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

DIAGNÓSTICO DE FALLAS EN ENGRANAJES PLANETARIOS A PARTIR DE REDES NEURONALES ENTRENADAS CON MODELO FENOMENOLÓGICO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MECÁNICO

GUILLERMO NICOLÁS GALLARDO GONZÁLEZ

PROFESOR GUÍA:
EDUARDO ANDRÉS SALAMANCA HENRIQUEZ

MIEMBROS DE LA COMISIÓN:
VIVIANA ISABEL MERUANE NARANJO
ENRIQUE ANDRÉS LÓPEZ DROGUETT

SANTIAGO DE CHILE
2019

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: GUILLERMO NICOLÁS GALLARDO GONZÁLEZ
FECHA: NOVIEMBRE 2019
PROF. GUÍA: EDUARDO ANDRÉS SALAMANCA HENRIQUEZ

DIAGNÓSTICO DE FALLAS EN ENGRANAJES PLANETARIOS A PARTIR DE REDES NEURONALES ENTRENADAS CON MODELO FENOMENOLÓGICO

En la actualidad el mantenimiento de máquinas industriales juega un rol crítico en la organización de grandes y pequeñas plantas productivas, debido a que un problema de gran magnitud puede derivar en días o semanas de detención, pudiendo costar enormes sumas de dinero a las empresas. Para la aplicación de una de las metodologías de mantenimiento que más fuerza está tomando últimamente, las llamadas Redes Neuronales, se requiere de un gran número de datos en distintas condiciones de operación y con distintos modos de falla para cada uno de los equipos de los que se requiere hacer un modelo, lo cual resulta sumamente complejo y costoso de conseguir. Debido a esto surge la necesidad de obtener datos para el entrenamiento de estos algoritmos de una manera más fácil.

El objetivo general es la generación de un Algoritmo de Aprendizaje Profundo entrenado con datos simulados de un modelo fenomenológico. Este algoritmo busca identificar las fallas por grieta en base de dientes de los engranajes con distintos grados de severidad, en la corona, el sol y los planetas. Para conseguir esto el trabajo se divide en tres partes principales: En la primera etapa se creo un programa en el software *MATLAB* para la simulación de vibraciones mecánicas de un reductor planetario, posteriormente se realizaron mediciones experimentales en reductores planetarios utilizando un banco de pruebas y se procesaron los datos analizando sus *Transformadas de Fourier* luego de ser aplicados los filtros AR, MED, ARMED y la envolvente de la *Transformada de Fourier*, buscando características asociadas a las frecuencias de falla. La última etapa consistió en entrenar una red neuronal artificial con los datos simulados y que esta pueda clasificar los distintos tipos de mediciones.

El análisis de las *Transformadas de Fourier* indicaron que el mejor método para identificar fallas en el equipo es el filtro MED, y que la clasificación de distintos modos de fallas o grados de severidad resulta compleja debido a la poca correlación entre el caso de estudio y sus espectros. El algoritmo resultante puede identificar la presencia de fallas en el equipo con 86 % de exactitud para casos sin falla y con 97 % de exactitud para casos con falla.

Dedicado a mis padres por su apoyo incondicional, los amo.

Dedicado a Trini, por enseñarme muchas cosas que me han permitido ser más feliz.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Introducción

En la actualidad las industrias requieren de métodos eficientes para la transmisión de energía mecánica, área en donde las cajas reductoras con engranajes planetarios destacan en comparación con equipos tradicionales debido a su alta eficiencia, precisión, estructura compacta y pesos relativamente ligeros. Estos equipos consistes de un engranaje central llamado *sol*, un *anillo exterior* llamado *corona*, un elemento que lleva a los planetas llamado *portador* y un número determinado de *planetas* que depende de la configuración de cada equipo. En la **Figura 1** se puede observar un reductor planetario, en donde un conjunto de *planetas* giran en torno a un *sol* centrados en un eje del *portador* y en contacto con la *corona*. El movimiento relativo entre todos los elementos de este conjunto generan *vibraciones mecánicas*, cuyo análisis tiene variadas aristas debido a la gran cantidad de componentes que se relacionan dentro del equipo y permite diagnosticar el estado del reductor.

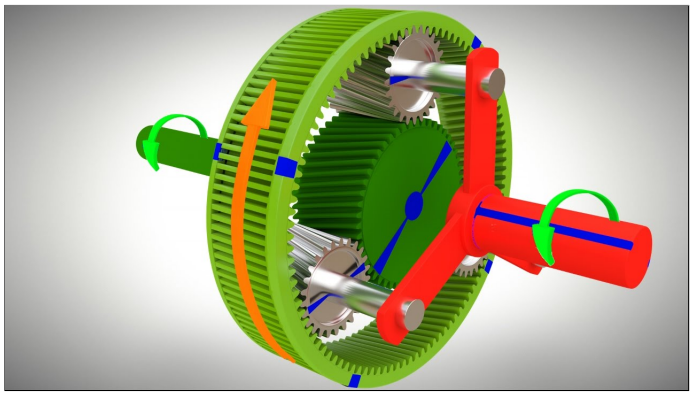

Figura 1: Engranaje planetario con *una corona, un sol, un portador y tres engranajes planetarios*

Una aproximación para el diagnóstico de fallas es a través de *Redes Neuronales Artificiales*, herramienta que hace uso de datos de vibraciones mecánicas en equipos con distintos estados de fallas conocidos para entrenar un modelo computacional que posteriormente permite *clasificar el estado de un equipo* a partir de sus datos de vibraciones. Finalmente, un *Modelo Fenomenológico*, el cual se basa principalmente en la cinemática del equipo, permite la simulación de las vibraciones mecánicas bajo distintas condiciones de operación y niveles de falla.

