(misdias),'Xticklabel',xauxlabel,'XTickLabelRotation',90,'Fontsize',9)
ylabel('Temperatura (°C)')

subplot(2,1,2)
```

```
562 plot(DN,hum2,DN,hum)
563 grid on
564 xlim([ddiainicial,ddiafinal])
565 set(gca,'Xtick',datenum(misdias),'Xticklabel',xauxlabel,'XTickLabelRotation',90,'Fontsize',9)
566 ylabel('Humedad Rel. (%)')
567
568
569
570 sacar=7;
571
572 temp=temp(sacar:end-sacar);
573 hum=hum(sacar:end-sacar);
574 f1=f1(sacar:end-sacar);
575 f2=f2(sacar:end-sacar);
576 f3=f3(sacar:end-sacar);
577 DN=DN(sacar:end-sacar);
578
579 % correlacion temperatura vs humedad
580 Npolinomio_temp_hum=3;
581 figure(100)
582 subplot(2,1,1)
583 scatter(temp2,hum2)
584 grid on
585 xlabel('Temperatura (°C)')
586 ylabel('Humedad Rel. (%)')
587 P = polyfit(temp2-Tm,hum2,Npolinomio_temp_hum);
588 xlinea=[min(temp2):(max(temp2)-min(temp2))/20:max(temp2)];
589 flinea = polyval(P,xlinea-Tm);
590 hold on
591 humprueba=polyval(P,temp2-Tm);
592 tsstd = std(hum2-humprueba);
593 plot(xlinea,flinea,'r')
594 plot(xlinea,flinea-1.96*tsstd,'g')
595 plot(xlinea,flinea+1.96*tsstd,'g')
596 R = corrcoef(humprueba,hum2);
597 esc_pol=[];
598 for i=1:Npolinomio_temp_hum
599     if i==Npolinomio_temp_hum
600         if P(i)>=0
601             esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')'];
602         else
603             esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')'];
604         end
605     else
606         if P(i)>=0 && i~=1
607             esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')^',num2str(
        ↪ Npolinomio_temp_hum+1-i,'%1.0f')];
608         else
609             esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')^',num2str(
        ↪ Npolinomio_temp_hum+1-i,'%1.0f')];
610         end
611     end
612 end
613 if P(Npolinomio_temp_hum+1)>=0
614     esc_pol=[esc_pol,'+',num2str(P(Npolinomio_temp_hum+1),'%1.3g')];
615 else
```

```
616     esc_pol=[esc_pol,num2str(P(Npolinomio_temp_hum+1),'%1.3g')];
617 end
618 text(0.4,0.9,{['H_r=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'%'],['R^2=',num2str(R(1,2)^2,'%1.3g')
        ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
619
620
621 subplot(2,1,2)
622 scatter(temp,hum)
623 grid on
624 xlabel('Temperatura (°C)')
625 ylabel('Humedad Rel. (%)')
626 P = polyfit(temp-Tm,hum,Npolinomio_temp_hum);
627 xlinea=[min(temp):(max(temp)-min(temp))/20:max(temp)];
628 flinea = polyval(P,xlinea-Tm);
629 hold on
630 humprueba=polyval(P,temp-Tm);
631 tsstd = std(hum-humprueba);
632 plot(xlinea,flinea,'r')
633 plot(xlinea,flinea-1.96*tsstd,'g')
634 plot(xlinea,flinea+1.96*tsstd,'g')
635 R = corrcoef(hum,humprueba);
636 esc_pol=[];
637 for i=1:Npolinomio_temp_hum
638     if i==Npolinomio_temp_hum
639         if P(i)>=0
640             esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')'];
641         else
642             esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')'];
643         end
644     else
645         if P(i)>=0 && i~=1
646             esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')^',num2str(
        ↪ Npolinomio_temp_hum+1-i,'%1.0f')];
647         else
648             esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')^',num2str(
        ↪ Npolinomio_temp_hum+1-i,'%1.0f')];
649         end
650     end
651 end
652 if P(Npolinomio_temp_hum+1)>=0
653     esc_pol=[esc_pol,'+',num2str(P(Npolinomio_temp_hum+1),'%1.3g')];
654 else
655     esc_pol=[esc_pol,num2str(P(Npolinomio_temp_hum+1),'%1.3g')];
656 end
657 text(0.4,0.9,{['H_r=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'%'],['R^2=',num2str(R(1,2)^2,'%1.3g')
        ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
658
659
660 % F1 v/s T
661
662 Npolinomio=2;
663 figure(2)
664 subplot(2,3,1)
665 scatter(temp,f1)
666 xlabel('Temperatura (°C)')
667 ylabel('F_1 (Hz)')
```

```
grid on
P = polyfit(temp-Tm,f1,Npolinomio);
xlinea=[min(temp):(max(temp)-min(temp))/20:max(temp)];
flinea = polyval(P,xlinea-Tm);
hold on
fprueba=polyval(P,temp-Tm);
tsstd = std(fprueba-f1);
plot(xlinea,flinea,'r')
plot(xlinea,flinea-1.96*tsstd,'g')
plot(xlinea,flinea+1.96*tsstd,'g')
R = corrcoef(fprueba,f1);
esc_pol=[];
for i=1:Npolinomio
    if i==Npolinomio
        if P(i)>=0
            esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')'];
        else
            esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')'];
        end
    else
        if P(i)>=0 && i~=1
            esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')^',num2str(
        ↪ Npolinomio+1-i,'%1.0f')];
        else
            esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')^',num2str(Npolinomio
        ↪ +1-i,'%1.0f')];
        end
    end
end
if P(Npolinomio+1)>=0
    esc_pol=[esc_pol,'+',num2str(P(Npolinomio+1),'%1.3g')];
else
    esc_pol=[esc_pol,num2str(P(Npolinomio+1),'%1.3g')];
end
text(0.4,0.9,{['F_1=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')
        ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')

figure(3)
subplot(2,3,1)
scatter(f1,fprueba,'k')
grid on
hold on
xL(2)=max(max([f1(:),fprueba(:)]));
xL(1)=min(min([f1(:),fprueba(:)]));
plot(xL,xL,'r')
plot(xL,xL+1.96*tsstd,'g')
plot(xL,xL-1.96*tsstd,'g')
xlabel('F_1 (Hz)')
ylabel('F_1^{(est)} (Hz)')
text(0.05,1,{['F_1=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')
        ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
axis equal


figure(4)
mu=mean(fprueba-f1);
```

```
720 subplot(2,3,1)
721 histogram(fprueba-f1,nbins,'Normalization','pdf')
722 x1=get(gca,'xlim');
723 xx=x1(1):(x1(2)-x1(1))/50:x1(2);
724 hold on
725 plot(xx,pdf('Normal',xx,mu,tsstd),'r')
726 xlabel('\epsilon_r')
727 title('F_1 v/s T')
728 grid on
729
730
731
732
733 Npolinomio_c=1;
734
735 figure(5)
736 subplot(3,2,1)
737 scatter(hum,f1./fprueba,'k')
738 grid on
739 Pc = polyfit(hum-Hrm,f1./fprueba,Npolinomio_c);
740 xlinea=[min(hum):(max(hum)-min(hum))/20:max(hum)];
741 flinea = polyval(Pc,xlinea-Hrm);
742 hold on
743 fcorreccion=polyval(Pc,hum-Hrm);
744 tsstd = std(fcorreccion-f1./fprueba);
745 plot(xlinea,flinea,'r')
746 plot(xlinea,flinea-1.96*tsstd,'g')
747 plot(xlinea,flinea+1.96*tsstd,'g')
748 R = corrcoef(fcorreccion,f1./fprueba);
749 esc_polc=[];
750 for i=1:Npolinomio_c
751     if i==Npolinomio_c
752         if Pc(i)>=0
753             esc_polc=[esc_polc,'+',num2str(Pc(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')'];
754         else
755             esc_polc=[esc_polc,num2str(Pc(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')'];
756         end
757     else
758         if Pc(i)>=0 && i~=1
759             esc_polc=[esc_polc,'+',num2str(Pc(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')^',num2str(
        ↪ Npolinomio_c+1-i,'%1.0f')];
760         else
761             esc_polc=[esc_polc,num2str(Pc(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')^',num2str(
        ↪ Npolinomio_c+1-i,'%1.0f')];
762         end
763     end
764 end
765
766 if Pc(Npolinomio_c+1)>=0
767     esc_polc=[esc_polc,'+',num2str(Pc(Npolinomio_c+1),'%1.3g')];
768 else
769     esc_polc=[esc_polc,num2str(Pc(Npolinomio_c+1),'%1.3g')];
770 end
771 text(0.05,1,{['F_1/F^{est}=',esc_polc],['\sigma=',num2str(tsstd,'%1.3g')],['R^2=',num2str(R(1,2)^2,'
        ↪ %1.3g')]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','
        ↪ white')
```

```
772
773 ylabel('F_1/F^{est}')
774 xlabel('Humedad Rel. (%)')
775 subplot(3,2,2)
776 scatter(hum,f1-fprueba,'k')
777 ylabel('F_1-F^{est}')
778 grid on
779 xlabel('Humedad Rel. (%)')
780
781
782
783 figure(105)
784 subplot(1,2,1)
785 scatter(hum-humprueba,f1./fprueba,'k')
786 grid on
787 Pc = polyfit(hum-humprueba,f1./fprueba,Npolinomio_c);
788 xlinea=[min(hum-humprueba):(max(hum-humprueba)-min(hum-humprueba))/20:max(hum-
        ↪ humprueba)];
789 flinea = polyval(Pc,xlinea);
790 hold on
791 fcorreccion2=polyval(Pc,hum-humprueba);
792 tsstd = std(fcorreccion2-f1./fprueba);
793 plot(xlinea,flinea,'r')
794 plot(xlinea,flinea-1.96*tsstd,'g')
795 plot(xlinea,flinea+1.96*tsstd,'g')
796 R = corrcoef(fcorreccion2,f1./fprueba);
797 esc_polc2=[];
798 for i=1:Npolinomio_c
799     if i==Npolinomio_c
800         if Pc(i)>=0
801             esc_polc2=[esc_polc2,'+',num2str(Pc(i),'%1.3g'),'*(Hr-Hr^{est}',')'];
802         else
803             esc_polc2=[esc_polc2,num2str(Pc(i),'%1.3g'),'*(Hr-Hr^{est}',')'];
804         end
805     else
806         if Pc(i)>=0 && i~=1
807             esc_polc2=[esc_polc2,'+',num2str(Pc(i),'%1.3g'),'*(Hr-Hr^{est}',')^',num2str(Npolinomio_c
        ↪ +1-i,'%1.0f')];
808         else
809             esc_polc2=[esc_polc2,num2str(Pc(i),'%1.3g'),'*(Hr-Hr^{est}',')^',num2str(Npolinomio_c+1-i,'
        ↪ %1.0f')];
810         end
811     end
812 end
813
814 if Pc(Npolinomio_c+1)>=0
815     esc_polc2=[esc_polc2,'+',num2str(Pc(Npolinomio_c+1),'%1.3g')];
816 else
817     esc_polc2=[esc_polc2,num2str(Pc(Npolinomio_c+1),'%1.3g')];
818 end
819 text(0.05,1,{['F_1/F^{est}=',esc_polc2],['\sigma=',num2str(tsstd,'%1.3g')],['R^2=',num2str(R(1,2)^2,'
        ↪ %1.3g')]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','
        ↪ white')
820
821 ylabel('F_1/F^{est}')
822 xlabel('Hr-Hr^{est} (%)')
```

```
subplot(1,2,2)
scatter(hum-humprueba,f1-fprueba,'k')
ylabel('F_1-F^{est}')
grid on
xlabel('Hr-Hr^{est} (%)')
```

```
figure(6)
subplot(2,3,1)
scatter(f1,fprueba.*fcorreccion,'k')
grid on
hold on
xL(2)=max(max([f1(:),fprueba(:).*fcorreccion(:)]));
xL(1)=min(min([f1(:),fprueba(:).*fcorreccion(:)]));
tsstd = std(fcorreccion(:).*fprueba(:)-f1);
plot(xL,xL,'r')
plot(xL,xL+1.96*tsstd,'g')
plot(xL,xL-1.96*tsstd,'g')
xlabel('F_1 (Hz)')
ylabel('F_1^{(est)} (Hz)')
R = corrcoef(fprueba(:).*fcorreccion(:),f1);
text(0.05,1,{['F_1=\{',esc_pol,'\} x '],['    \{',esc_polc,'\}'],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R
    ↪ ^2=',num2str(R(1,2)^2,'%1.3g')]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth'
    ↪ ,0.5,'BackgroundColor','white')
axis equal
```

```
figure(106)
subplot(2,1,1)
scatter(f1,fprueba.*fcorreccion2,'k')
grid on
hold on
xL(2)=max(max([f1(:),fprueba(:).*fcorreccion2(:)]));
xL(1)=min(min([f1(:),fprueba(:).*fcorreccion2(:)]));
tsstd = std(fcorreccion2(:).*fprueba(:)-f1);
plot(xL,xL,'r')
plot(xL,xL+1.96*tsstd,'g')
plot(xL,xL-1.96*tsstd,'g')
xlabel('F_1 (Hz)')
ylabel('F_1^{(est)} (Hz)')
R = corrcoef(fprueba(:).*fcorreccion2(:),f1);
text(0.05,1,{['F_1=\{',esc_pol,'\} x '],['    \{',esc_polc2,'\}'],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R
    ↪ ^2=',num2str(R(1,2)^2,'%1.3g')]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth'
    ↪ ,0.5,'BackgroundColor','white')
axis equal
```

```
figure(1)
subplot(4,1,2)
hold on
```

```
875 plot(DN,fprueba(:).*fcorreccion(:),'b--',DN,fprueba(:),'--r',DN,fprueba(:).*fcorreccion2(:),'--g',DN_2,
        ↪ f1arx,'--c')
876 legend('data','T y Hr','T','T y Hr v2','SS')
877
878
879 figure(6)
880 mu=mean(fprueba(:).*fcorreccion(:)-f1);
881 subplot(2,3,4)
882 histogram(fprueba(:).*fcorreccion(:)-f1,nbins,'Normalization','pdf')
883 x1=get(gca,'xlim');
884 xx=x1(1):(x1(2)-x1(1))/50:x1(2);
885 hold on
886 plot(xx,pdf('Normal',xx,mu,tsstd),'r')
887 xlabel('\epsilon_r')
888 title('F_1 v/s T-Hr')
889 grid on
890
891 figure(106)
892 mu=mean(fprueba(:).*fcorreccion2(:)-f1);
893 subplot(2,1,2)
894 histogram(fprueba(:).*fcorreccion2(:)-f1,nbins,'Normalization','pdf')
895 x1=get(gca,'xlim');
896 xx=x1(1):(x1(2)-x1(1))/50:x1(2);
897 hold on
898 plot(xx,pdf('Normal',xx,mu,tsstd),'r')
899 xlabel('\epsilon_r')
900 title('F_1 v/s T-Hr')
901 grid on
902
903
904
905 % ['F_1=',num2str(P(1),'%1.3g'),'*T+',num2str(P(2),'%1.3g')],
906
907 figure(2)
908 subplot(2,3,2)
909 scatter(temp,f2)
910 xlabel('Temperatura (°C)')
911 ylabel('F_2 (Hz)')
912 grid on
913 P = polyfit(temp-Tm,f2,Npolinomio);
914 xlinea=[min(temp):(max(temp)-min(temp))/20:max(temp)];
915 flinea = polyval(P,xlinea-Tm);
916 hold on
917 fprueba=polyval(P,temp-Tm);
918 tsstd = std(fprueba-f2);
919 plot(xlinea,flinea,'r')
920 plot(xlinea,flinea-1.96*tsstd,'g')
921 plot(xlinea,flinea+1.96*tsstd,'g')
922 R = corrcoef(fprueba,f2);
923 esc_pol=[];
924 for i=1:Npolinomio
925     if i==Npolinomio
926         if P(i)>=0
927             esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')'];
928         else
929             esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')'];
```

```
930         end
931     else
932         if P(i)>=0 && i~=1
933             esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')^',num2str(
        ↪ Npolinomio+1-i,'%1.0f')];
934         else
935             esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')^',num2str(Npolinomio
        ↪ +1-i,'%1.0f')];
936         end
937     end
938 end
939 if P(Npolinomio+1)>=0
940     esc_pol=[esc_pol,'+',num2str(P(Npolinomio+1),'%1.3g')];
941 else
942     esc_pol=[esc_pol,num2str(P(Npolinomio+1),'%1.3g')];
943 end
944 text(0.4,0.9,{['F_2=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')
        ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
945
946 figure(3)
947 subplot(2,3,2)
948 scatter(f2,fprueba,'k')
949 grid on
950 hold on
951 xL(2)=max(max([f2(:),fprueba(:)]));
952 xL(1)=min(min([f2(:),fprueba(:)]));
953 plot(xL,xL,'r')
954 plot(xL,xL+1.96*tsstd,'g')
955 plot(xL,xL-1.96*tsstd,'g')
956 xlabel('F_2 (Hz)')
957 ylabel('F_2^{(est)} (Hz)')
958 text(0.05,1,{['F_2=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')
        ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
959 axis equal
960
961 figure(4)
962 mu=mean(fprueba-f2);
963 subplot(2,3,2)
964 histogram(fprueba-f2,nbins,'Normalization','pdf')
965 x1=get(gca,'xlim');
966 xx=x1(1):(x1(2)-x1(1))/50:x1(2);
967 hold on
968 plot(xx,pdf('Normal',xx,mu,tsstd),'r')
969 xlabel('\epsilon_r')
970 title('F_2 v/s T')
971 grid on
972
973 figure(5)
974 subplot(3,2,3)
975 scatter(hum,f2./fprueba,'k')
976 grid on
977 Pc = polyfit(hum-Hrm,f2./fprueba,Npolinomio_c);
978 xlinea=[min(hum):(max(hum)-min(hum))/20:max(hum)];
979 flinea = polyval(Pc,xlinea-Hrm);
980 hold on
981 fcorreccion=polyval(Pc,hum-Hrm);
```

```
tsstd = std(fcorreccion-f2./fprueba);
plot(xlinea,flinea,'r')
plot(xlinea,flinea-1.96*tsstd,'g')
plot(xlinea,flinea+1.96*tsstd,'g')
R = corrcoef(fcorreccion,f2./fprueba);
esc_polc=[];
for i=1:Npolinomio_c
    if i==Npolinomio_c
        if Pc(i)>=0
            esc_polc=[esc_polc,'+',num2str(Pc(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')'];
        else
            esc_polc=[esc_polc,num2str(Pc(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')'];
        end
    else
        if Pc(i)>=0 && i~=1
            esc_polc=[esc_polc,'+',num2str(Pc(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')^',num2str(
    ↪ Npolinomio_c+1-i,'%1.0f')];
        else
            esc_polc=[esc_polc,num2str(Pc(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')^',num2str(
    ↪ Npolinomio_c+1-i,'%1.0f')];
        end
    end
end

if Pc(Npolinomio_c+1)>=0
    esc_polc=[esc_polc,'+',num2str(Pc(Npolinomio_c+1),'%1.3g')];
else
    esc_polc=[esc_polc,num2str(Pc(Npolinomio_c+1),'%1.3g')];
end
text(0.05,1,{['F_2/F^{est}=',esc_polc],['\sigma=',num2str(tsstd,'%1.3g')],['R^2=',num2str(R(1,2)^2,'
    ↪ %1.3g')]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','
    ↪ white')

ylabel('F_2/F^{est}')
xlabel('Humedad Rel. (%)')
subplot(3,2,4)
scatter(hum,f2-fprueba,'k')
ylabel('F_2-F^{est}')
grid on
xlabel('Humedad Rel. (%)')

figure(6)
subplot(2,3,2)
scatter(f2,fprueba.*fcorreccion,'k')
grid on
hold on
xL(2)=max(max([f2(:),fprueba(:).*fcorreccion(:)]));
xL(1)=min(min([f2(:),fprueba(:).*fcorreccion(:)]));
tsstd = std(fcorreccion(:).*fprueba(:)-f2);
plot(xL,xL,'r')
plot(xL,xL+1.96*tsstd,'g')
plot(xL,xL-1.96*tsstd,'g')
xlabel('F_2 (Hz)')
ylabel('F_2^{(est)} (Hz)')
R = corrcoef(fprueba(:).*fcorreccion(:),f2);
text(0.05,1,{['F_2=\{',esc_pol,'\} x '],['     \{',esc_polc,'\}'],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R
```