0.1. Motivación

Conseguir datos para el entrenamiento de una *Red Neuronal Artificial* resulta complejo, pues se necesita datos de las vibraciones mecánicas etiquetados para distintas condiciones de operación, para distintos modos de falla, y debido a que requeriría de un laboratorio para poder generar y adquirir esta información , es prácticamente imposible contar con un conjunto de datos para trabajar en donde se conozca el estado del equipo asociado a cada medición. Aún teniendo los datos disponibles, el modelo creado va a ser específico para el equipo de esos datos, limitando enormemente su rango de aplicación. Tendiendo esto en cuenta es posible crear simulaciones de datos de vibraciones mecánicas con distintos estados de falla y condiciones de operación con un *Modelos Fenomenológico*, para utilizarlos como datos de entrenamiento de una *Red Neuronal Artificial*, y de esta manera evitar tener que conseguir o generar una base de datos.

0.2. Objetivos

Objetivo general: Generar una *Red Neuronal Artificial* entrenada con datos simulados de un *Modelo Fenomenológico*, el cual permita la identificación falla por grieta en base de dientes.

Objetivos específicos:

- Desarrollar un modelo de vibraciones mecánicas para engranajes planetarios en *el software MATLAB* incluyendo los un modo de falla con distintos grados de severidad.
- Obtener datos de vibraciones mecánicas en equipos de reductores planetarios y generar las etiquetas del tipo de falla utilizando herramientas de mantenimiento predictivo.
- Entrenar una *Red Neuronal Artificial* entrenada con datos simulados, para el diagnóstico de los dos modos de fallas estudiados.
- Corroborar los resultados de confiabilidad de la *Red Neuronal Artificial* utilizando los datos de vibraciones mecánicas reales.

0.3. Alcances

El alcance de este trabajo de memoria es la creación de dos programas, uno en el *software MATLAB* para la simulación de vibraciones mecánicas de un reductor planetario que incluya parámetros de operación como velocidad de giro, número de *planetas*, diámetro, paso y parámetros de falla como tamaño de grieta de uno o más dientes, y otro en el *software Python* para el diagnóstico de reductores planetarios a través del uso de *Redes Neuronales Artificiales* entrenadas con datos generados por la simulación.

Capítulo 1

Antecedentes

1.1. Engranajes planetarios

La efectividad de los engranajes planetarios viene desde su diseño, el cual permite obtener altas relaciones de transmisión con dimensiones y pesos relativamente bajos comparados con otros tipos de reductores. En la **Ecuación 1.1** se puede observar la relación que indica la relación de transmisión en un engranaje planetario, y que depende de los radios del *sol* y de la *corona* en función del número de dientes de cada componente y sus radios:

$$1 + \frac{z_R}{z_S} = 1 + \frac{r_R}{r_S} \tag{1.1}$$

En donde:

- z_R = Número de dientes de la corona.
- z_S = Número de dientes del sol.
- r_R = Radio del la corona.
- r_S = Radio del sol.

En las cajas reductoras el uso de un mayor número de planetas genera una mejor distribución de los esfuerzos a través de los engranes, aumentando su vida útil y permitiendo trabajar con fuerzas de mayor magnitud que en equipos convencionales. Cuando se trabaja con reductores de estas características normalmente el anillo exterior se encuentra fijo y estático, por lo que será la que se analizará en este estudio.

1.2. Modelo fenomenológico

El modelo fenomenológico permite la simulación de las vibraciones mecánicas a partir del estudio de la física e interacción de los componentes del equipo a partir de conocimientos teóricos y experimentales, modelando matemáticamente el fenómeno. En este caso se realiza aplicando distintas operaciones matemáticas a señales sinusoidales de distintas frecuencias, amplitudes y fases para generar la señal requerida, además de la aplicación de otras operaciones basadas en distribuciones de probabilidades que agrega variabilidad a la señal, algo que es característico de estas *y que se conocen a partir de fenómenos como el error de transmisión, variación en la frecuencia de giro o ruido en general.*

Para el caso de engranajes planetarios las mediciones siempre se realizan en la carcasa del equipo, ya que resulta complejo medir los componentes móviles, por lo que el modelo fenomenológico estará basado en modelar las vibraciones desde ese punto de referencia, tal como se muestra en la **Figura 1.1**.

Desde este punto la posición de la corona y del sol son constantes, sin embargo, la posición de los planetas varía constantemente, lo que genera un *fenómeno de modulación en amplitud el cual debe será incluido en el modelo.*

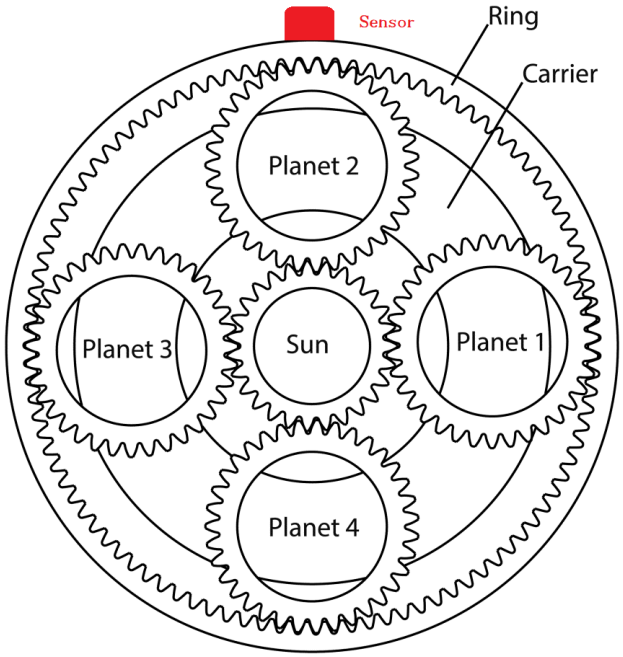

Figura 1.1: Posición del sensor en la caracasa del anillo exterior.

Un elemento de suma importancia en el análisis de vibraciones con engranajes son las *frecuencias de engrane*, que corresponden a las frecuencias características asociadas al funcionamiento de este tipo de equipos y que se pueden encontrar normalmente en el *Espectro de Fourier de las vibraciones.* A continuación se describen estas frecuencias en función de las características de sus componentes:

- Frecuencia de engrane: Cuando un planeta ha completado una revolución ha engranado la misma cantidad de dientes con el sol y con el anillo exterior, por lo que la frecuencia de engrane entre estos tres componentes es la misma. En la **Ecuación 1.2** se muestra la fórmula para el cálculo de la frecuencia de engrane.

$$f_m = f_H \cdot z_r = \frac{z_r \cdot z_s}{z_r + z_s} \cdot f_s \tag{1.2}$$

- Frecuencia con falla localizada: Cuando la falla se encuentra localizada en un diente se espera una señal modulada por un impulso por cada impacto. Si K es el número de planetas en el reductor, por cada giro del sol se producirán K impactos. En las Ecuaciones 1.3, 1.4 y 1.5se muestran las fórmulas para el cálculo de las frecuencias con falla distribuida para el sol, planetas y corona.

$$f_{sl} = K \cdot \frac{z_r}{z_r + z_s} \cdot f_s \tag{1.3}$$

$$f_{pl} = \frac{z_r \cdot z_s}{z_p \cdot (z_r + z_s)} \cdot f_s \tag{1.4}$$

$$f_{rl} = K \cdot \frac{z_s}{z_r + z_s} \cdot f_s \tag{1.5}$$

En la **Figura 1.2** se puede observar de perfil la forma de una grieta en uno de los dientes un engranaje.

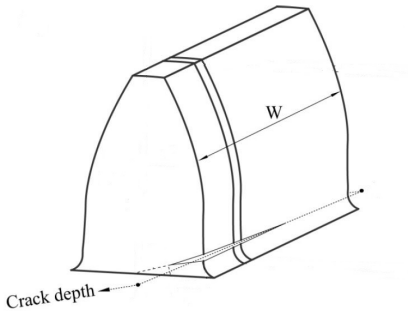

Figura 1.2: Ejemplo de grieta en la base de un diente .

1.2.1. Elementos del modelo

Existe una gran variedad de fenómenos que se pueden tener en consideración, y en el modelo de este estudio se tendrá en consideración las siguientes:

- Excitaciones aleatorias: Estas pueden ser generadas por impurezas en la lubricación o impulsos aleatorios debido a variaciones en la superficie de los engranajes.
- Grieta en base del diente: Corresponde a la formación de grietas en la base de los dientes debido al desarrollo de una falla local. Este tipo de problemas genera variaciones en la rigidez del diente que se hace notar cuando este engrana con otro diente.

La manera en que estas fallas se pueden incluir dentro del modelo es incluyendo componentes a las frecuencias características de las fallas a distintas amplitudes, agregando ruido a la señal y utilizando una distribución de probabilidades para modelar las frecuencias de giro, ya que el funcionamiento de un equipo mecánico nunca es perfecto y las frecuencias de giro tienden a ser variables en torno a un determinado rango, más parecido a una distribución.

1.3. Machine Learning

Las *Redes Neuronales Artificiales* son un modelo computacional basado en el funcionamiento de los componentes de las neuronas de cerebros biológicos. Consisten en capas de neuronas con conexiones entre ellas, partiendo por una *capa inicial* en donde se introducen todos los datos iniciales, una o más *capas escondidas*, en donde se realizan las conexiones para identificar qué datos resultan más relevantes, una *capa de salida* .

Los componentes principales de estas redes son:

- Conexiones: Cada nodo tiene conexiones con otros nodos de la anterior y la siguiente capa, y cada conexión tiene un peso asociado. Estos pesos relacionan la importancia de la información que carga un nodo en relación a los otros.
- Nodos: En las *capas escondidas y la capa de salida* los nodos poseen valores asociados a una función de activación *no - lineal*, cuyo argumento depende de las conexiones que tiene con nodos anteriores y los pesos asignados a estas.

La arquitectura de una *ANN (Artificial Neural Networks)* está relacionado con los parámetros que se seleccionan para la configuración de la red, entre los que se encuentran *el número de capas, el número de neuronas entre cada capa, las funciones de activación de los nodos de cada capa, entre otros.* En la **Figura 1.3** se puede observar la arquitectura de una res bastante simple, en donde se distingue una capa de entrada, 1 capa escondido y 1 capa de salida, los valores que toma cada nodo y los pesos asociados a las neuronas de cada capa.

Una de las principales aplicaciones de las *ANN* es el la clasificación de estados. Para esto la red utiliza un algoritmo iterativo que le permite *ser entrenada con la información de los datos que se le entrega*, y a través de un algoritmo llamado *backpropagation* puede actualizar los pesos de sus conexiones en función del error en la predicción realizada inicialmente y el valor real al que debía llegar, conocido como etiqueta. A continuación se describe de manera general el funcionamiento de una *ANN* basada en el documento [3]

1. Se asignan pesos aleatorios a todas las conexiones para comenzar el algoritmo.
2. Se utilizan lo datos de entrada y los pesos de las conexiones para calcular los valores de los nodos utilizando las funciones de activación asignadas hasta la capa de salida.
3. Se calcula el error entre los resultados a través del algoritmo y los resultados reales.
4. Desde la capa de salida se modifican los pesos de las conexiones hasta llegar a la capa de entrada.
5. Se repite el procesos hasta que el criterio de convergencia se cumple.

Una vez que la red haya sido entrenada, es posible clasificar datos sin necesidad de conocer sus etiquetas, teniendo una confianza en que el resultado va a ser aproximadamente proporcional al criterio de convergencia al que se llegó en el entrenamiento si es que los datos introducidos son semejantes a los del entrenamiento. Una aplicación directa de esto es la realización de un algoritmo de clasificación que permita identificar el estado de una máquina, en este caso en particular, se podría estudiar si un reductor planetario posee el tipo de falla descrito anteriormente, *grieta en base del diente* o si simplemente el equipo se encuentra en buen estado, sin falla.