```
    ↪ ^2=',num2str(R(1,2)^2,' %1.3g')]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth'
    ↪ ,0.5,'BackgroundColor','white')
axis equal


figure(1)
subplot(4,1,3)
hold on
plot(DN,fprueba(:).*fcorreccion(:),'b--',DN,fprueba(:),'--r')


figure(6)
mu=mean(fprueba(:).*fcorreccion(:)-f2);
subplot(2,3,5)
histogram(fprueba(:).*fcorreccion(:)-f2,nbins,'Normalization','pdf')
x1=get(gca,'xlim');
xx=x1(1):(x1(2)-x1(1))/50:x1(2);
hold on
plot(xx,pdf('Normal',xx,mu,tsstd),'r')
xlabel('\epsilon_r')
title('F_2 v/s T-Hr')
grid on

% figure(5)
% subplot(3,2,3)
% scatter(hum,f2./fprueba,'k')
% grid on
% ylabel('F_2/F^{est}')
% xlabel('Humedad Rel. (%)')
% subplot(3,2,4)
% scatter(hum,f2-fprueba,'k')
% ylabel('F_2-F^{est}')
% grid on
% xlabel('Humedad Rel. (%)')


figure(2)
subplot(2,3,3)
scatter(temp,f3)
xlabel('Temperatura (°C)')
ylabel('F_3 (Hz)')
grid on
P = polyfit(temp-Tm,f3,Npolinomio);
xlinea=[min(temp):(max(temp)-min(temp))/20:max(temp)];
flinea = polyval(P,xlinea-Tm);
hold on
fprueba=polyval(P,temp-Tm);
tsstd = std(fprueba-f3);
plot(xlinea,flinea,'r')
plot(xlinea,flinea-1.96*tsstd,'g')
plot(xlinea,flinea+1.96*tsstd,'g')
R = corrcoef(fprueba,f3);
esc_pol=[];
for i=1:Npolinomio
    if i==Npolinomio
        if P(i)>=0
```

```
1088            esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')'];
1089        else
1090            esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')'];
1091        end
1092    else
1093        if P(i)>=0 && i~=1
1094            esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')^',num2str(
        ↪ Npolinomio+1-i,'%1.0f')];
1095        else
1096            esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(T-',num2str(Tm,'%1.0f'),')^',num2str(Npolinomio
        ↪ +1-i,'%1.0f')];
1097        end
1098    end
1099end
1100if P(Npolinomio+1)>=0
1101    esc_pol=[esc_pol,'+',num2str(P(Npolinomio+1),'%1.3g')];
1102else
1103    esc_pol=[esc_pol,num2str(P(Npolinomio+1),'%1.3g')];
1104end
1105text(0.4,0.9,{['F_3=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')
        ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
1106
1107figure(3)
1108subplot(2,3,3)
1109scatter(f3,fprueba,'k')
1110grid on
1111hold on
1112xL(2)=max(max([f3(:),fprueba(:)]));
1113xL(1)=min(min([f3(:),fprueba(:)]));
1114plot(xL,xL,'r')
1115plot(xL,xL+1.96*tsstd,'g')
1116plot(xL,xL-1.96*tsstd,'g')
1117xlabel('F_3 (Hz)')
1118ylabel('F_3^{(est)} (Hz)')
1119text(0.05,1,{['F_3=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')
        ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
1120axis equal
1121
1122
1123figure(4)
1124mu=mean(fprueba-f3);
1125subplot(2,3,3)
1126histogram(fprueba-f3,nbins,'Normalization','pdf')
1127x1=get(gca,'xlim');
1128xx=x1(1):(x1(2)-x1(1))/50:x1(2);
1129hold on
1130plot(xx,pdf('Normal',xx,mu,tsstd),'r')
1131xlabel('\epsilon_r')
1132title('F_3 v/s T')
1133grid on
1134
1135
1136figure(5)
1137subplot(3,2,5)
1138scatter(hum,f3./fprueba,'k')
1139grid on
```

```
Pc = polyfit(hum-Hrm,f3./fprueba,Npolinomio_c);
xlinea=[min(hum):(max(hum)-min(hum))/20:max(hum)];
flinea = polyval(Pc,xlinea-Hrm);
hold on
fcorreccion=polyval(Pc,hum-Hrm);
tsstd = std(fcorreccion-f3./fprueba);
plot(xlinea,flinea,'r')
plot(xlinea,flinea-1.96*tsstd,'g')
plot(xlinea,flinea+1.96*tsstd,'g')
R = corrcoef(fcorreccion,f3./fprueba);
esc_polc=[];
for i=1:Npolinomio_c
  if i==Npolinomio_c
    if Pc(i)>=0
      esc_polc=[esc_polc,'+',num2str(Pc(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')'];
    else
      esc_polc=[esc_polc,num2str(Pc(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')'];
    end
  else
    if Pc(i)>=0 && i~=1
      esc_polc=[esc_polc,'+',num2str(Pc(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')^',num2str(
    ↪ Npolinomio_c+1-i,'%1.0f')];
    else
      esc_polc=[esc_polc,num2str(Pc(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')^',num2str(
    ↪ Npolinomio_c+1-i,'%1.0f')];
    end
  end
end

if Pc(Npolinomio_c+1)>=0
  esc_polc=[esc_polc,'+',num2str(Pc(Npolinomio_c+1),'%1.3g')];
else
  esc_polc=[esc_polc,num2str(Pc(Npolinomio_c+1),'%1.3g')];
end
text(0.05,1,{['F_3/F^{est}=',esc_polc],['\sigma=',num2str(tsstd,'%1.3g')],['R^2=',num2str(R(1,2)^2,'
    ↪ %1.3g')]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','
    ↪ white')

ylabel('F_3/F^{est}')
xlabel('Humedad Rel. (%)')
subplot(3,2,6)
scatter(hum,f3-fprueba,'k')
ylabel('F_3-F^{est}')
grid on
xlabel('Humedad Rel. (%)')

figure(6)
subplot(2,3,3)
scatter(f3,fprueba.*fcorreccion,'k')
grid on
hold on
xL(2)=max(max([f3(:),fprueba(:).*fcorreccion(:)]));
xL(1)=min(min([f3(:),fprueba(:).*fcorreccion(:)]));
tsstd = std(fcorreccion(:).*fprueba(:)-f3);
plot(xL,xL,'r')
plot(xL,xL+1.96*tsstd,'g')
```

```
plot(xL,xL−1.96∗tsstd,'g')
xlabel('F_3 (Hz)')
ylabel('F_3^{(est)} (Hz)')
R = corrcoef(fprueba(:).∗fcorreccion(:),f3);
text(0.05,1,{['F_3=\{',esc_pol,'\} x '],['     \{',esc_polc,'\}'],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')]},'units','normalized','EdgeColor','k','LineStyle','−','LineWidth',0.5,'BackgroundColor','white')
axis equal


figure(1)
subplot(4,1,4)
hold on
plot(DN,fprueba(:).∗fcorreccion(:),'b−−',DN,fprueba(:),'−−r')

figure(6)
mu=mean(fprueba(:).∗fcorreccion(:)−f3);
subplot(2,3,6)
histogram(fprueba(:).∗fcorreccion(:)−f3,nbins,'Normalization','pdf')
x1=get(gca,'xlim');
xx=x1(1):(x1(2)−x1(1))/50:x1(2);
hold on
plot(xx,pdf('Normal',xx,mu,tsstd),'r')
xlabel('\epsilon_r')
title('F_3 v/s T−Hr')
grid on


% figure(5)
% subplot(3,2,5)
% scatter(hum,f3./fprueba,'k')
% grid on
% ylabel('F_3/F^{est}')
% xlabel('Humedad Rel. (%)')
% subplot(3,2,6)
% scatter(hum,f3−fprueba,'k')
% ylabel('F_3−F^{est}')
% grid on
% xlabel('Humedad Rel. (%)')



Npolinomio=3;

figure(2)
subplot(2,3,4)
scatter(hum,f1)
xlabel('Humedad Rel. (%)')
ylabel('F_1 (Hz)')
grid on
P = polyfit(hum−Hrm,f1,Npolinomio);
xlinea=[min(hum):(max(hum)−min(hum))/20:max(hum)];
flinea = polyval(P,xlinea−Hrm);
hold on
fprueba=polyval(P,hum−Hrm);
tsstd = std(fprueba−f1);
```

```
plot(xlinea,flinea,'r')
plot(xlinea,flinea-1.96*tsstd,'g')
plot(xlinea,flinea+1.96*tsstd,'g')
R = corrcoef(fprueba,f1);
esc_pol=[];
for i=1:Npolinomio
    if i==Npolinomio
        if P(i)>=0
            esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')'];
        else
            esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')'];
        end
    else
        if P(i)>=0 && i~=1
            esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')^',num2str(
        ↪ Npolinomio+1-i,'%1.0f')];
        else
            esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')^',num2str(Npolinomio
        ↪ +1-i,'%1.0f')];
        end
    end
end
if P(Npolinomio+1)>=0
    esc_pol=[esc_pol,'+',num2str(P(Npolinomio+1),'%1.3g')];
else
    esc_pol=[esc_pol,num2str(P(Npolinomio+1),'%1.3g')];
end
text(0.05,1,{['F_1=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')
        ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')

figure(3)
subplot(2,3,4)
scatter(f1,fprueba,'k')
grid on
hold on
xL(2)=max(max([f1(:),fprueba(:)]));
xL(1)=min(min([f1(:),fprueba(:)]));
plot(xL,xL,'r')
plot(xL,xL+1.96*tsstd,'g')
plot(xL,xL-1.96*tsstd,'g')
xlabel('F_1 (Hz)')
ylabel('F_1^{(est)} (Hz)')
text(0.05,1,{['F_1=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')
        ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
axis equal


figure(4)
mu=mean(fprueba-f1);
subplot(2,3,4)
histogram(fprueba-f1,nbins,'Normalization','pdf')
x1=get(gca,'xlim');
xx=x1(1):(x1(2)-x1(1))/50:x1(2);
hold on
plot(xx,pdf('Normal',xx,mu,tsstd),'r')
xlabel('\epsilon_r')
```

```
1298 title('F_1 v/s Hr')
1299 grid on
1300
1301
1302
1303 figure(2)
1304 subplot(2,3,5)
1305 scatter(hum,f2)
1306 xlabel('Humedad Rel. (%)')
1307 ylabel('F_2 (Hz)')
1308 grid on
1309 P = polyfit(hum-Hrm,f2,Npolinomio);
1310 xlinea=[min(hum):(max(hum)-min(hum))/20:max(hum)];
1311 flinea = polyval(P,xlinea-Hrm);
1312 hold on
1313 fprueba=polyval(P,hum-Hrm);
1314 tsstd = std(fprueba-f2);
1315 plot(xlinea,flinea,'r')
1316 plot(xlinea,flinea-1.96*tsstd,'g')
1317 plot(xlinea,flinea+1.96*tsstd,'g')
1318 R = corrcoef(fprueba,f2);
1319 esc_pol=[];
1320 for i=1:Npolinomio
1321     if i==Npolinomio
1322         if P(i)>=0
1323             esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')'];
1324         else
1325             esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')'];
1326         end
1327     else
1328         if P(i)>=0 && i~=1
1329             esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')^',num2str(
        ↪ Npolinomio+1-i,'%1.0f')];
1330         else
1331             esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')^',num2str(Npolinomio
        ↪ +1-i,'%1.0f')];
1332         end
1333     end
1334 end
1335 if P(Npolinomio+1)>=0
1336     esc_pol=[esc_pol,'+',num2str(P(Npolinomio+1),'%1.3g')];
1337 else
1338     esc_pol=[esc_pol,num2str(P(Npolinomio+1),'%1.3g')];
1339 end
1340 text(0.05,1,{['F_2=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')
        ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
1341
1342 figure(3)
1343 subplot(2,3,5)
1344 scatter(f2,fprueba,'k')
1345 grid on
1346 hold on
1347 xL(2)=max(max([f2(:),fprueba(:)]));
1348 xL(1)=min(min([f2(:),fprueba(:)]));
1349 plot(xL,xL,'r')
1350 plot(xL,xL+1.96*tsstd,'g')
```

```
plot(xL,xL-1.96*tsstd,'g')
xlabel('F_2 (Hz)')
ylabel('F_2^{(est)} (Hz)')
text(0.05,1,{['F_2=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')
    ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
axis equal

figure(4)
mu=mean(fprueba-f2);
subplot(2,3,5)
histogram(fprueba-f2,nbins,'Normalization','pdf')
x1=get(gca,'xlim');
xx=x1(1):(x1(2)-x1(1))/50:x1(2);
hold on
plot(xx,pdf('Normal',xx,mu,tsstd),'r')
xlabel('\epsilon_r')
title('F_2 v/s Hr')
grid on



figure(2)
subplot(2,3,6)
scatter(hum,f3)
xlabel('Humedad Rel. (%)')
ylabel('F_3 (Hz)')
grid on
P = polyfit(hum-Hrm,f3,Npolinomio);
xlinea=[min(hum):(max(hum)-min(hum))/20:max(hum)];
flinea = polyval(P,xlinea-Hrm);
hold on
fprueba=polyval(P,hum-Hrm);
tsstd = std(fprueba-f3);
plot(xlinea,flinea,'r')
plot(xlinea,flinea-1.96*tsstd,'g')
plot(xlinea,flinea+1.96*tsstd,'g')
R = corrcoef(fprueba,f3);
esc_pol=[];
for i=1:Npolinomio
  if i==Npolinomio
    if P(i)>=0
      esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')'];
    else
      esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')'];
    end
  else
    if P(i)>=0 && i~=1
      esc_pol=[esc_pol,'+',num2str(P(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')^',num2str(
    ↪ Npolinomio+1-i,'%1.0f')];
    else
      esc_pol=[esc_pol,num2str(P(i),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')^',num2str(Npolinomio
    ↪ +1-i,'%1.0f')];
    end
  end
end
if P(Npolinomio+1)>=0
```

```
1404    esc_pol=[esc_pol,'+',num2str(P(Npolinomio+1),'%1.3g')];
1405else
1406    esc_pol=[esc_pol,num2str(P(Npolinomio+1),'%1.3g')];
1407end
1408text(0.05,1,{['F_3=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')
        ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
1409
1410figure(3)
1411subplot(2,3,6)
1412scatter(f3,fprueba,'k')
1413grid on
1414hold on
1415xL(2)=max(max([f3(:),fprueba(:)]));
1416xL(1)=min(min([f3(:),fprueba(:)]));
1417plot(xL,xL,'r')
1418plot(xL,xL+1.96*tsstd,'g')
1419plot(xL,xL-1.96*tsstd,'g')
1420xlabel('F_3 (Hz)')
1421ylabel('F_3^{(est)} (Hz)')
1422text(0.05,1,{['F_3=',esc_pol],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')
        ↪ ]},'units','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
1423axis equal
1424
1425% figure(2)
1426% subplot(2,3,6)
1427% scatter(hum,f3)
1428% xlabel('Humedad Rel.(%)')
1429% grid on
1430% ylabel('F_3 (Hz)')
1431% P = polyfit(hum,f3,Npolinomio);
1432% xlinea=[min(hum),max(hum)];
1433% flinea = polyval(P,xlinea);
1434% hold on
1435% fprueba=polyval(P,hum);
1436% tsstd = std(fprueba-f3);
1437% plot(xlinea,flinea,'r')
1438% plot(xlinea,flinea-1.96*tsstd,'g')
1439% plot(xlinea,flinea+1.96*tsstd,'g')
1440% R = corrcoef(fprueba,f3);
1441% text(0.05,0.85,{['F_3=',num2str(P(1),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')+',num2str(P(2)+P(1)*
        ↪ Hrm,'%1.2f')],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')]},'units
        ↪ ','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
1442%
1443% figure(3)
1444% subplot(2,3,6)
1445% scatter(f3,fprueba,'k')
1446% grid on
1447% hold on
1448% xL(2)=max(max([f3(:),fprueba(:)]));
1449% xL(1)=min(min([f3(:),fprueba(:)]));
1450% plot(xL,xL,'r')
1451% plot(xL,xL+1.96*tsstd,'g')
1452% plot(xL,xL-1.96*tsstd,'g')
1453% xlabel('F_3 (Hz)')
1454% ylabel('F_3^{(est)} (Hz)')
1455% text(0.05,0.85,{['F_3=',num2str(P(1),'%1.3g'),'*(Hr-',num2str(Hrm,'%1.1f'),')+',num2str(P(2)+P(1)*
```

```
    ↪ Hrm,'%1.2f')],['\sigma=',num2str(tsstd,'%1.3g'),'Hz'],['R^2=',num2str(R(1,2)^2,'%1.3g')]},'units
    ↪ ','normalized','EdgeColor','k','LineStyle','-','LineWidth',0.5,'BackgroundColor','white')
% axis equal

figure(4)
mu=mean(fprueba-f3);
subplot(2,3,6)
histogram(fprueba-f3,nbins,'Normalization','pdf')
x1=get(gca,'xlim');
xx=x1(1):(x1(2)-x1(1))/50:x1(2);
hold on
plot(xx,pdf('Normal',xx,mu,tsstd),'r')
xlabel('\epsilon_r')
title('F_3 v/s Hr')
grid on
```

A.15. micro.sh

Archivo ejecutable que se programó en la Raspberry pi 3 para poder ejecutar el código expuesto en 8.1.11 cada una hora.