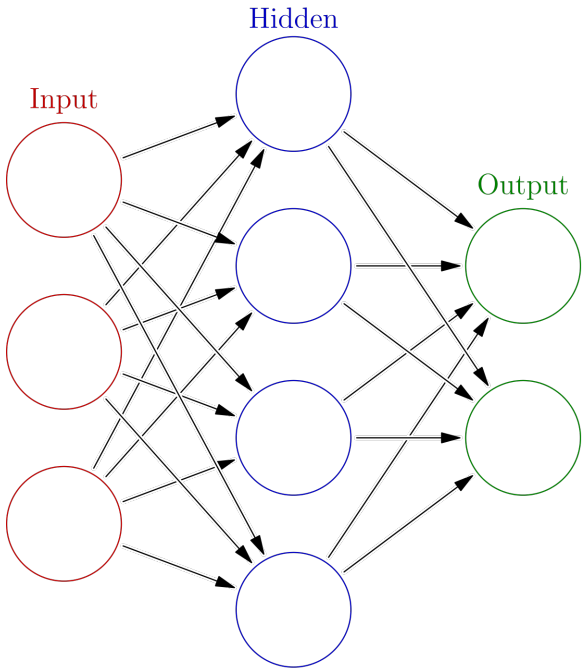

Figura 1.3: Ejemplo de una Red Neuronal Simple.

1.4. Mantenimiento predictivo

El *mantenimiento predictivo* es una estrategia de mantenimiento basada en el monitoreo de equipos a través de las mediciones de sus parámetros operacionales con la finalidad de identificar a tiempo cambios en sus condiciones de operación que puedan ser indicadores de fallas y tomar acciones para corregirlas antes de que estas aumenten en magnitud. Se diferencia de otras estrategias de mantenimiento pues *permite aprovechar al máximo el tiempo de uso de un equipo*, evitando tener que someterlo a labores de mantenimiento antes de que sea necesario, o cuando la falla ya se ha desarrollado lo suficiente como para que tenga consecuencias más graves, desde el desgaste de otros componentes además del que se posee la falla hasta la detención del proceso productivo debido a una falla catastrófica en el equipo.

Equipos de medición

El mantenimiento predictivo *se basa fuertemente en el uso de sensores que puedan medir variables de interés, principalmente vibraciones mecánicas, y en otros. Los principales equipos utilizados se detallan a continuación:*

- Acelerómetro: Aceleración, rango de frecuencias relativamente altas. $\left[\frac{m}{s^2}\right]$
- Velocímetro: Velocidad, rango de frecuencias medio.$\left[\frac{m}{s}\right]$
- Proxímetro: Posición, rango de frecuencias bajas $[m]$
- Termómetro: Temperatura, la cual aumenta en equipos con fallas debido al roce que se produce. $[°C]$
- Tacómetro: Revoluciones por minuto, es un equipo que permite medir la cantidad de revoluciones del giro de un eje en tiempo real. $[RPM]$

Figura 1.4: Sensor EAGLE de la marca OneProd.

En la **Figura 1.4** se puede observar un sensor de vibraciones mecánicas de la marca *OneProd* llamado EAGLE, el cual tiene las siguientes características:

- Acelerómetro triaxial: Puede medir en tres dimensiones espaciales de manera sincrónica.
- Sensor de temperatura: El equipo puede medir la temperatura de la superficie en la que se encuentra.
- Mediciones inalámbricas: Permite monitorear de manera remota y periódica, por lo que no es necesario que nadie se acerque al equipo para realizar las mediciones, disminuyendo completamente el riesgo a un daño material.

1.4.1. Métodos de análisis de fallas

Una vez propuesto el modelo fenomenológico se debe encontrar una manera efectiva de realizar el análisis de las fallas, para lo cual se utilizará principalmente la *Transformada de Fourier*. La transformada de Fourier es una herramienta que permite pasar desde el dominio del tiempo al dominio de la frecuencia, y tiene excelentes aplicaciones en el análisis de fallas en equipos rotatorios debido a que estos son principalmente fenómenos periódicos.

El tipo de fallas que se estudiarán en este trabajo consisten principalmente en fallas locales debido a grietas que se generan y desarrollan, las cuales se generan de manera natural en zonas de grandes esfuerzos, como son las áreas de contacto entre dientes dentro de un engranaje. En los engranajes planetarios en particular existen varias interacciones, ya que se tiene que tener en cuenta que tienen más componentes de un par de engranaje corona - piñon. Este tipo de fallas se caracterizan por tener una forma de onda con impactos, caracterizados por una amplitud relativamente alta comparada con la señal misma, y de un periodo de duración relativamente corto, así como por el hecho de tener una periodicidad asociada a las características mismas de la falla. Debido a esto se utilizarán técnicas de filtrado de señales como el *Filtro Autoregresivos (AR), la Deconvolución de Mínima Entropía (MED) y una combinación de estos llamado ARMED.*|

La importancia de este proceso reside en facilitar el trabajo a la *Red Neuronal* para poder trabajar con los dato para identificar las fallas. Esto se puede conseguir de varias maneras, *por ejemplo filtrando el ruido de la señal*, para lo cual es necesario saber en qué bandas de frecuencias se encuentran las componentes que pueden estar asociadas a fallas, para lo cual el *modelo fenomenológico* servirá de base.

1.4.2. Filtro AR

Un *modelo Autoregresivo (AR)* es la representación de un proceso variable en el tiempo, y se enmarca en el contexto de estadística y procesamiento de señales [10]. La característica de este modelo es que relaciona la variable de salida de manera lineal con los valores anteriores a este en la sucesión temporal. La cantidad de valores anteriores que se toman en consideración para realizar le predicción lineal es llamado *orden del modelo*. Una de los requisitos de estos modelos es para que se ajusten apropiadamente la señal que se estudia es asumir que ésta es estacionaria. En la **Ecuación 1.6** se puede observar la notación característica de un proceso *Autoregresivo*.

$$X_n = C + \sum_{\mathrm{i}=1}^{p} \phi_{\mathrm{i}} \cdot X_{t-\mathrm{i}} + \varepsilon_t \tag{1.6}$$

En donde:

- X_n: Predicción del valor siguiente en el proceso.
- C: Constante del proceso.
- $\phi_1 ... \phi_p$: Parámetros del modelo AR.
- p: Órden del proceso.
- ε_t: Residuo.