```
#!/bin/sh

cd /home/pi/pObspy/micros
python pengeo3.py


octave --persist --eval 'shmpen'
```

Anexo B

Registros

B.1. 21 de Octubre de 2021

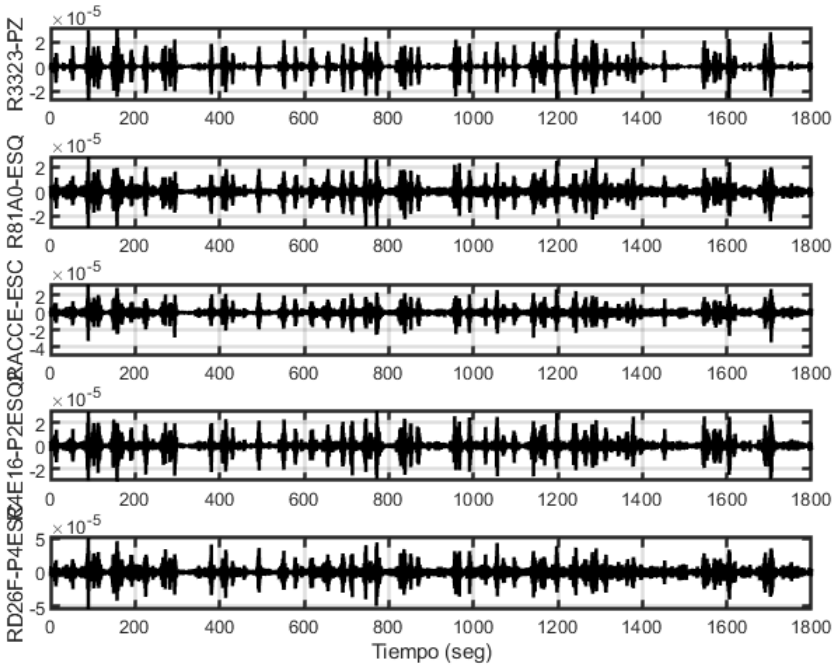

Figura B.1: Registro de sensores el día 21-10-2021 de 01:00 a 01:30

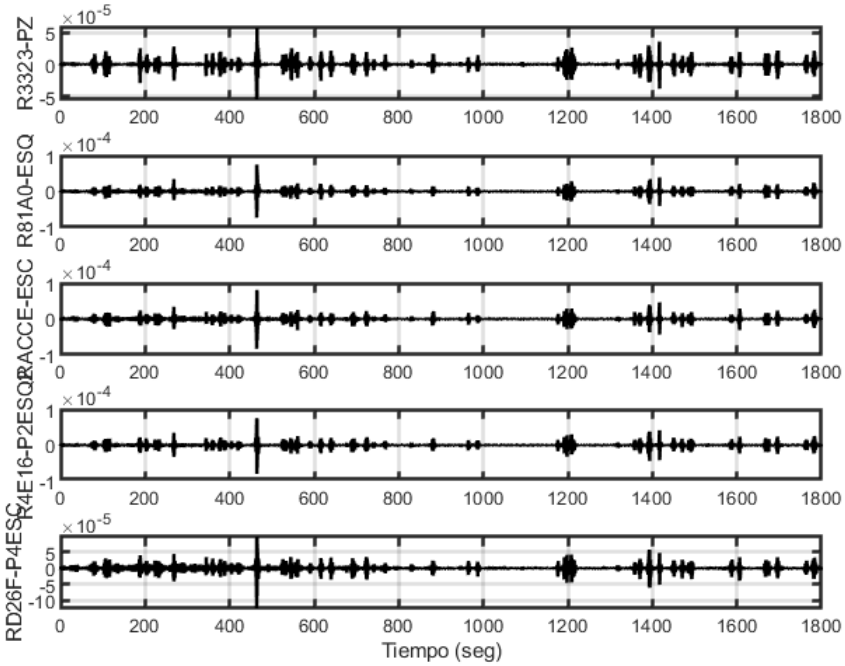

Figura B.2: Registro de sensores el día 21-10-2021 de 01:30 a 02:00

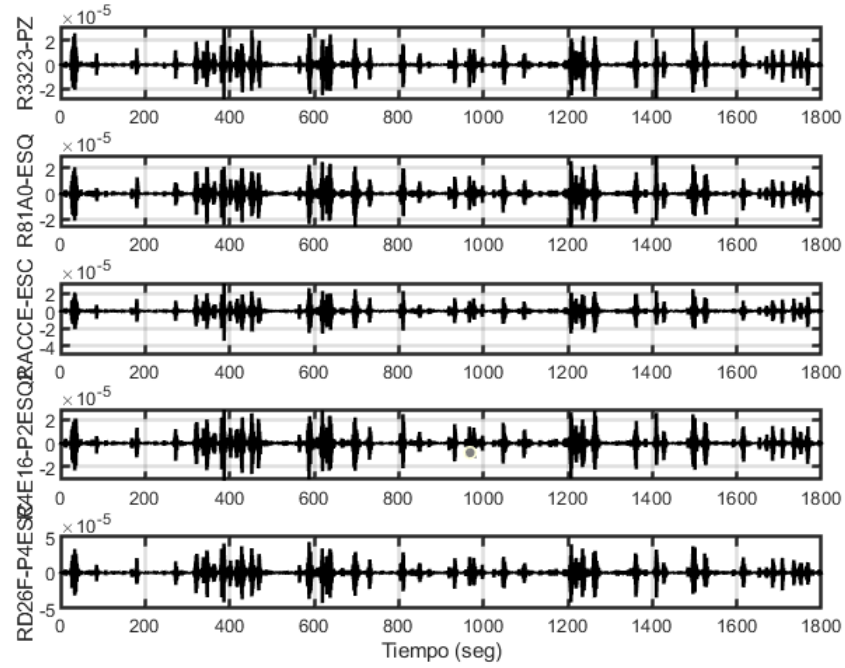

Figura B.3: Registro de sensores el día 21-10-2021 de 02:00 a 02:30

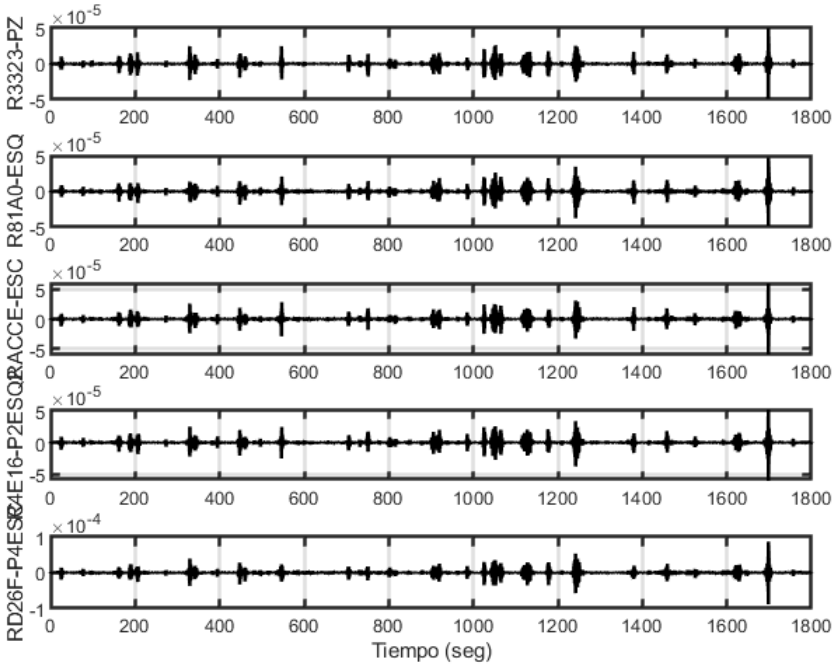

Figura B.4: Registro de sensores el día 21-10-2021 de 02:30 a 03:00

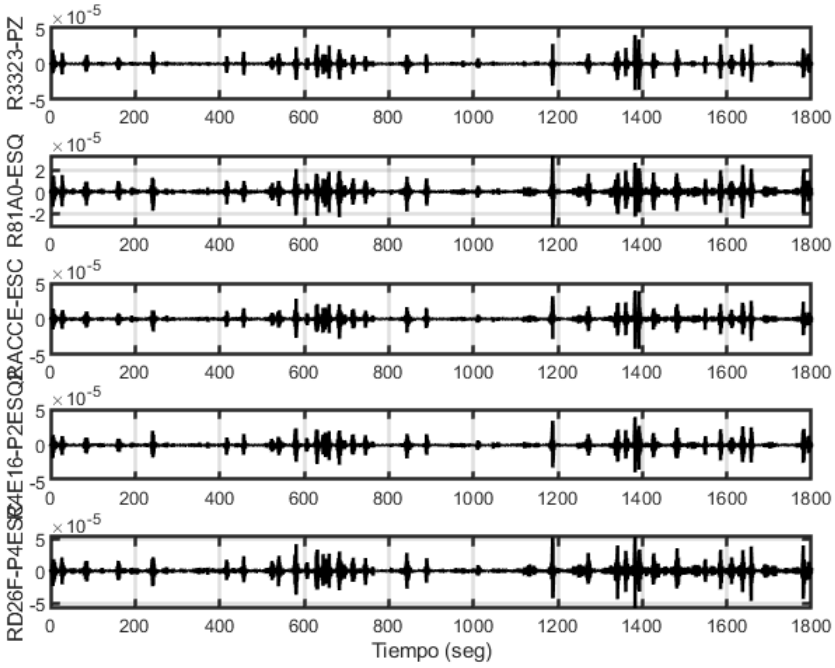

Figura B.5: Registro de sensores el día 21-10-2021 de 03:00 a 03:30

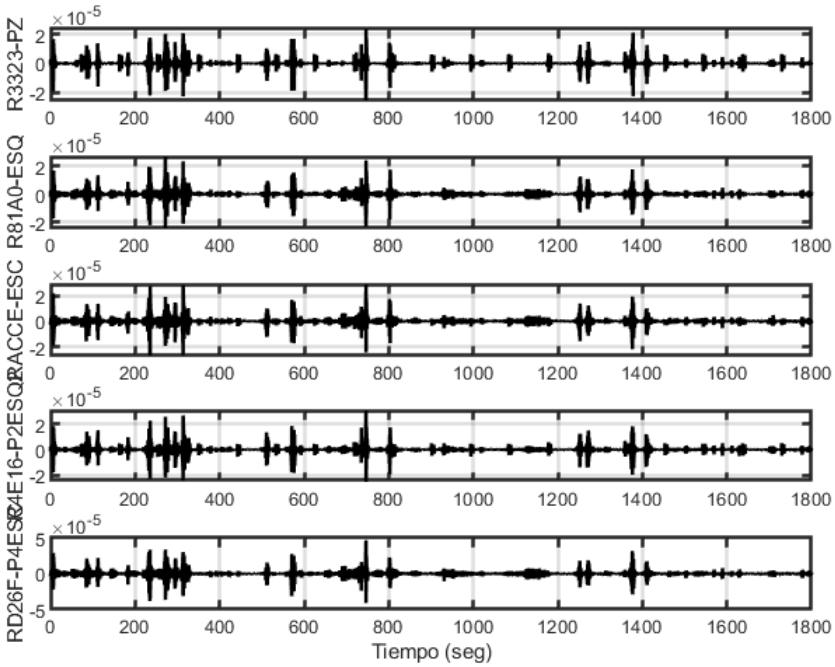

Figura B.6: Registro de sensores el día 21-10-2021 de 04:00 a 04:30

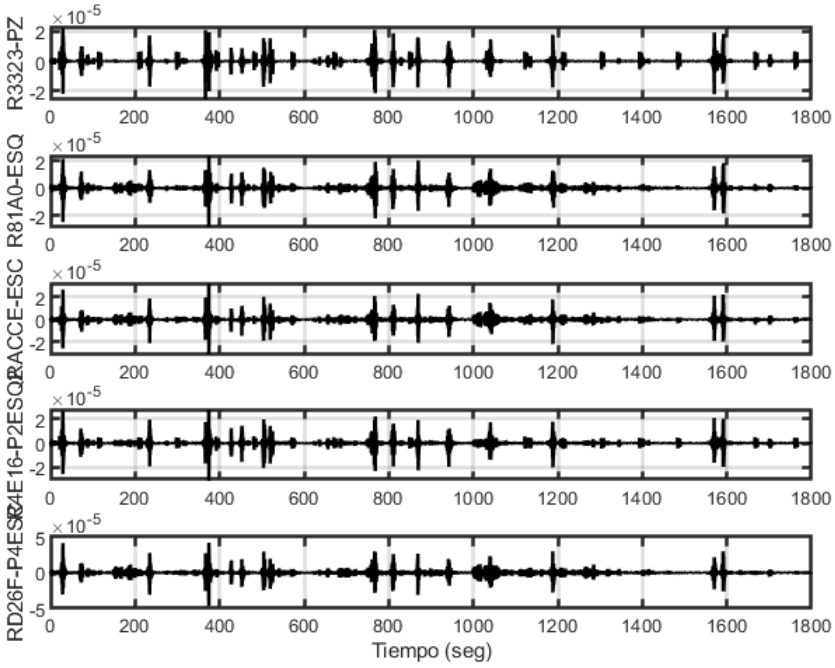

Figura B.7: Registro de sensores el día 21-10-2021 de 04:30 a 05:00

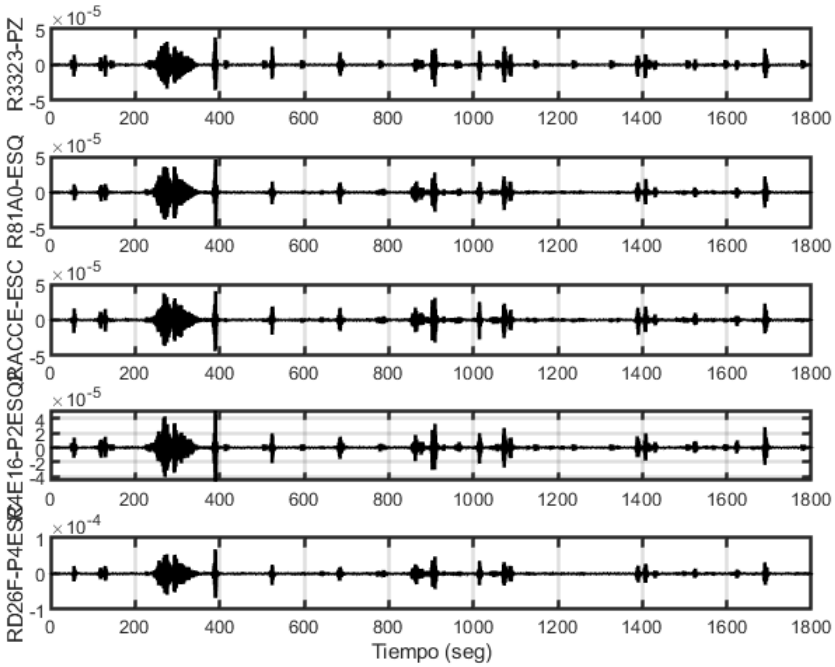

Figura B.8: Registro de sensores el día 21-10-2021 de 05:00 a 05:30

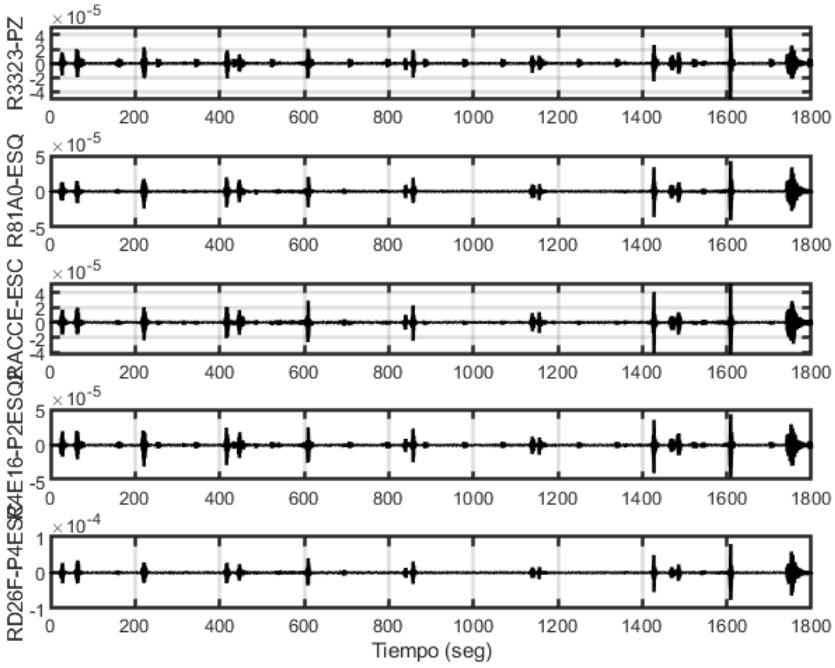

Figura B.9: Registro de sensores el día 21-10-2021 de 05:30 a 06:00

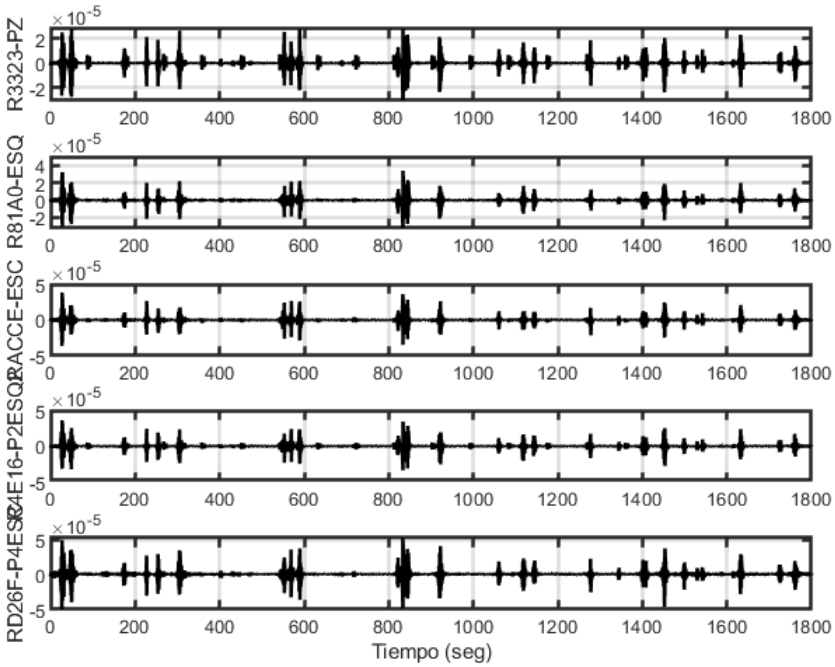