La idea de aplicar un modelo *Autoregresivo* para generar un filtro es separar la señal real en dos partes, una estacionaria y otra no - estacionaria, asociada al residuo. La parte estacionaria corresponde a los fenómenos que se *repiten constantemente en la señal y están asociados a su funcionamiento regular y sin falla*, como la modulación de los planetas a la frecuencia de giro del portador, o las señales asociadas a la frecuencia de giro del motor y del portador, mientras que la parte no - estacionaria *se relaciona con los fenómenos de impactos debido a grietas en los dientes y falla en general, sobre todo de fenómenos impulsivos*. De esta manera, con el *modelo Autoregresivo* se busca extraer la parte estacionaria de la señal para posteriormente sustraerla a la señal original, dejando únicamente las componentes no - estacionarias y relacionadas con las fallas en los equipos.

Para realizar este proceso se utiliza la función *aryule* en *MATLAB*, la cual calcula y entrega los parámetros de un modelo *Autoregresivo* de un orden determinado. Para determinar el orden del modelo se utiliza la *kurtosis* para calcular el nivel de impulsividad de la señal resultante una vez que se sustrae la componente estacionaria de los datos experimentales. La *kurtosis*, cuya ecuación se puede observar en la **Ecuación 1.7**, es un parámetro utilizado en estadístico que, dicho de manera simple, indica lo propensa que es la variable a tener valores alejados del promedio en su distribución, lo cual es sumamente útil pues los impactos debido a grietas son fenómenos que escapan de la distribución normal de datos en las vibraciones mecánicas de un reductor. De esta manera *el código prueba con los órdenes entre 1 y 1000 para el filtro AR, y calcula la kurtosis del resultante, quedándose con el orden que entregue el valor de kurtosis más alto.*

$$K[X] = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] \tag{1.7}$$

En donde:

- $K[X]$: Kurtosis del conjunto de datos X.
- $E[X]$: Esperanza del conjunto de datos X.
- μ: Media del conjunto de datos X.
- σ: Desviación estándar del conjunto de datos X.

1.4.3. Filtro MED

El filtro de *Deconvolución de Mínima Entropía* o MED por sus siglas en inglés *(Minimum Entropy Deconvolution)* es un tipo de procesamiento creado en 1978 por Ralph A. Wiggins [7] para facilitar la extracción de información anómala en mediciones de procesos geológicos, usualmente con forma de *peaks* separados en el tiempo, los cuales se encuentran convolucionados con otros tipos de señales. Para poder aplicar el procesamiento la señal debe cumplir con las siguientes características:

- La señal debe contener *peaks* separados en el tiempo.
- La señal debe estar convolucionada con otra señal de tipo estacionaria.

El objetivo del procedimiento es encontrar los parámetros de un filtro de frecuencias el cual pueda hacer que la señal se simplifique, en el aspecto que quede únicamente con *peaks* en el tiempo, eliminando todas las componentes estacionarias y cualquier fuente de ruido. Según las palabras del autor del artículo [7] *' El método maximiza el orden, o de manera equivalente, minimiza la entropía de la señal, de acá el nombre Deconvolución de Mínima Entropía'.*

El método descrito en la referencia busca maximizar la *Varianza normalizada* de los datos una vez aplicado el filtro, la cual se describe en las **Ecuaciones 1.8, 1.9, 1.10 y 1.11**.

$$V = \sum_{\mathrm{i}} V_{\mathrm{i}} \tag{1.8}$$

$$V_{\mathrm{i}} = \frac{\sum_j y_{\mathrm{i}j}^4}{\left(\sum_j y_{\mathrm{i}j}^2\right)^2} \tag{1.9}$$

$$y_{\mathrm{i}j} = \sum_{k=1}^{N_f} f_k \cdot x_{\mathrm{i},j-k} \tag{1.10}$$

$$x_{\mathrm{i}j} \begin{cases} \mathrm{i} = 1 \ldots N_s \\ j = 1 \ldots N_t \end{cases} \tag{1.11}$$

En donde:

- N_s: Número de señales de muestra. [S/U]
- N_s: Largo de las señales. [S/U]
- $y_{\mathrm{i}j}$: Salida de la señal después de aplicar el filtro MED.
- V : Sumatoria normalizada de los cuadrados de las varianzas de los datos.

Para encontrar los parámetros del filtro se desea maximizar la *Varianza normalizada*, para lo cual propone la **Ecuación 1.12**, cuya solución se encuentra indicada en la **Ecuación 1.13**.

$$\frac{\partial V}{\partial f_k} = 0 = \sum_{\mathrm{i}} \frac{\partial V_{\mathrm{i}}}{\partial f_k} \tag{1.12}$$

$$\mathbf{R} \cdot \mathbf{f} = \mathbf{g} \tag{1.13}$$

En donde:

- R: Es una matriz de autocorrelación calculada como una suma ponderada de las autocorrelaciones de las señales de entrada.
- g: Es un vector de relación - cruzada *(cross - correlation)* calculado como una suma ponderada entre la señal original y la señal filtrada al cubo.

El proceso *altamente no - lineal*, por lo tanto se debe resolver de manera iterativa, en donde en primera instancia se asume un valor de f, calculando R y g, para posteriormente calcular nuevamente el valor de f, además, la secuencia iterativa puede no llevar al valor máximo de la *Varianza Normalizada*, sin embargo, se puede encontrar un máximo útil.

1.4.4. Filtro ARMED

El *filtro ARMED* corresponde al uso de ambas técnicas presentadas en las secciones anteriores en conjunto, aplicando en primera instancia un *filtro AR* para eliminar la componente estacionaria de la señal para luego aplicar el *filtro MED* para encontrar un filtro que maximice la varianza de la señal.

Capítulo 2

Metodología

2.1. Programación del modelo Fenomenológico

El primero paso en este trabajo de memoria consiste en la programación de un modelo fenomenológico, el cual permita realizar la simulación de vibraciones mecánicas de un reductor planetario en condiciones con y sin fallas. El modelo será programado en el *software MATLAB*, y tendrá en consideración modelos como el de las referencias [1] [2] [4] [6] [11] [9] . Se realizarán distintas combinaciones de los modos de fallas estudiados, principalmente grieta en base del diente en los distintos componentes del reductor, y con distintos grados de severidad, para simular una amplia variedad de casos. La metodología para este paso incluye:

- Generar el modelo fenomenológico de un reductor planetario de una sola etapa con un número determinado de planetas.
- Incluir *excitaciones aleatorias* en el modelo a través de variabilidad en las frecuencias del modelo.
- Incluir errores de grieta en base del diente a través de la inclusión de componentes a las frecuencias de falla a distintas amplitudes.
- Configurar una variedad de casos y combinaciones de casos para el estudio posterior.