Figura B.10: Registro de sensores el día 21-10-2021 de 06:00 a 06:30

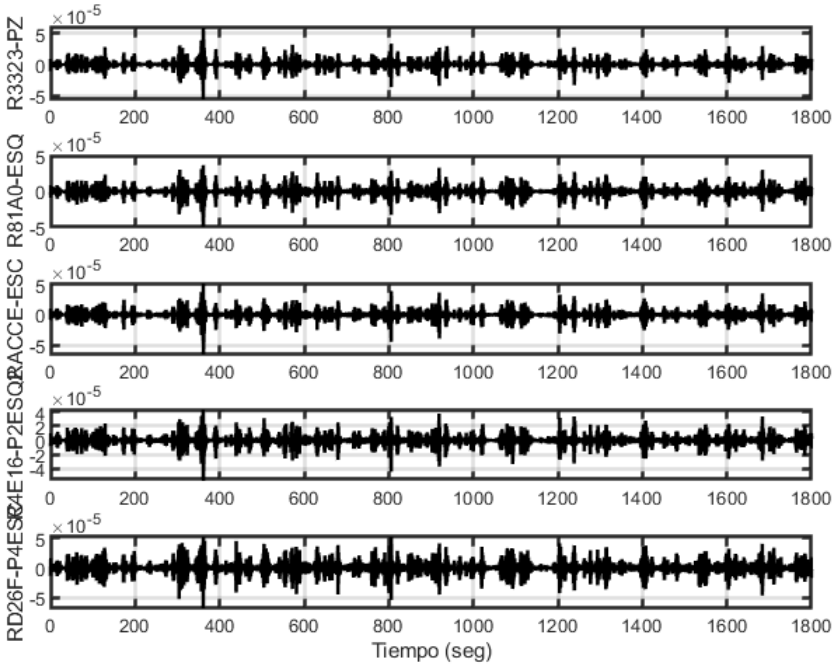

Figura B.11: Registro de sensores el día 21-10-2021 de 11:00 a 11:30

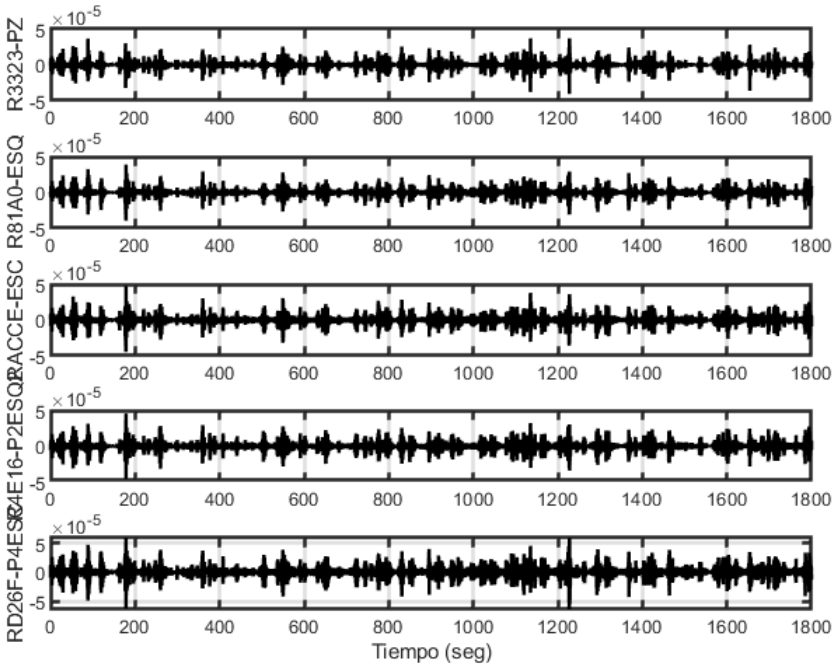

Figura B.12: Registro de sensores el día 21-10-2021 de 11:30 a 12:00

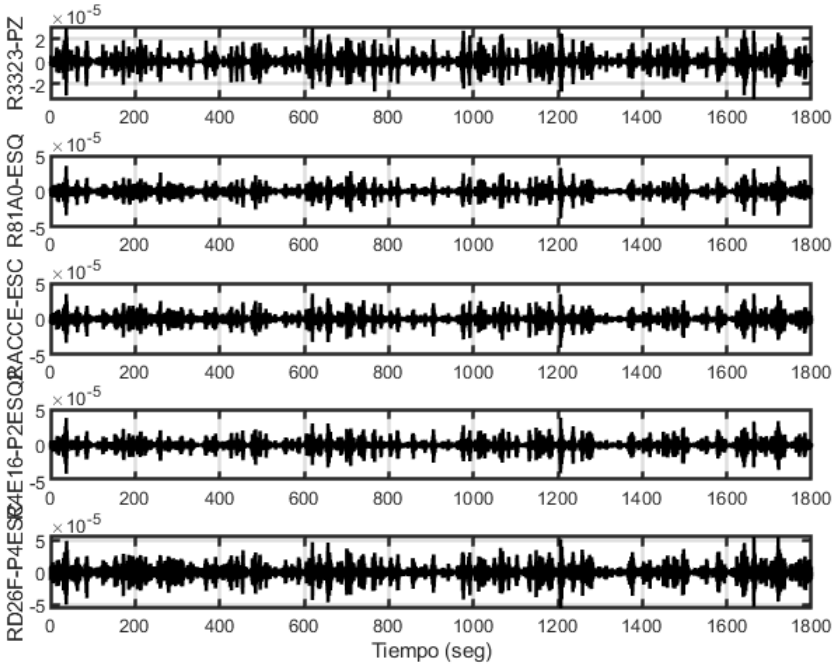

Figura B.13: Registro de sensores el día 21-10-2021 de 12:00 a 12:30

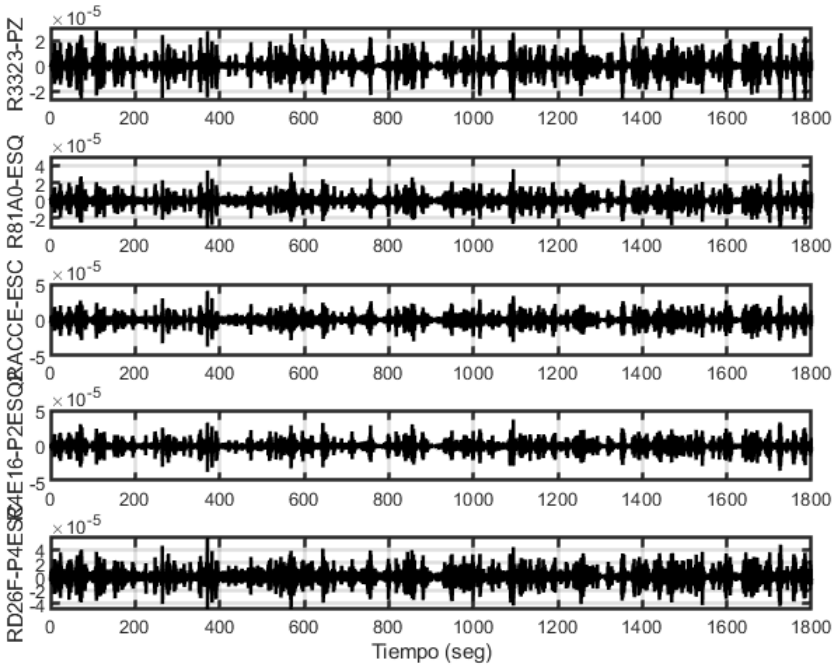

Figura B.14: Registro de sensores el día 21-10-2021 de 12:30 a 13:00

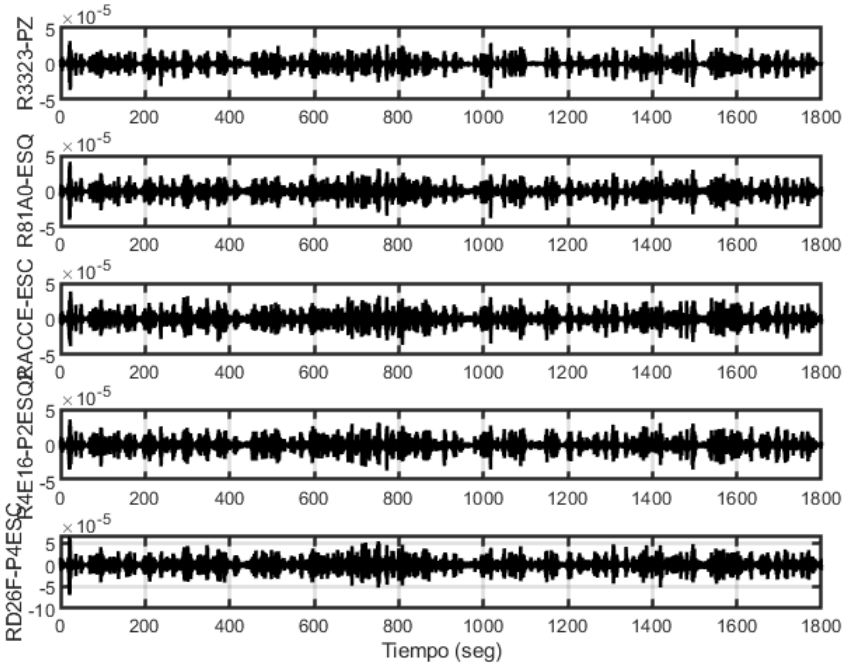

Figura B.15: Registro de sensores el día 21-10-2021 de 13:00 a 13:30

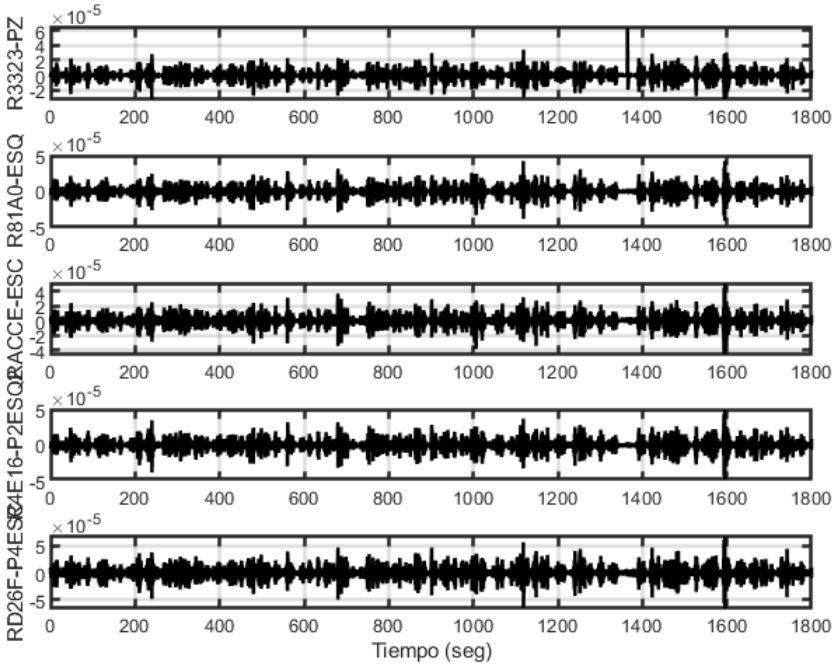

Figura B.16: Registro de sensores el día 21-10-2021 de 13:30 a 14:00

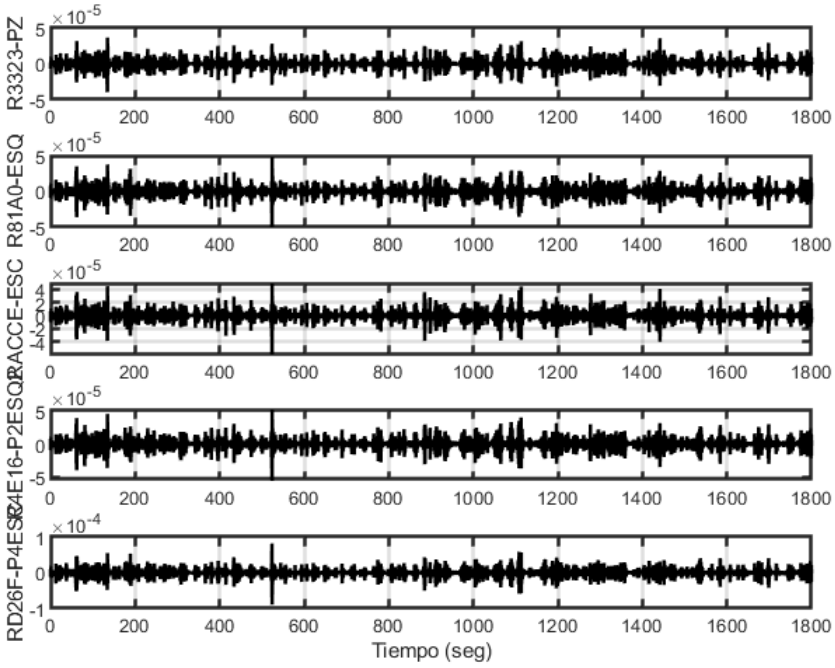

Figura B.17: Registro de sensores el día 21-10-2021 de 14:00 a 14:30

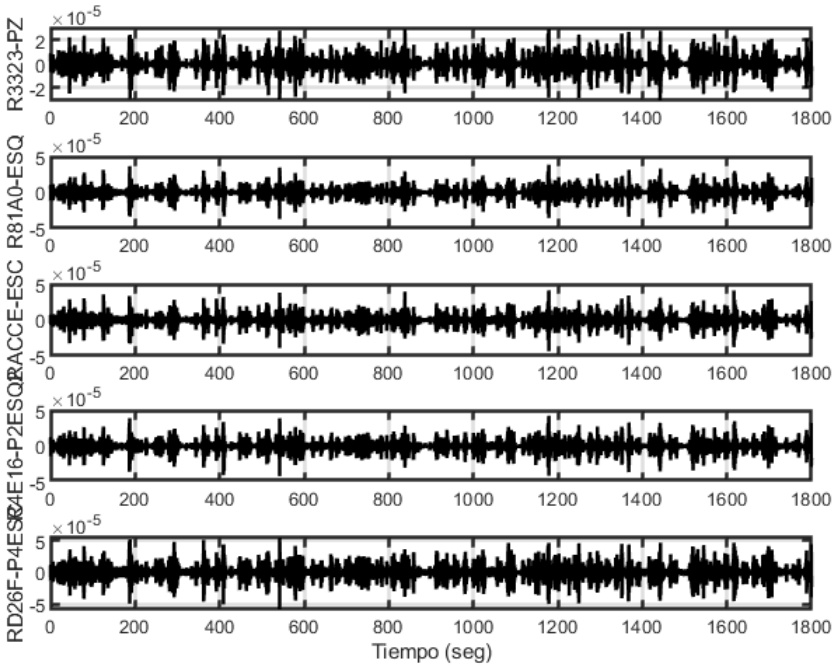

Figura B.18: Registro de sensores el día 21-10-2021 de 14:30 a 15:00

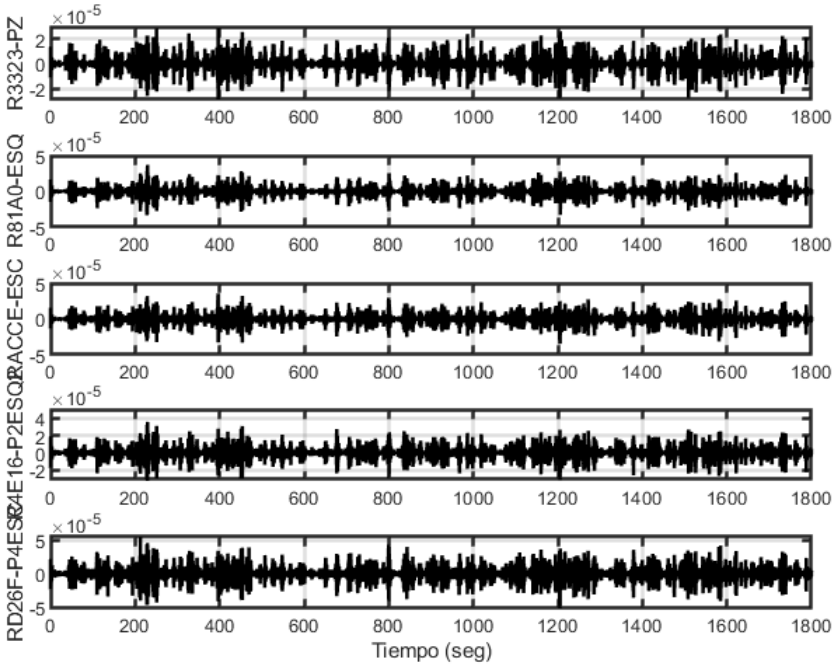

Figura B.19: Registro de sensores el día 21-10-2021 de 15:00 a 15:30

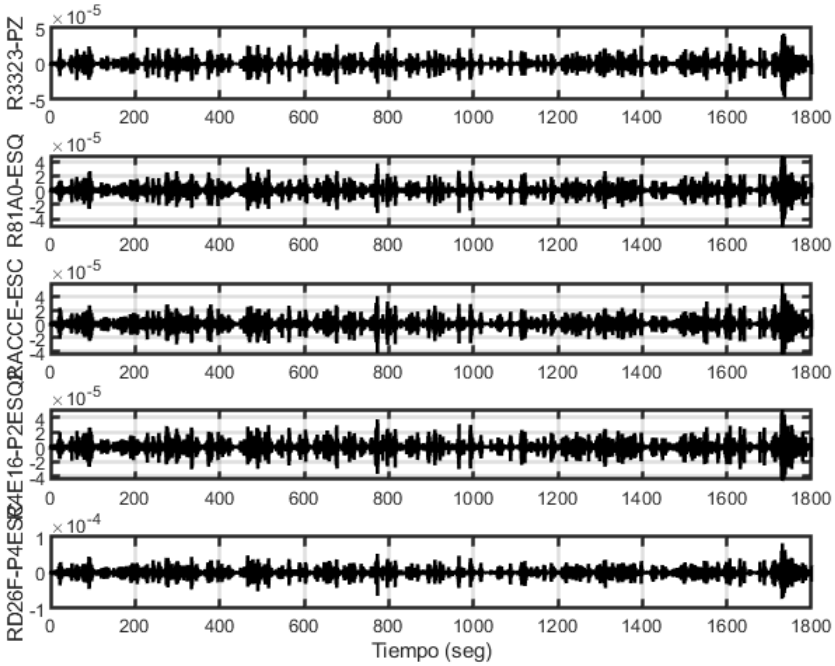

Figura B.20: Registro de sensores el día 21-10-2021 de 15:30 a 16:00

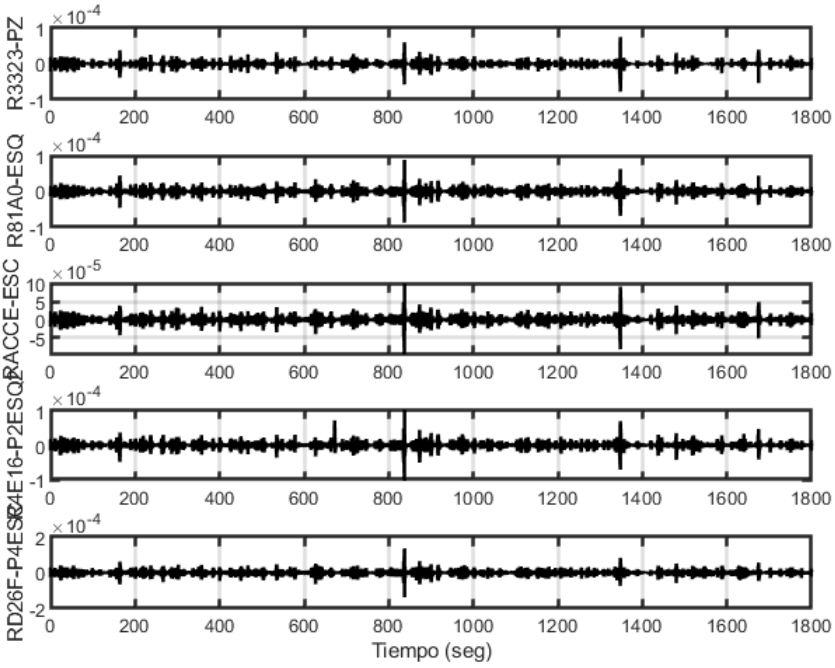