2.2. Mediciones experimentales en reductores

Tanto para corroborar el resultado de la *ANN* como para corroborar los resultados del *modelo fenomenológico* se tienen que realizar mediciones en reductores planetarios en equipos reales. Para esto se realizará un montaje experimental con un motor de partida, equipo que consiste en un motor que se conecta a una fuente de corriente continua y posee un reductor planetario a su salida, aumentando el torque a la salida. Los componentes del motor y el montaje se puede observar en la **Figura 2.1**

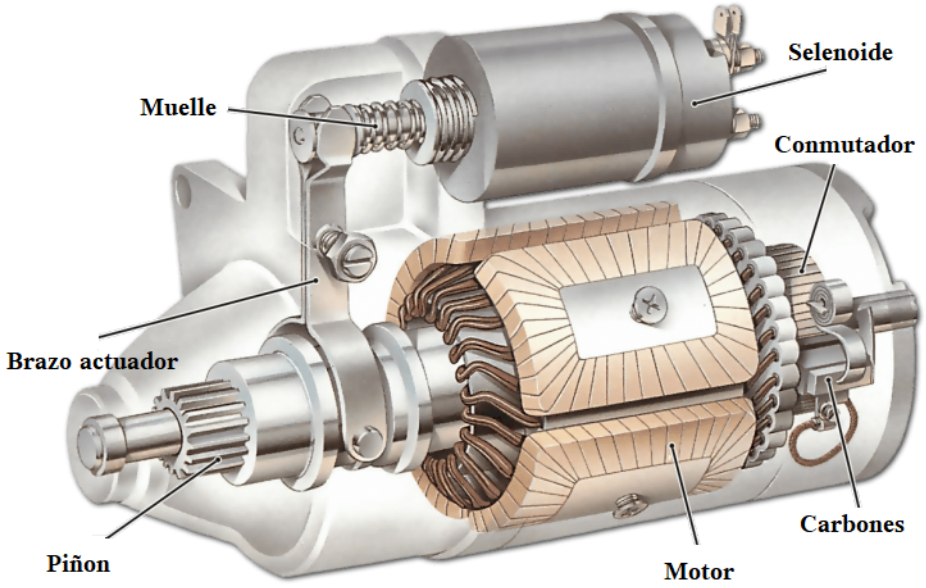

Figura 2.1: Componentes de un motor de partida .

Debido a que el equipo que se utiliza está nuevo se sabe que se encuentra en un estado nominal. Para la inclusión de fallas se utilizará una herramienta *dremel* con un disco de corte para debilitar la zona en torno a la base de los dientes en los distintos componentes del reductor.

Se construirá un montaje experimental que considere la aplicación de carga sobre el eje mediante un *Freno Prony* utilizando una correa unida a un dinamómetro, con el cual se podrá dimensionar la carga aplicada sobre el eje y mantenerla constante durante todas las mediciones.

Utilizando las mediciones realizadas se puede comparar la semejanza entre los datos experimentales y datos simulados bajo las mismas condiciones de trabajo, una manera de hacer de esto sería comparar el contenido frecuencial de ambas mediciones, el cual debería ser semejante.

2.3. Programación de red neuronal

Para la creación de la la red neuronal se considerará la programación de la *Red Neuronal Artificial (ANN)* para realizar la clasificación de los distintos modos de falla. Las tareas a seguir para este paso se describen a continuación:

- Crear y etiquetar simulaciones vibraciones mecánicas con distintas condiciones de falla para entrenar la red neuronal.
- Etiquetar datos experimentales de las vibraciones en reductores planetarios.
- Entrenar distintas Redes Neuronales para cada una de las configuraciones de las que se disponen datos. El entrenamiento se realiza con los datos con etiquetas generados a partir de las simulaciones hasta obtener un alto porcentaje de confiabilidad en las redes.
- Validar las redes con los datos experimentales. Si se obtienen buenos resultados de confiabilidad entonces se corrobora la utilidad del modelo.

Capítulo 3

Programación del Modelo Fenomenológico

En esta sección se discutirán todos los aspectos relacionados con el modelado y programación del modelo fenomenológico del engranaje planetario. Se realizará una descripción del modelo, identificando sus principales características y con alcances respecto a los supuestos tomados, se describen las ecuaciones utilizadas y se explica de qué manera se incluyen los distintos tipos de fallas en el modelo. Finalmente se describen las funciones utilizadas en el *software MATLAB* para generar las señales.

3.1. Descripción del modelo

El modelo busca representar las vibraciones de un engranaje planetario de una sola etapa utilizando el conocimiento experimental que se tiene acerca del comportamiento de este tipo de equipos en distintos casos de fallas, para lo cual se utilizan referencias como [4].

Para esto se generarán las señales temporales para el caso sin falla, o funcionamiento normal, y distintos modos de fallas, correspondiente a grieta en la base de los distintos engranajes que componen al reductor planetario, buscando describir las características de las señales a través de ecuaciones cinemáticas del equipo.

El modelo debe tener en consideración que en general en el trabajo con equipos mecánicos existen variaciones en cuanto a las frecuencias de giro, aleatoriedad en las amplitudes y ruido en general, por lo que se utilizarán distribuciones de probabilidad para modelar estos componentes en la señal temporal, buscando agregar realismo a la simulación.

3.1.1. Modelo para la señal del equipo en estado nominal

Cuando el equipo funciona en estado nominal la señal de la vibración que percibe el sensor ubicado en la carcasa del equipo debe tener componentes a la velocidad de entrada en el eje conectado al sol, la velocidad de salida del portador, y el engrane entre los dientes de los planetas - corona y planetas - sol.