Figura B.21: Registro de sensores el día 21-10-2021 de 16:00 a 16:30

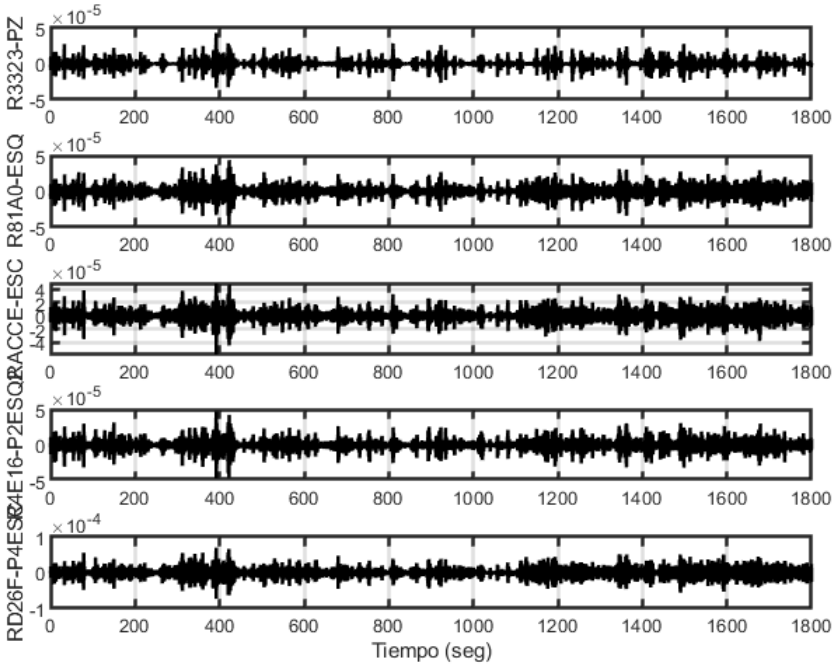

Figura B.22: Registro de sensores el día 21-10-2021 de 16:30 a 17:00

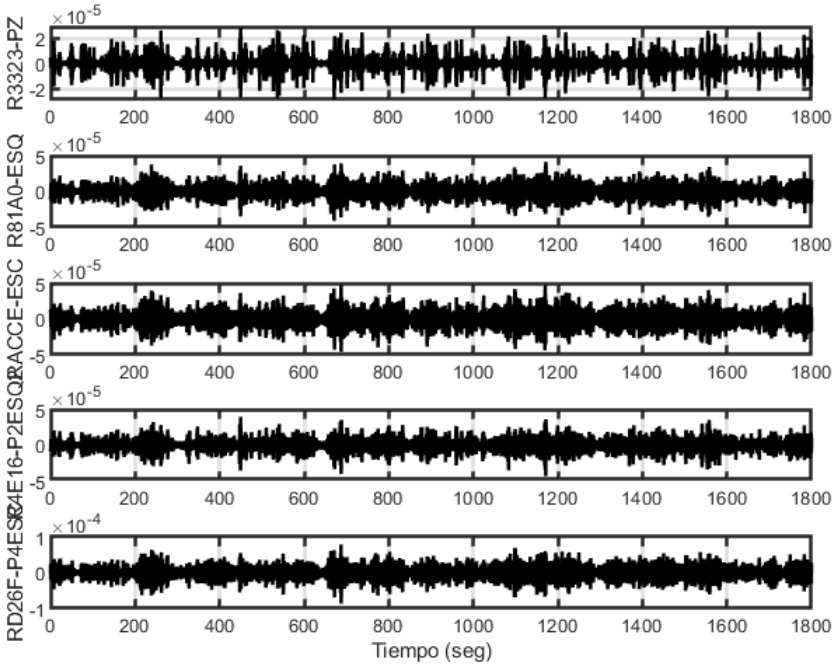

Figura B.23: Registro de sensores el día 21-10-2021 de 17:00 a 17:30

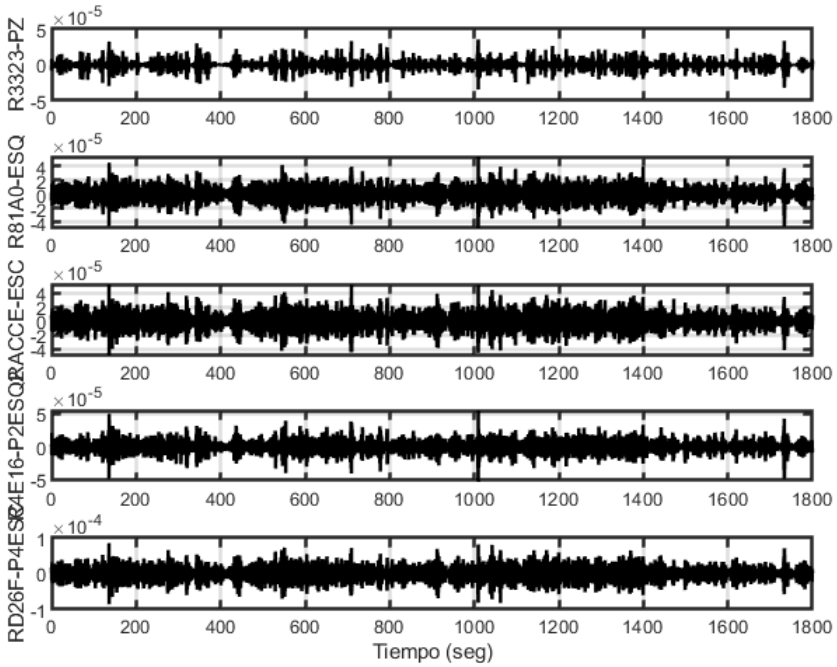

Figura B.24: Registro de sensores el día 21-10-2021 de 17:30 a 18:00

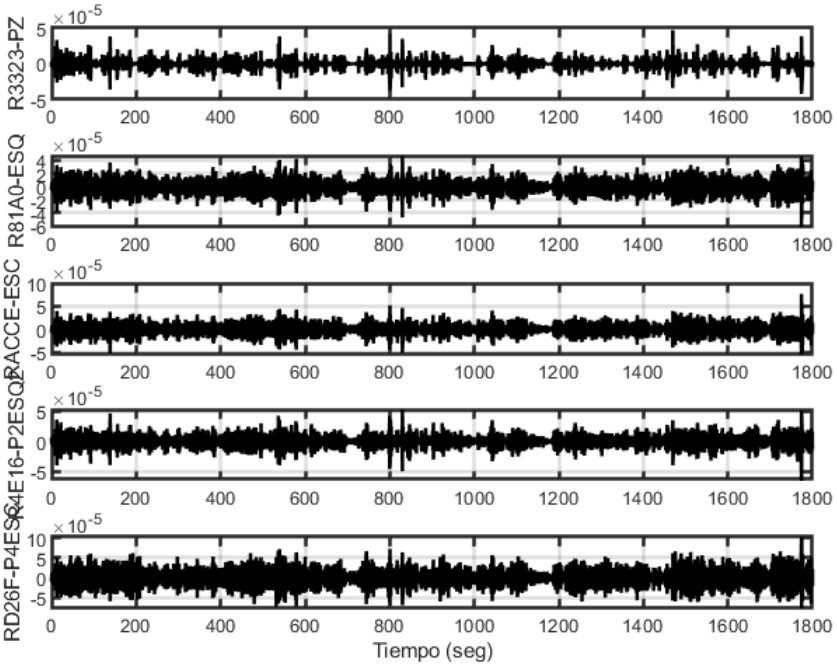

Figura B.25: Registro de sensores el día 21-10-2021 de 18:00 a 18:30

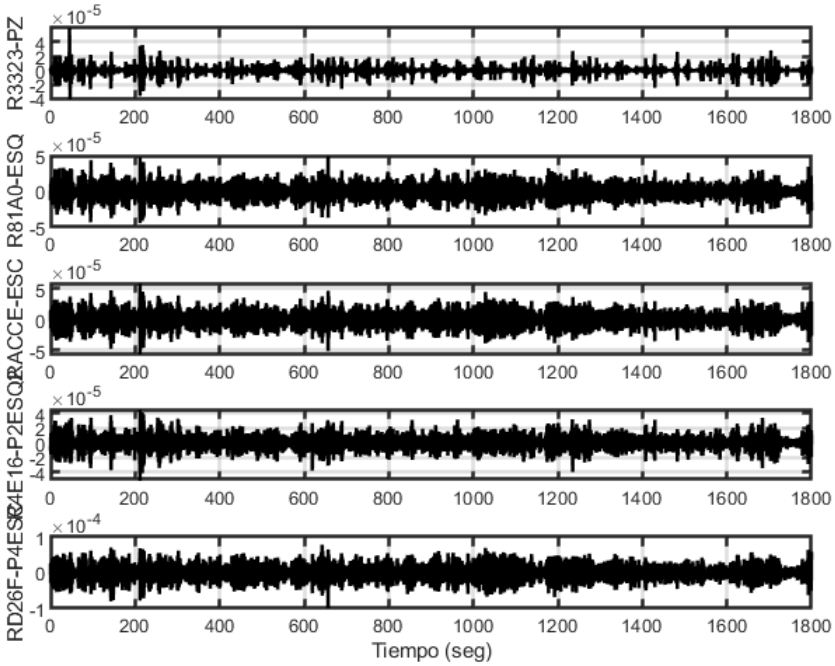

Figura B.26: Registro de sensores el día 21-10-2021 de 18:30 a 19:00

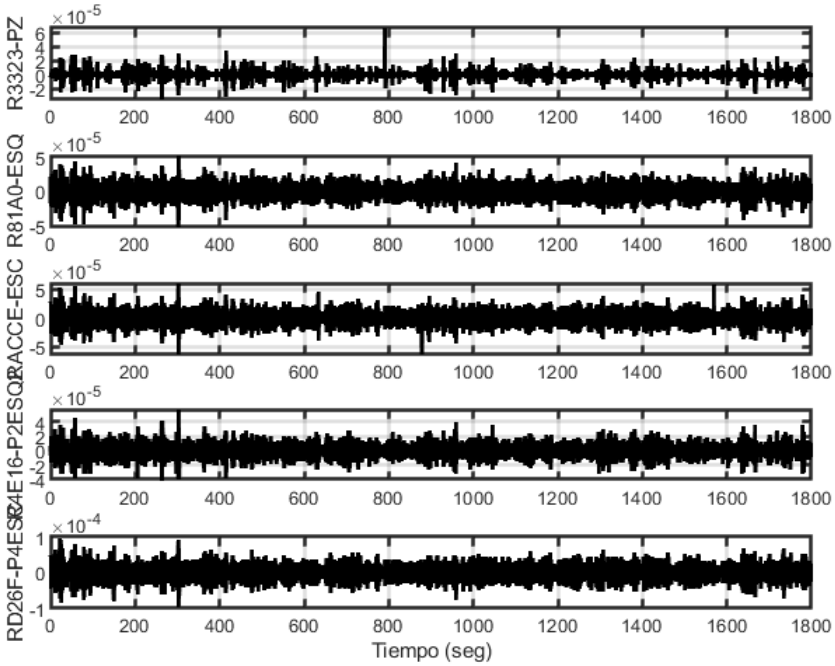

Figura B.27: Registro de sensores el día 21-10-2021 de 19:00 a 19:30

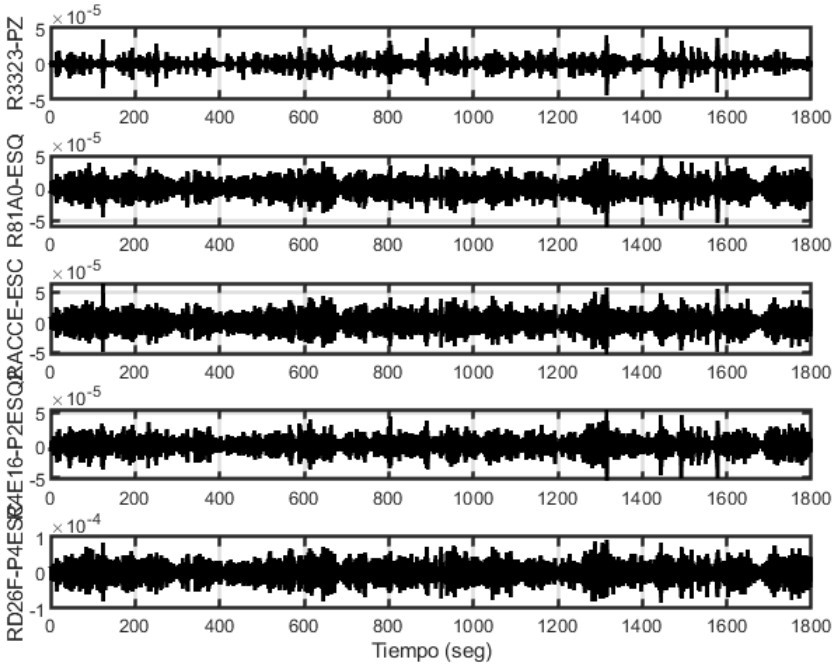

Figura B.28: Registro de sensores el día 21-10-2021 de 20:30 a 21:00

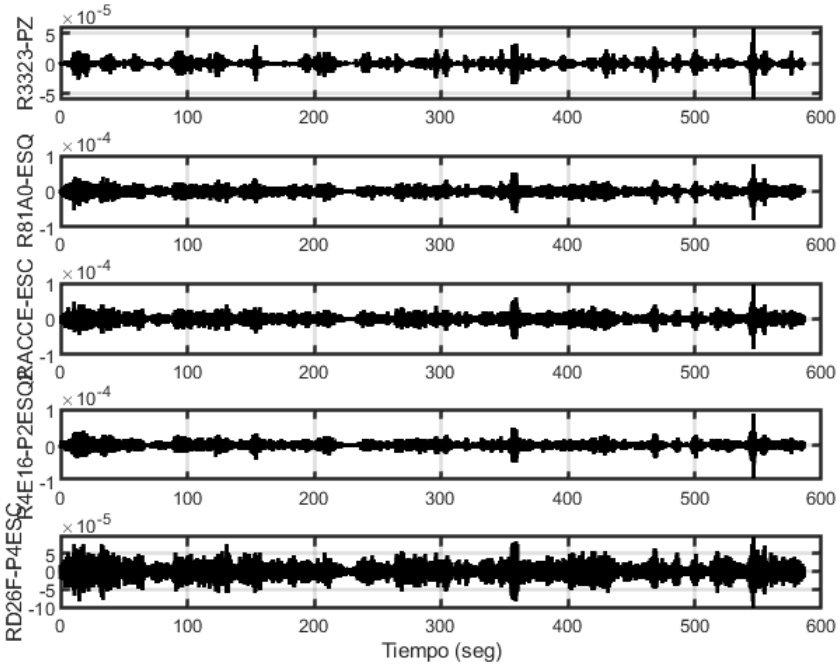

Figura B.29: Registro de sensores el día 21-10-2021 de 21:00 a 21:30

Anexo C

Análisis FDD

C.1. 21 de Octubre de 2021

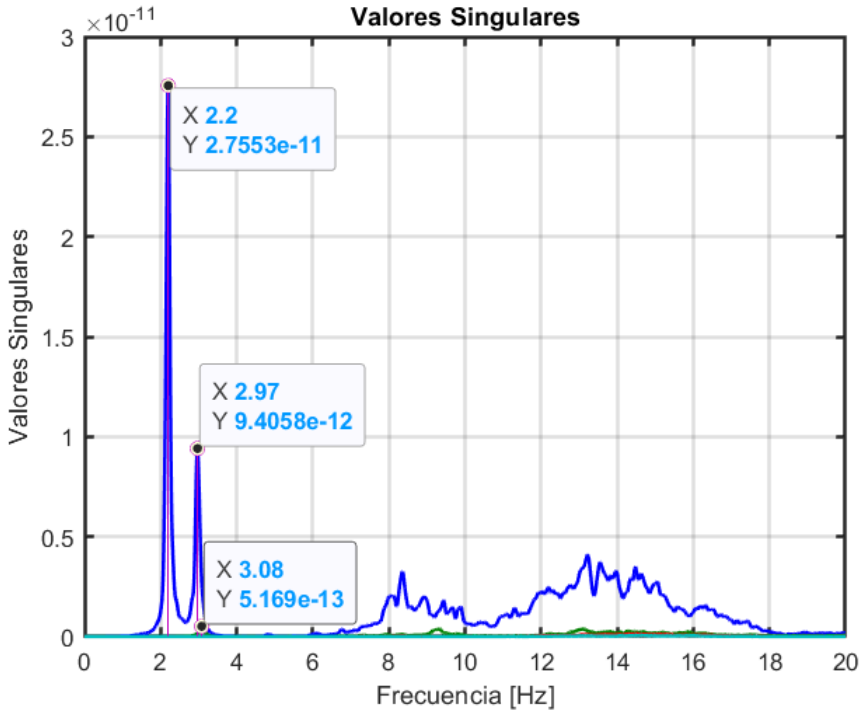

Figura C.1: Análisis FDD de las estacion entre las 1:00 y 1:30 UTC el día 21-10-2021

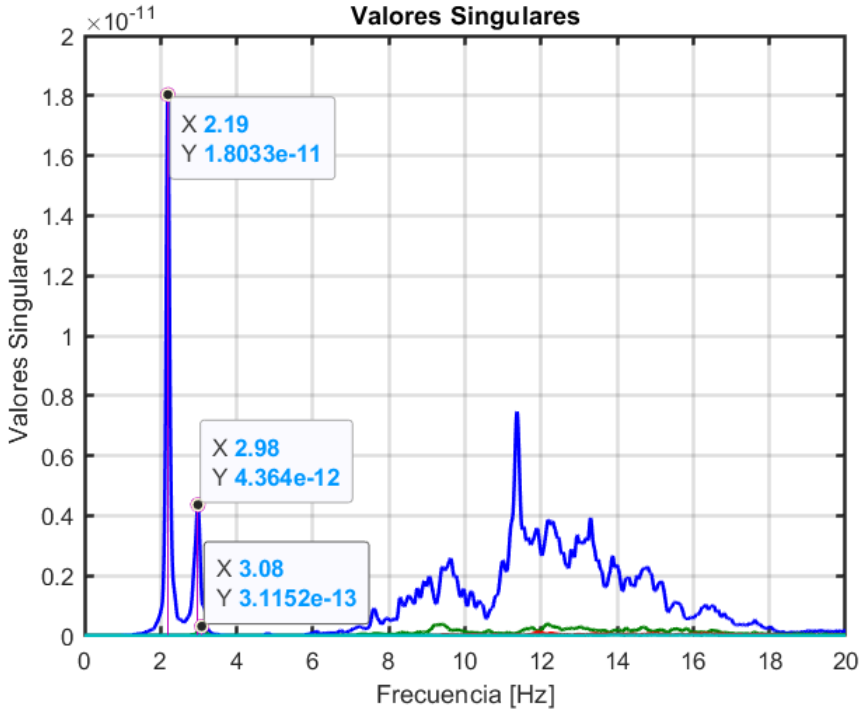

Figura C.2: Análisis FDD de las estacion entre las 1:30 y 2:00 UTC el día 21-10-2021

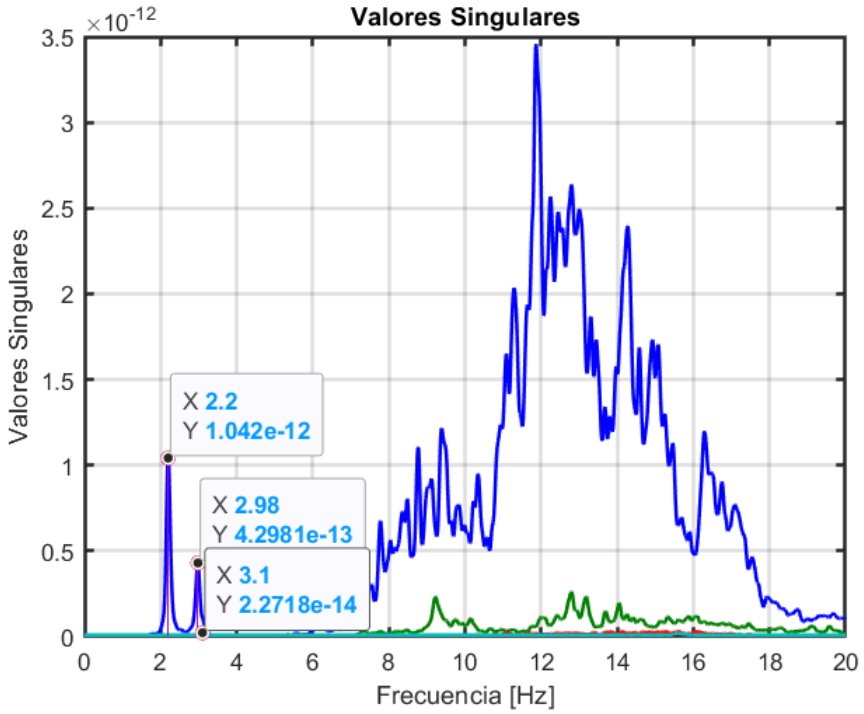

Figura C.3: Análisis FDD de las estacion entre las 2:00 y 2:30 UTC el día 21-10-2021

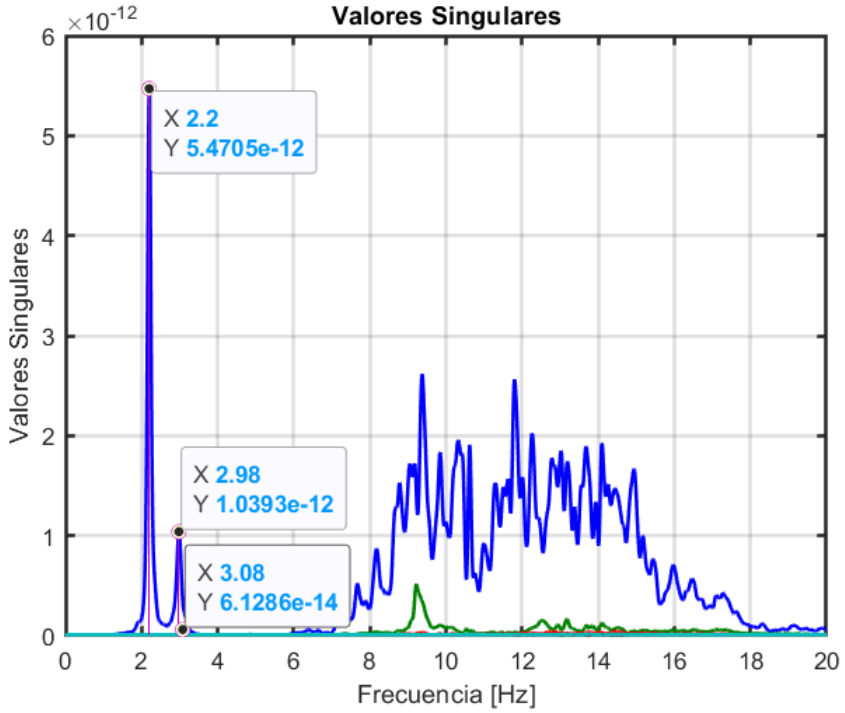

Figura C.4: Análisis FDD de las estacion entre las 2:30 y 3:00 UTC el día 21-10-2021

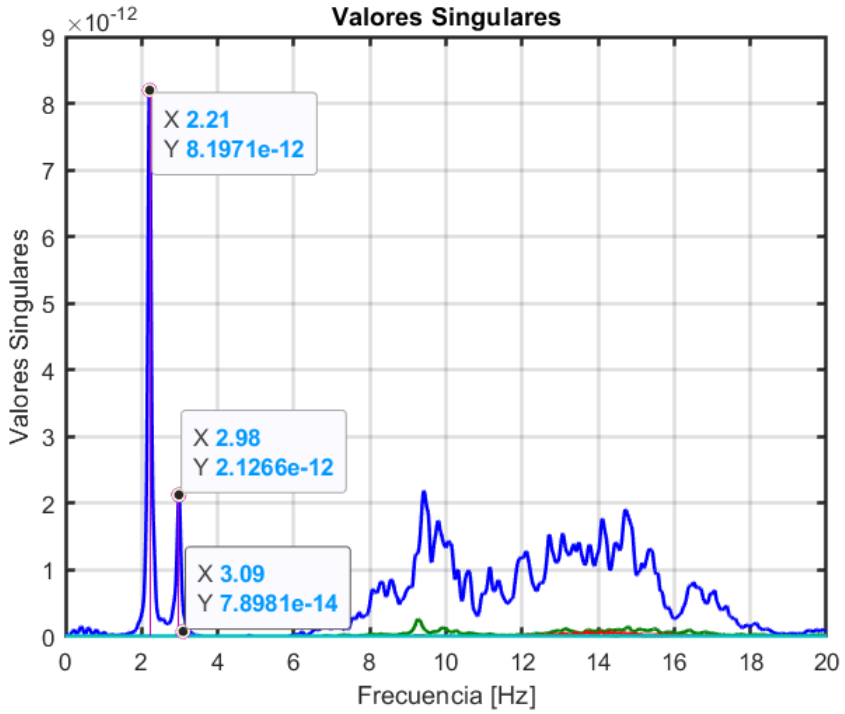

Figura C.5: Análisis FDD de las estacion entre las 3:00 y 3:30 UTC el día 21-10-2021

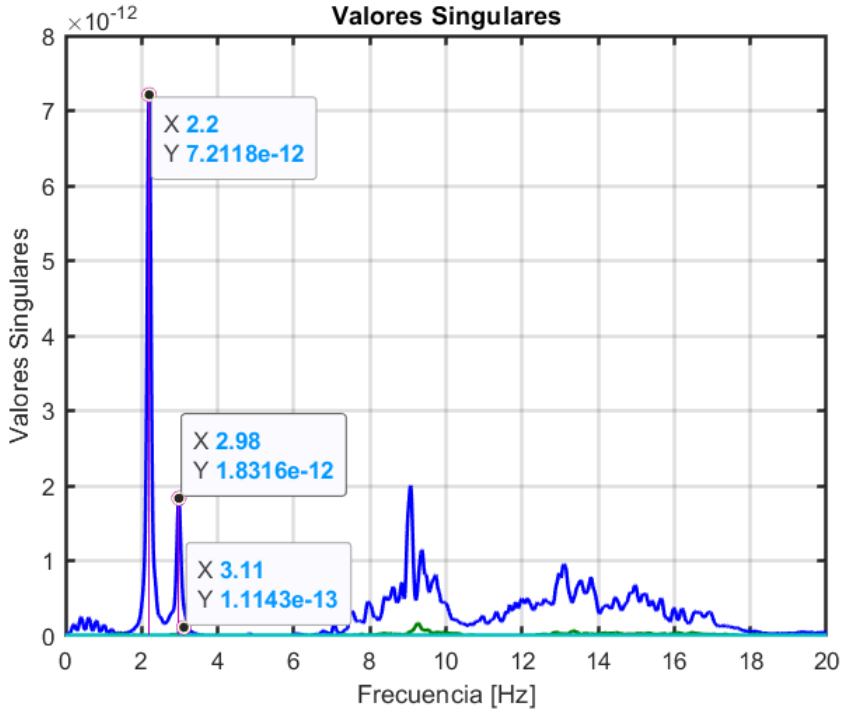

Figura C.6: Análisis FDD de las estacion entre las 4:00 y 4:30 UTC el día 21-10-2021

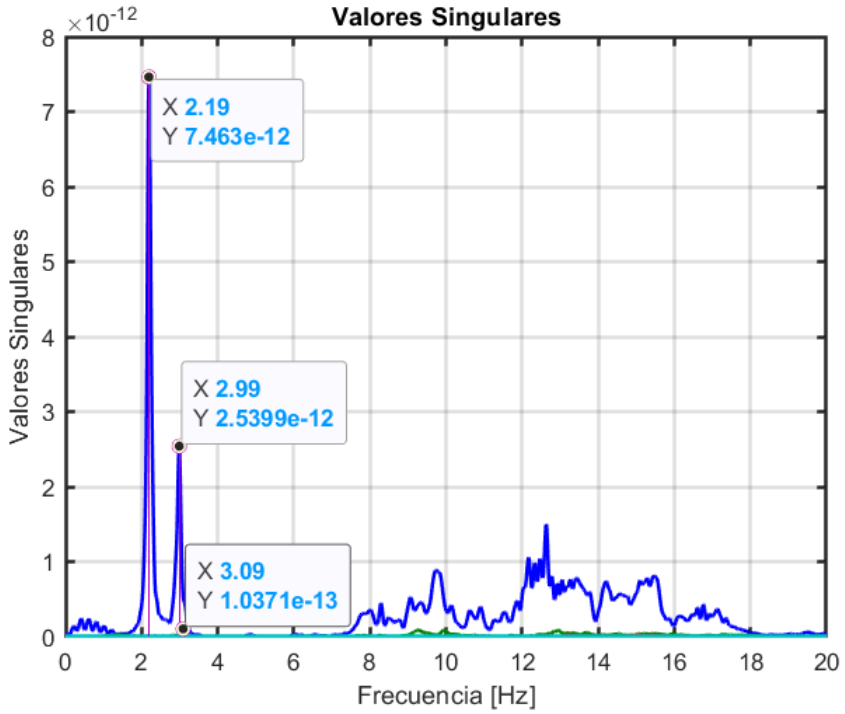

Figura C.7: Análisis FDD de las estacion entre las 4:30 y 5:00 UTC el día 21-10-2021

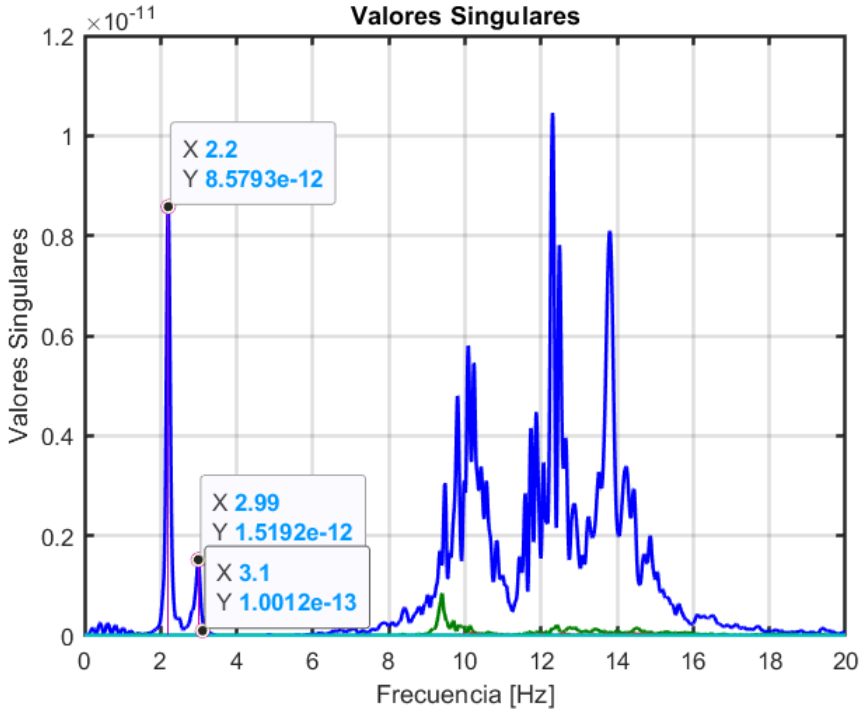

Figura C.8: Análisis FDD de las estacion entre las 5:00 y 5:30 UTC el día 21-10-2021

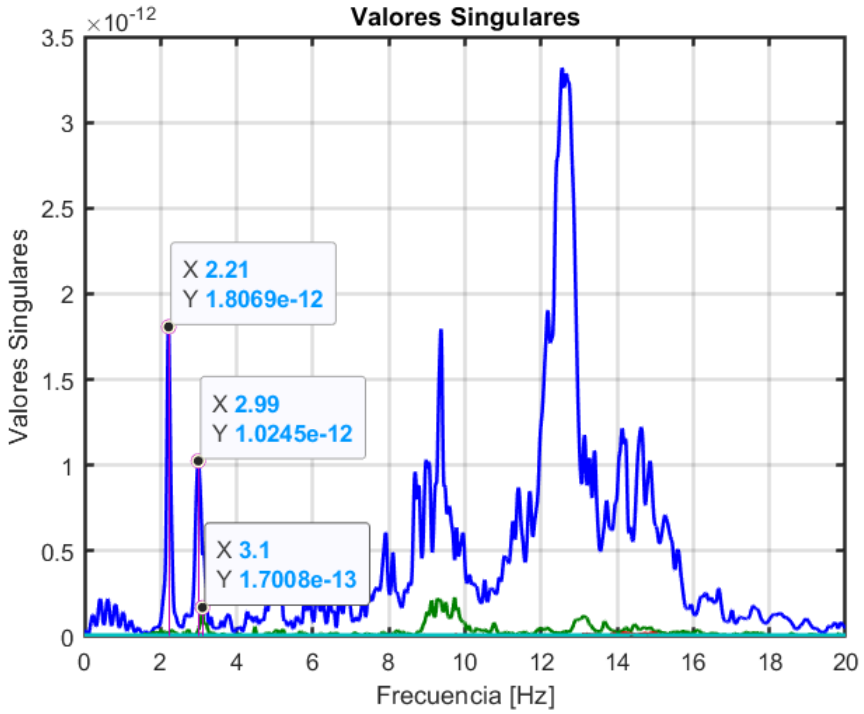

Figura C.9: Análisis FDD de las estacion entre las 5:30 y 6:00 UTC el día 21-10-2021

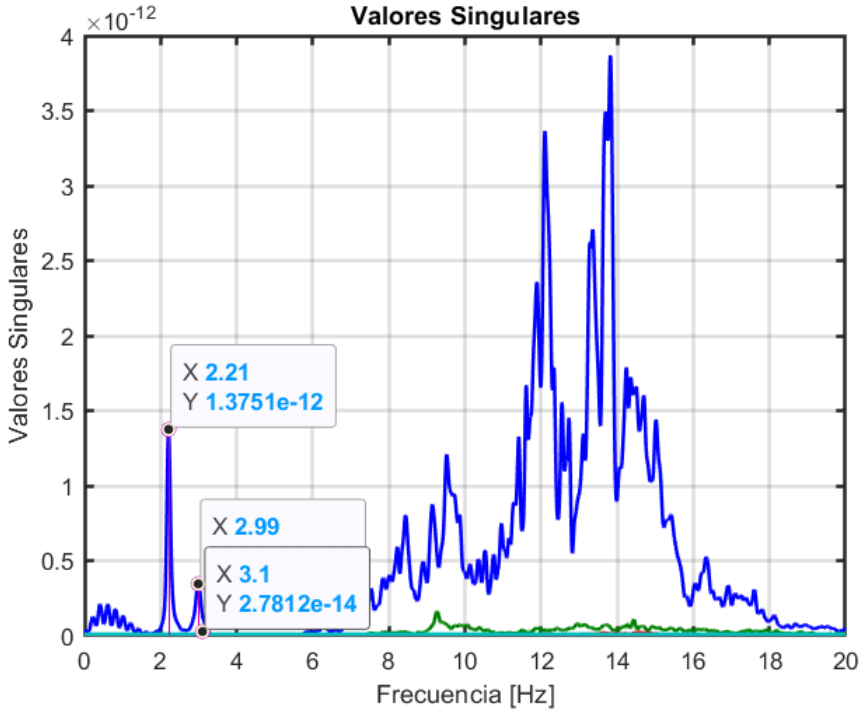

Figura C.10: Análisis FDD de las estacion entre las 6:00 y 6:30 UTC el día 21-10-2021

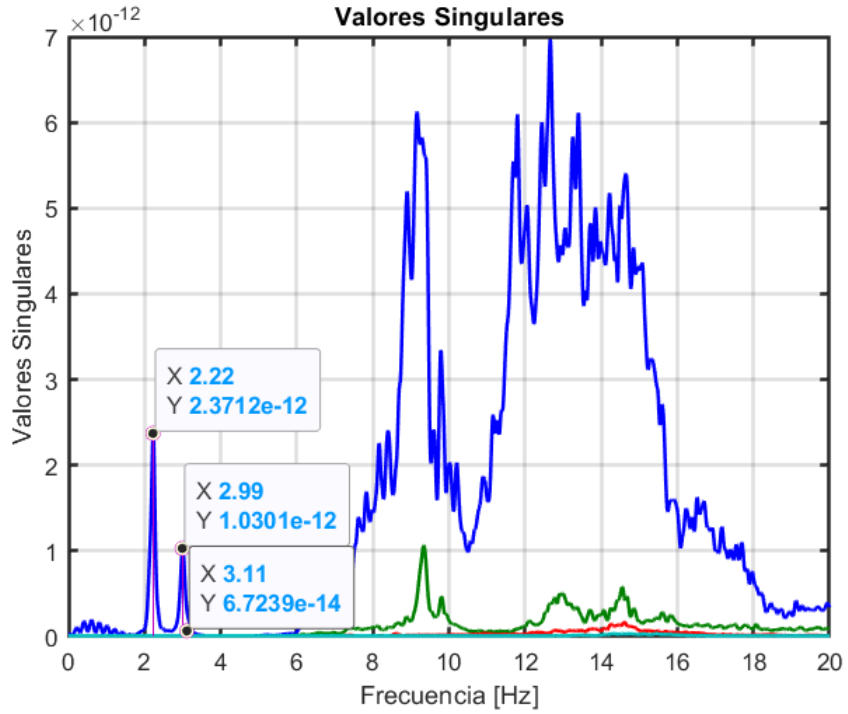