Para modelar la señal asociada a las velocidades de giro de entrada y de salida se tiene en consideración las características del equipo utilizado, las cuales se pueden observar en la **Tabla 3.1**. Para la simulación de las señales se utilizarán los parámetros obtenidos durante una de las mediciones del montaje experimental las cuales se resumen en la **Tabla 3.2**, sin embargo, se debe tener en cuenta que el modelo permite simular equipos con distintas características físicas y a distintas frecuencias de giro. Estas frecuencias son teóricas, estando basadas en las **Ecuaciones 1.2, 1.4 y 1.5.**

Tabla 3.1: Resultados de ensayos de dureza

Características	Magnitud	Unidad
Número de dientes del anillo	50	[S/U]
Número de dientes del sol	10	[S/U]
Número de dientes de los planetas	20	[S/U]
Velocidad de giro de entrada	16.110	[RPM]
Velocidad de giro de salida	2.685	[RPM]

Tabla 3.2: Frecuencias características del reductor.

Características	Magnitud	Unidad
Frecuencia de giro del sol	268	[Hz]
Frecuencia de giro del carrier	44,75	[Hz]
Frecuencia de engrane	2.237	[Hz]
Frecuencia de falla local en el sol	671.2	[Hz]
Frecuencia de falla local del anillo	134.2	[Hz]
Frecuencia de falla local del planeta	111.8	[Hz]

3.1.2. Modelado de la señal del planeta

Al ser un modelo fenomenológico se estudia el problema desde la posición del sensor, en el casing del equipo como se muestra en la **Figura 1.1**, desde donde se transmiten las vibraciones desde todo el equipo, pero principalmente de la interacción entre la corona y los planetas, cuya frecuencia característica es la frecuencia de engrane, la cual se calcula según la **Ecuación 1.2**. Se debe tener en consideración que cada vez que un diente del planeta engrana con un diente del anillo exterior y del sol, lo cual ocurre simultáneamente, se genera un impacto, por lo que para modelar este comportamiento se utiliza una señal sinusoidal a la frecuencia de engrane, además, para tener en cuenta que el engrane no siempre es perfecto debido a problemas como el *error de transmisión*, por lo que se agrega ruido a la señal utilizando una *Distribución Normal* con $\mu = 0$ y $\sigma = 1$, de amplitud modificable, tal como se muestra en la **Figura 3.1**, en donde la amplitud del error es equivalente al 5 % de la amplitud de la señal sinusoidal que simula el engrane.

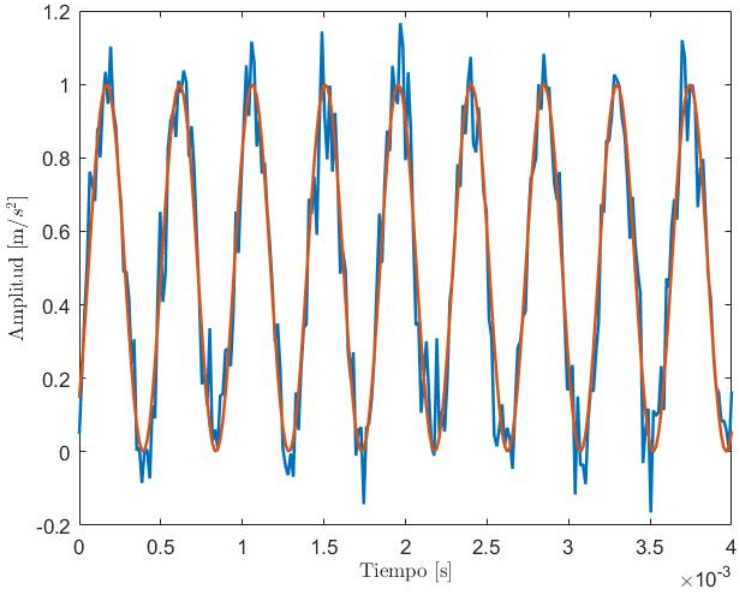

Figura 3.1: Señal simulada del engrane de 1 planeta con ruido gaussiano.

Ahora, en el equipo existen 3 planetas, los cuales se encuentran distribuidos uniformemente sobre el aro exterior, y si se tiene en consideración que este se encuentra estático respecto al movimiento de los otros componentes se puede calcular la fase de cada planeta tomándolo como punto de referencia, según como se indica en la **Tabla 3.3**. En la **Figura 3.2** se pueden observar las señales de los tres planetas por separado y la fase correspondiente.

Tabla 3.3: Fase de los planetas en el equipo.

Parámetro	Magnitud	Unidad
Fase del planeta 1	0	[rad]
Fase del planeta 2	$\frac{2\cdot\pi}{3}$	[rad]
Fase del planeta 3	$\frac{4\cdot\pi}{3}$	[rad]

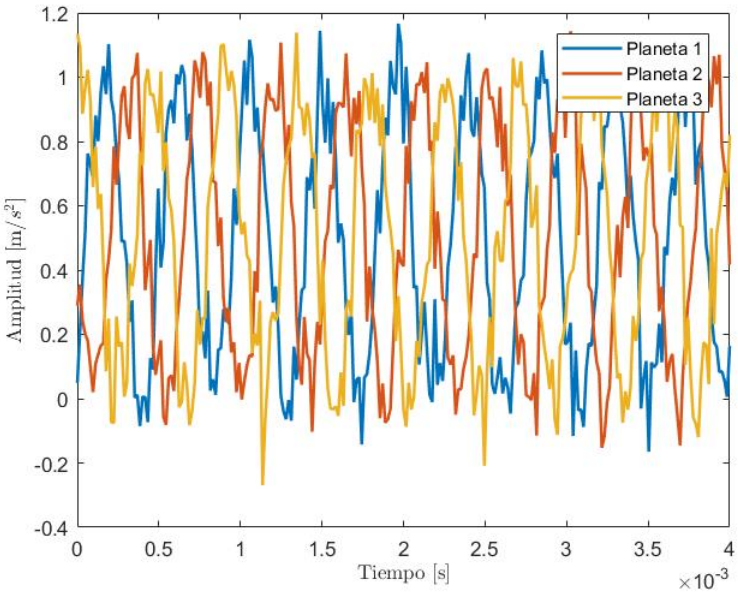

Figura 3.2: Señal simulada del engrane de los 3 planetas con ruido.