Figura C.11: Análisis FDD de las estacion entre las 11:00 y 11:30 UTC el día 21-10-2021

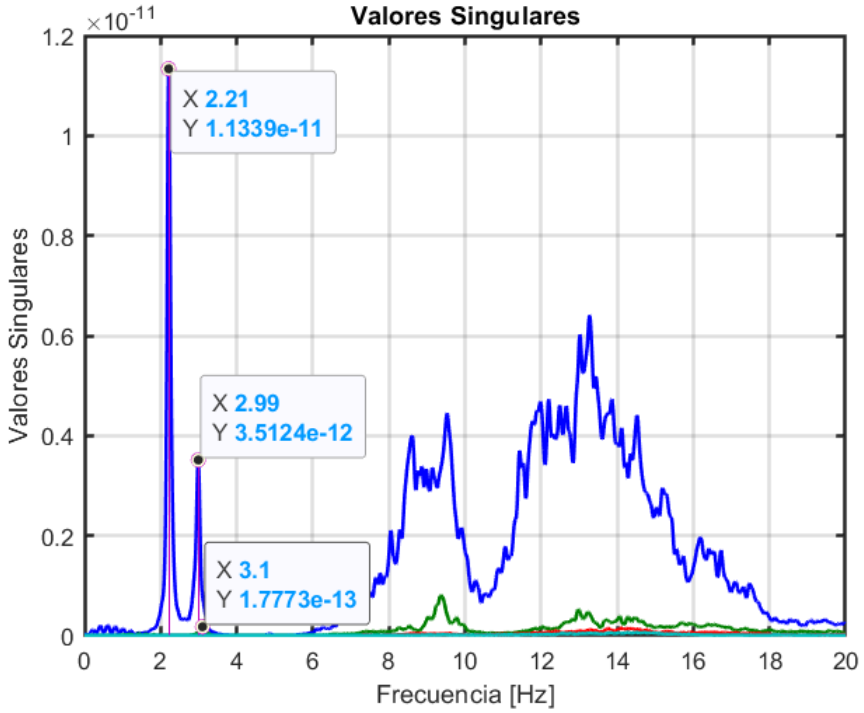

Figura C.12: Análisis FDD de las estacion entre las 11:30 y 12:00 UTC el día 21-10-2021

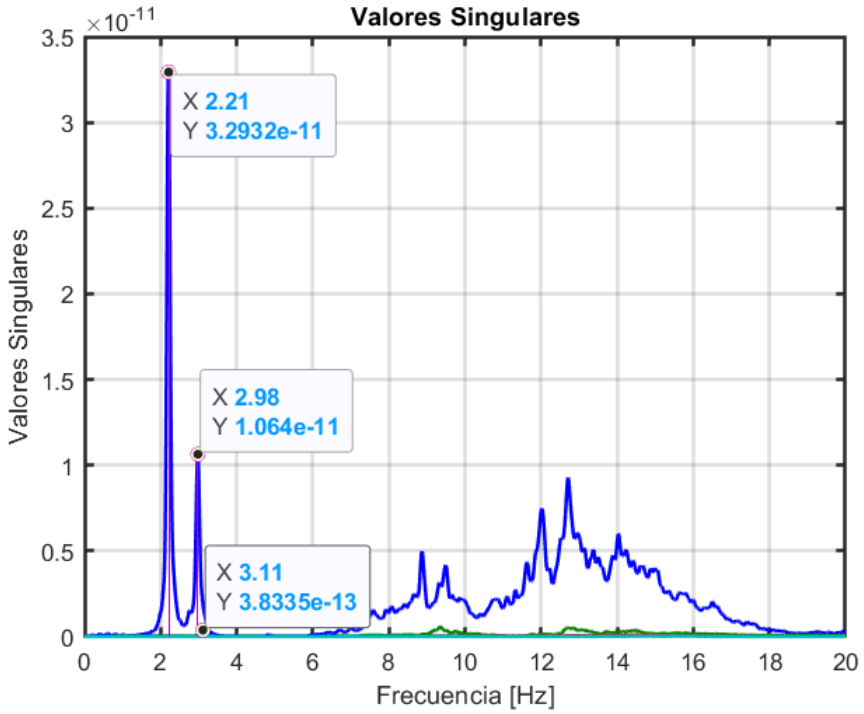

Figura C.13: Análisis FDD de las estacion entre las 12:00 y 12:30 UTC el día 21-10-2021

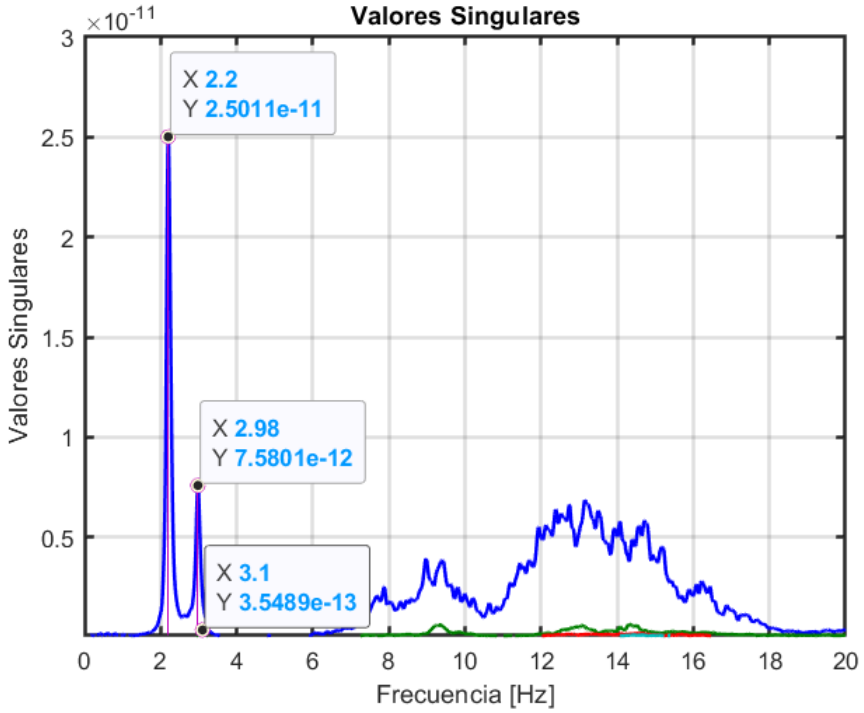

Figura C.14: Análisis FDD de las estacion entre las 12:30 y 13:00 UTC el día 21-10-2021

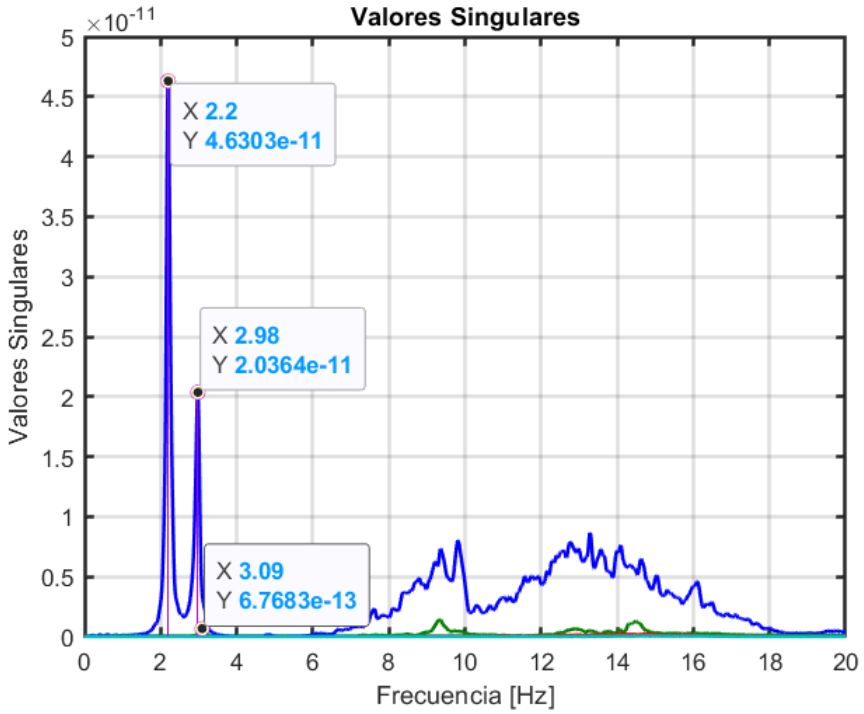

Figura C.15: Análisis FDD de las estacion entre las 13:00 y 13:30 UTC el día 21-10-2021

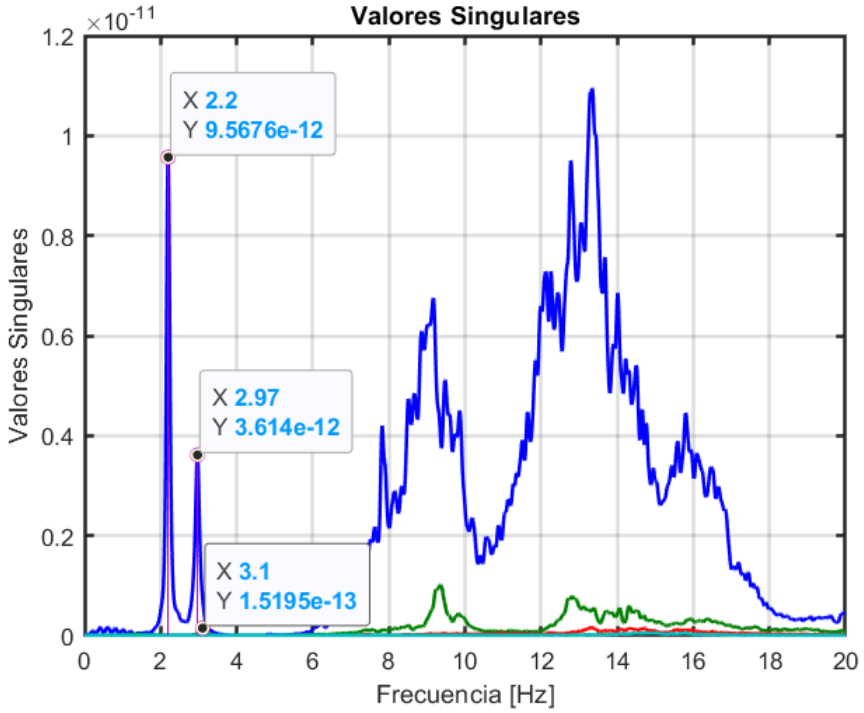

Figura C.16: Análisis FDD de las estacion entre las 13:30 y 14:00 UTC el día 21-10-2021

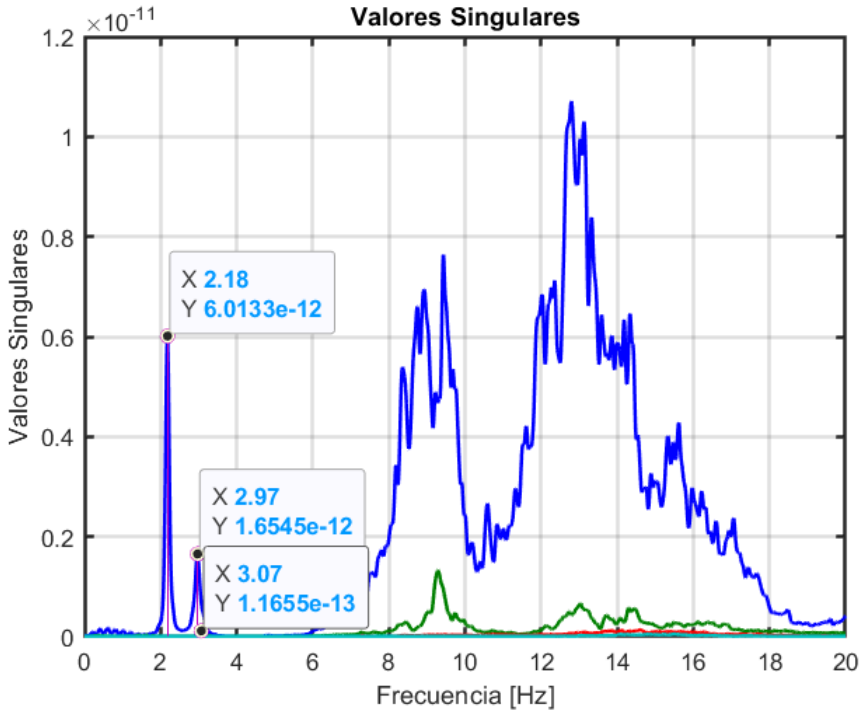

Figura C.17: Análisis FDD de las estacion entre las 14:00 y 14:30 UTC el día 21-10-2021

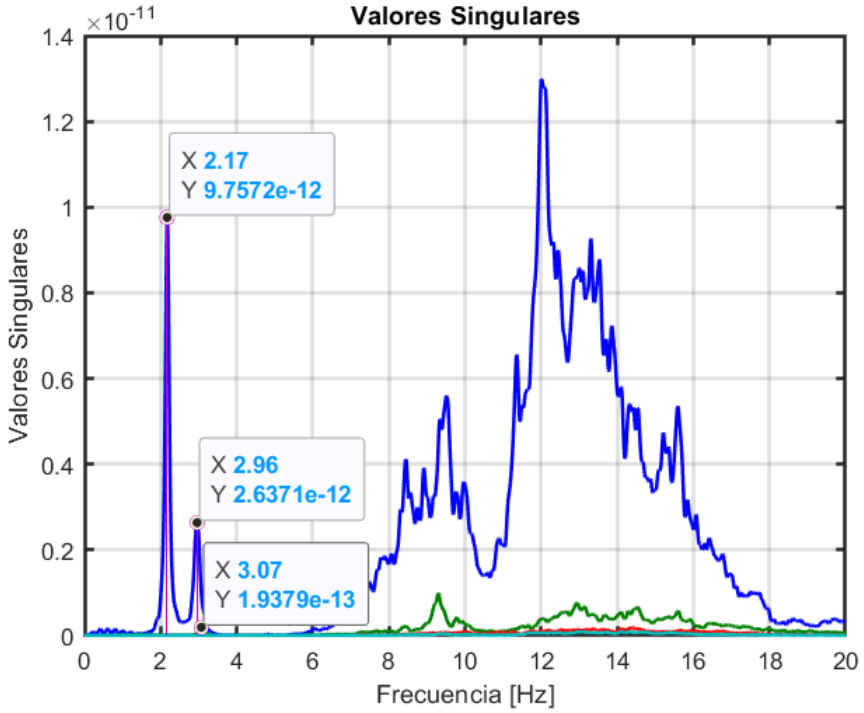

Figura C.18: Análisis FDD de las estacion entre las 14:30 y 15:00 UTC el día 21-10-2021

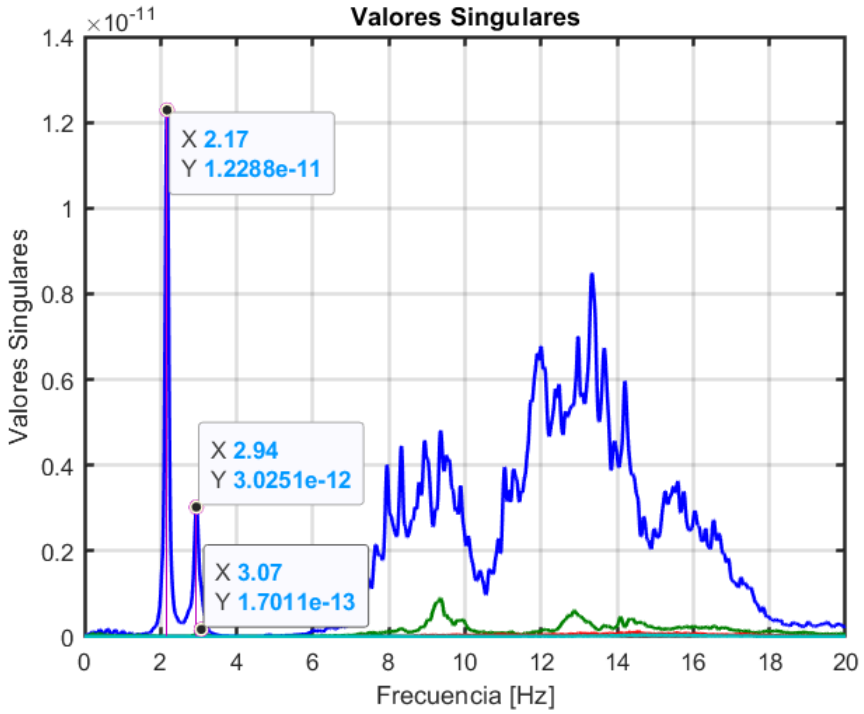

Figura C.19: Análisis FDD de las estacion entre las 15:00 y 15:30 UTC el día 21-10-2021

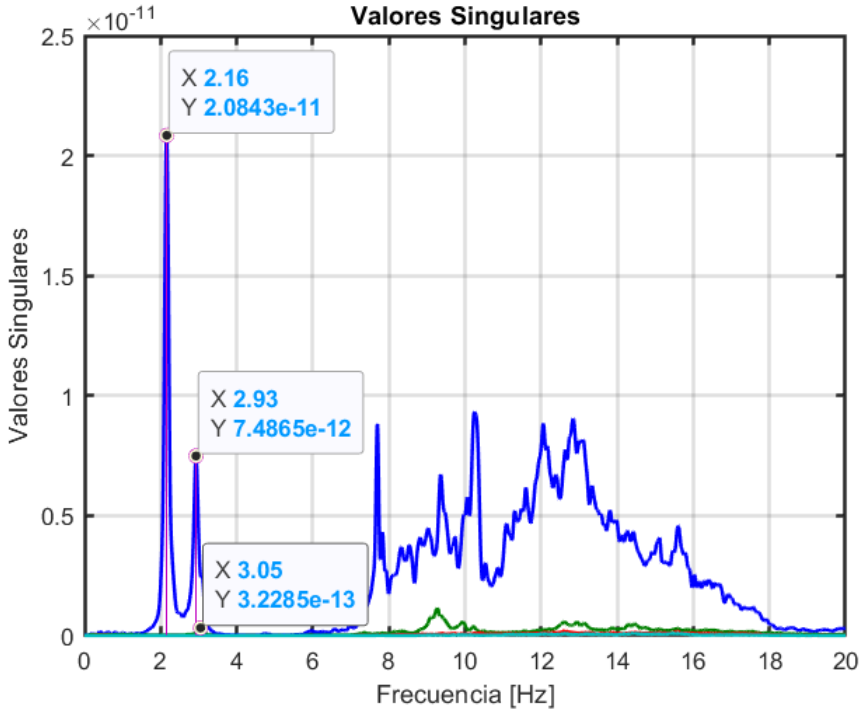

Figura C.20: Análisis FDD de las estacion entre las 15:30 y 16:00 UTC el día 21-10-2021

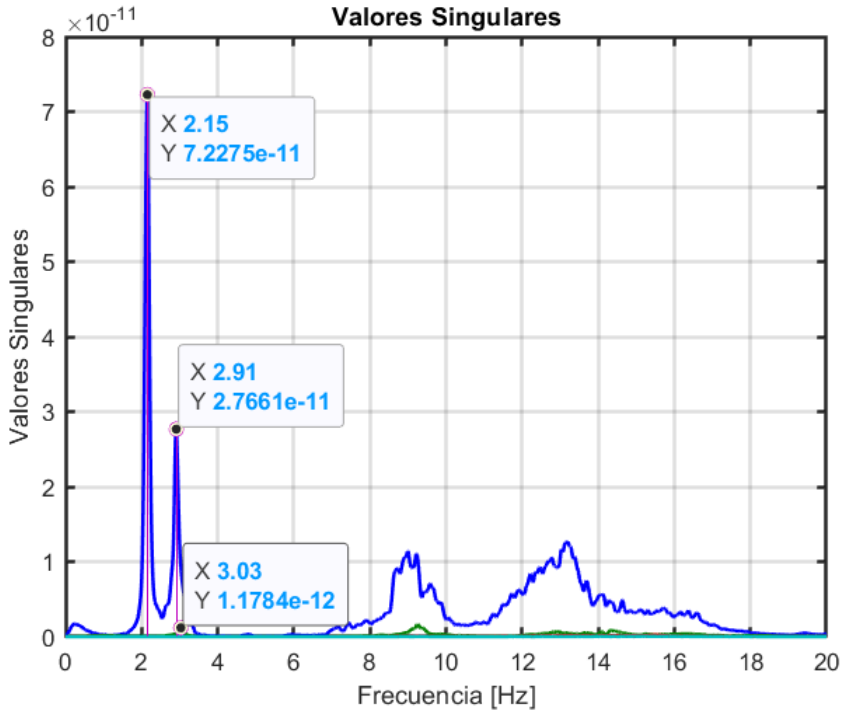

Figura C.21: Análisis FDD de las estacion entre las 16:00 y 16:30 UTC el día 21-10-2021

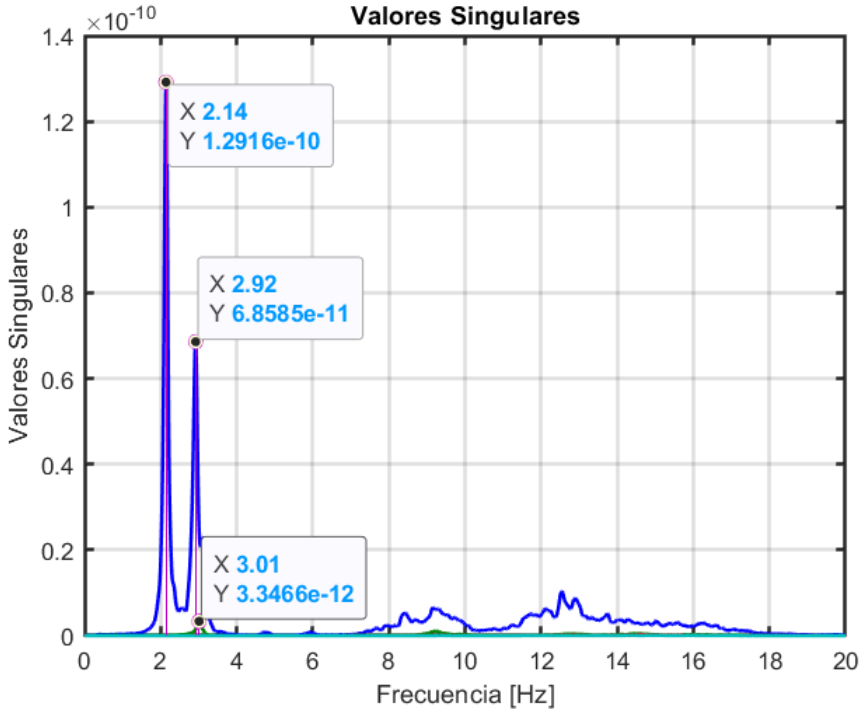

Figura C.22: Análisis FDD de las estacion entre las 16:30 y 17:00 UTC el día 21-10-2021

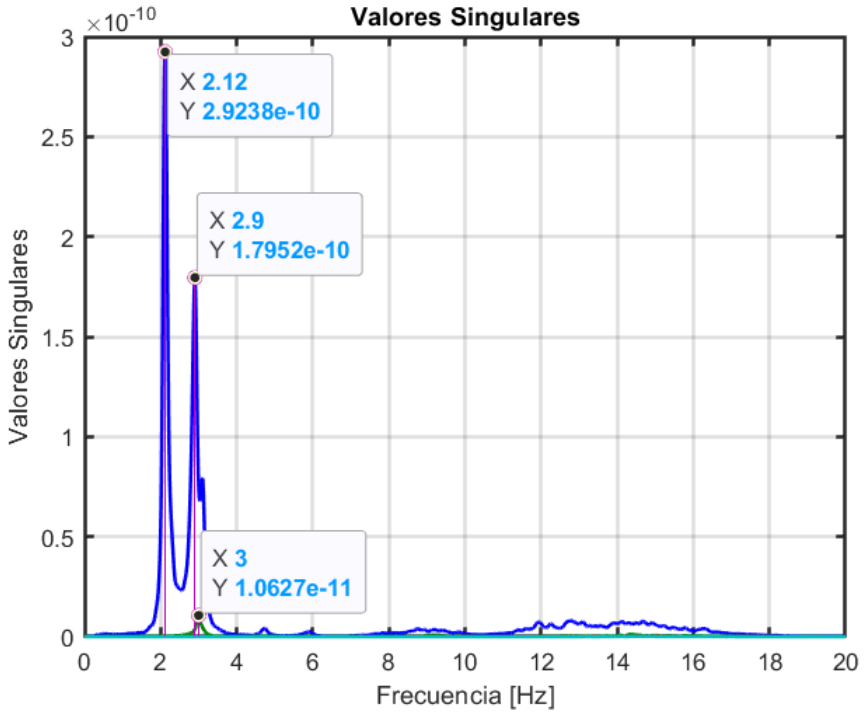

Figura C.23: Análisis FDD de las estacion entre las 17:00 y 17:30 UTC el día 21-10-2021

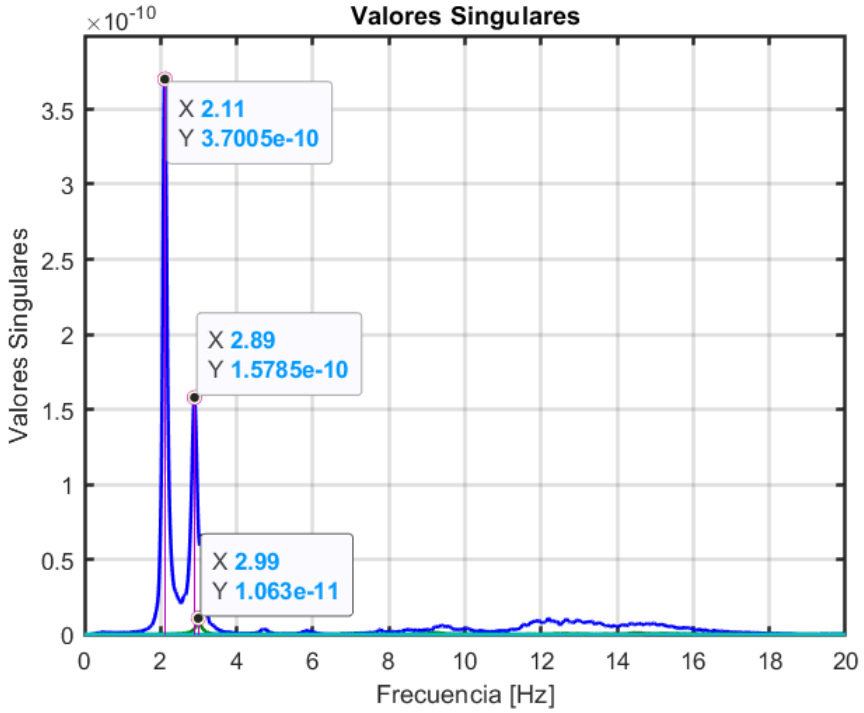

Figura C.24: Análisis FDD de las estacion entre las 17:30 y 18:00 UTC el día 21-10-2021

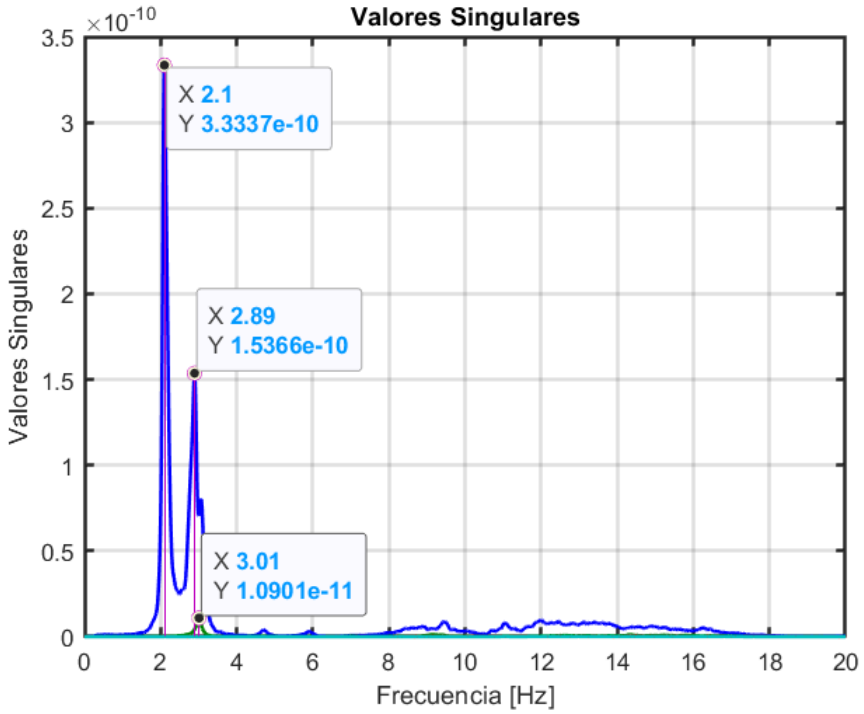

Figura C.25: Análisis FDD de las estacion entre las 18:00 y 18:30 UTC el día 21-10-2021

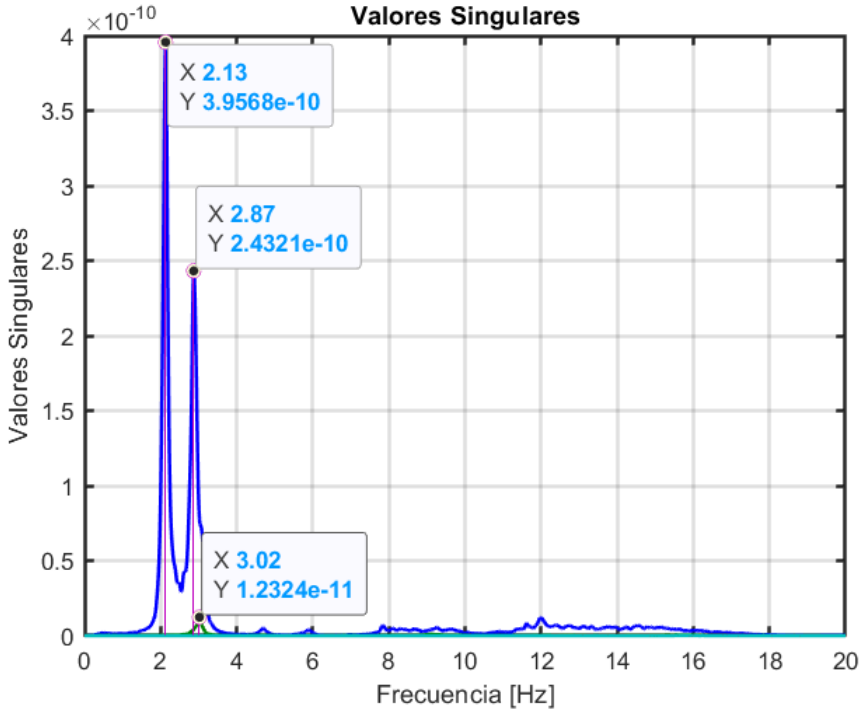

Figura C.26: Análisis FDD de las estacion entre las 18:30 y 19:00 UTC el día 21-10-2021

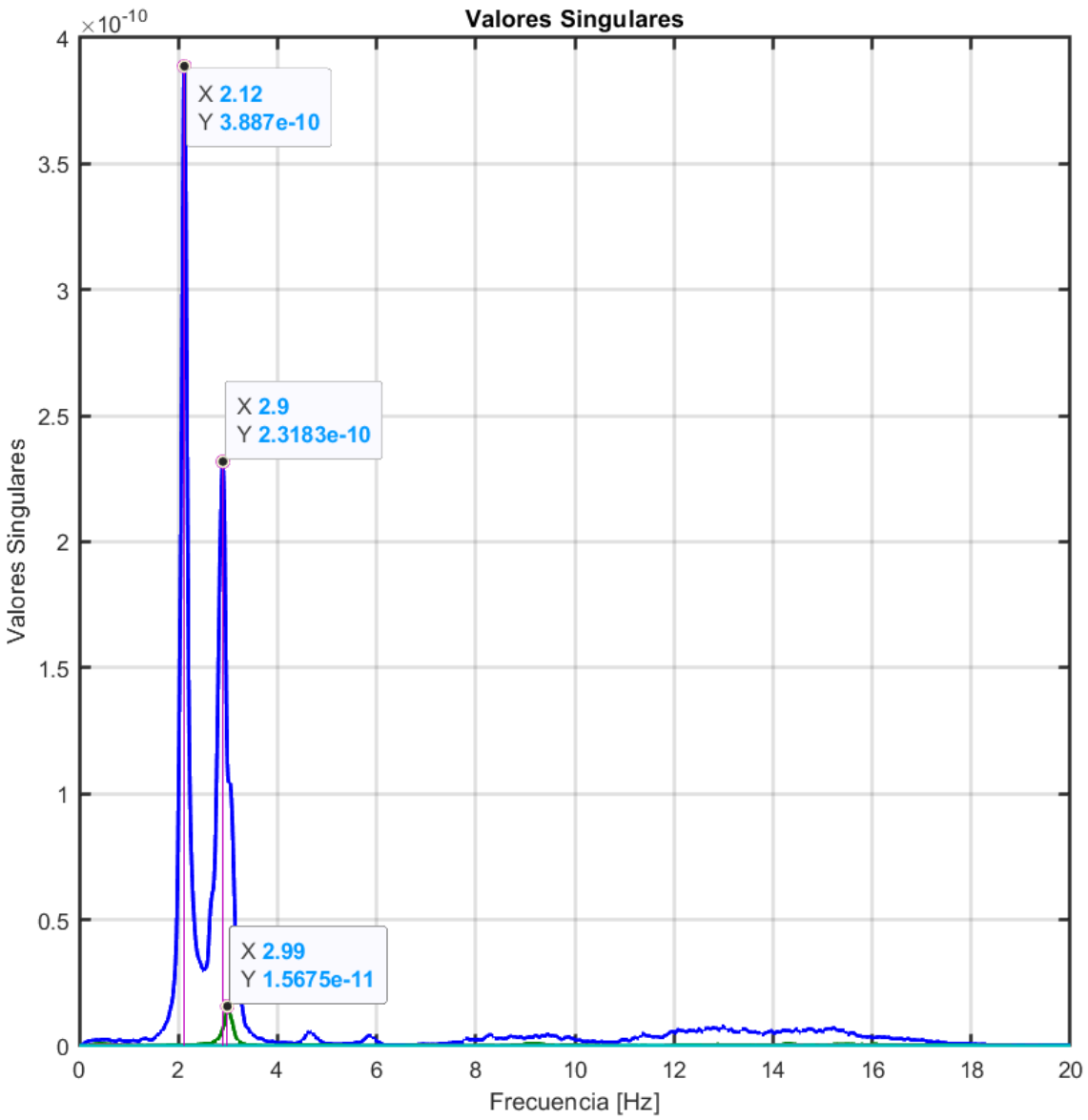

Figura C.27: Análisis FDD de las estacion entre las 19:00 y 19:30 UTC el día 21-10-2021

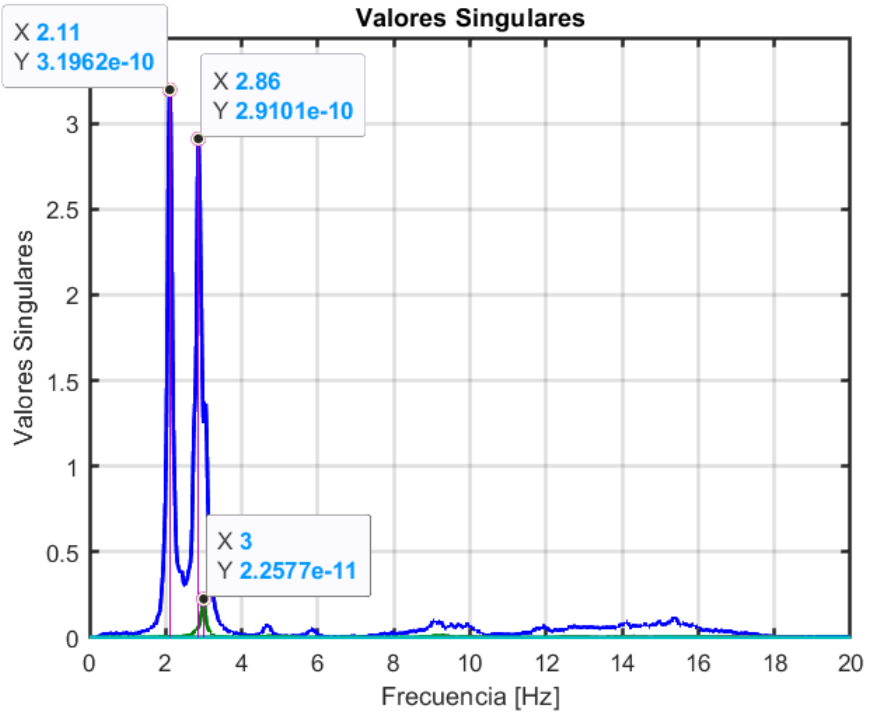

Figura C.28: Análisis FDD de las estacion entre las 20:30 y 21:00 UTC el día 21-10-2021

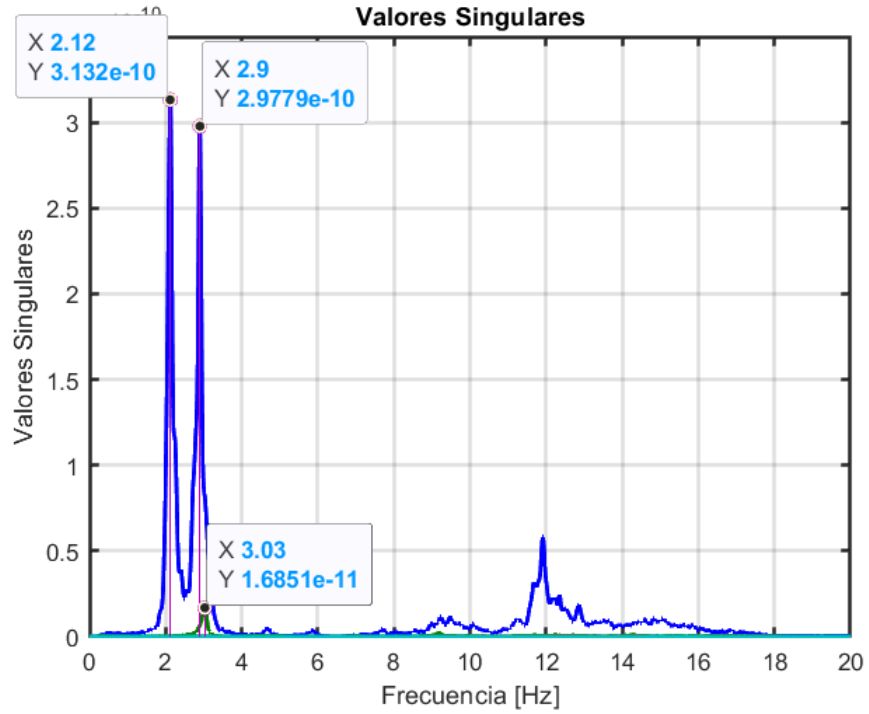

Figura C.29: Análisis FDD de las estacion entre las 21:00 y 21:30 UTC el día 21-10-2021