Cuando uno de los planetas pasa más cerca del sensor, la amplitud de la vibración de este es máxima desde el punto de referencia, y de la misma manera, cuando el planeta se encuentra más lejos del sensor la amplitud de la vibración es mínima, para lo cual se agrega al modelo un fenómeno de modulación en amplitud, en el que la vibración de los planetas se encuentra modulada a la *señal del giro del portador*, variando la amplitud de la señal que mide el sensor dependiendo de la distancia a la que se encuentra de este. Las señales que modulan a cada planeta dependen de la posición de cada uno respecto al aro exterior, y se encuentran en fase según lo indicado en la **Tabla 3.4**. En la **Figura 3.3** se puede observar la señal que modula al planeta 1, y en la **Figura 3.4** se puede observar la aceleración del planeta 1 siendo modulada en amplitud por la frecuencia del carrier. En la **Figura 3.5** se puede observar en la *Transformada de Fourier* de la señal modulada del planeta 1 , en donde se observa claramente una componente a la frecuencia de engrane, y dos bandas laterales ubicadas a una distancia igual a la frecuencia del carrier desde el centro, lo que es característico de las señales con modulación en amplitud, además, se puede observar la frecuencia del carrier, la cual aparece debido a las características de esta señal y el ruido distribuido uniformemente a través del espectro.

Tabla 3.4: Fase de las señales que modulan los planetas del equipo.

Parámetro	Magnitud	Unidad
Fase modulación planeta 1	0	[rad]
Fase modulación planeta 2	$\frac{2\cdot\pi}{3}$	[rad]
Fase modulación planeta 3	$\frac{4\cdot\pi}{3}$	[rad]

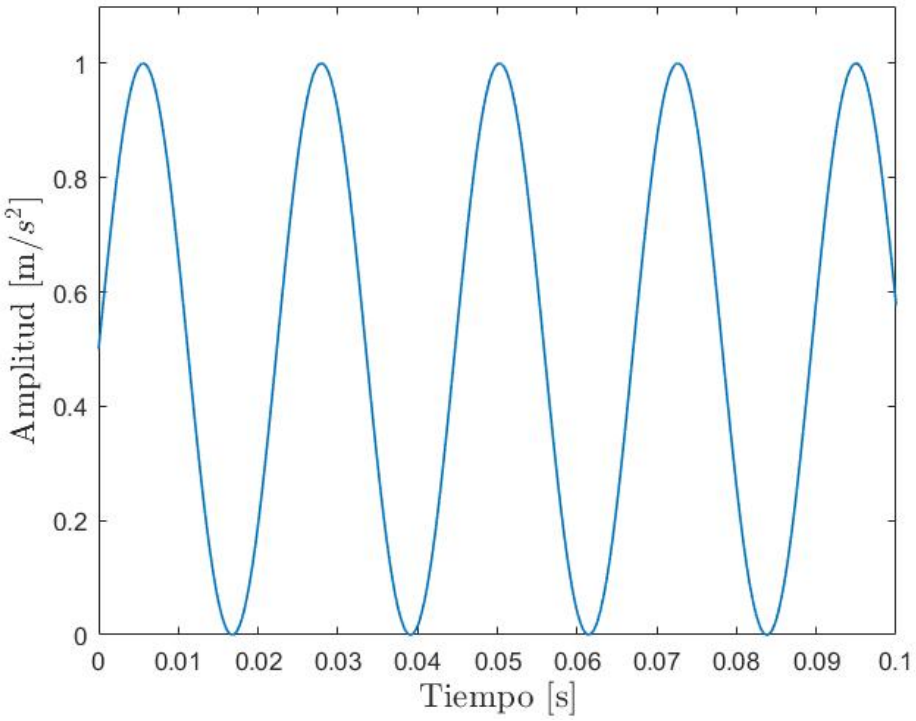

Figura 3.3: Señal que modula la vibración del planeta 1.

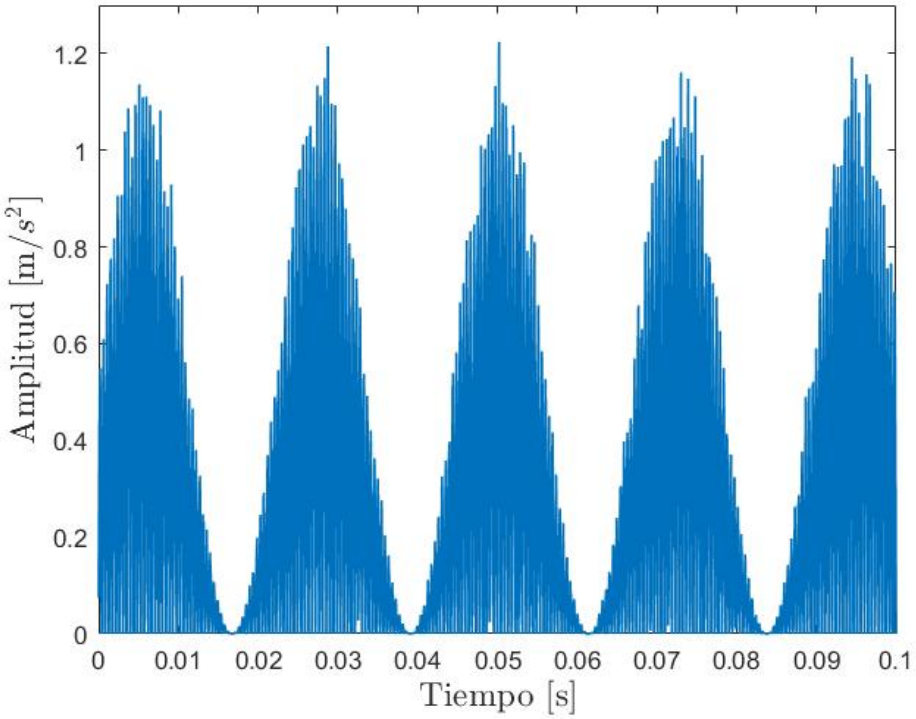

Figura 3.4: Aceleración del planeta 1 modulada a la frecuencia del carrier.

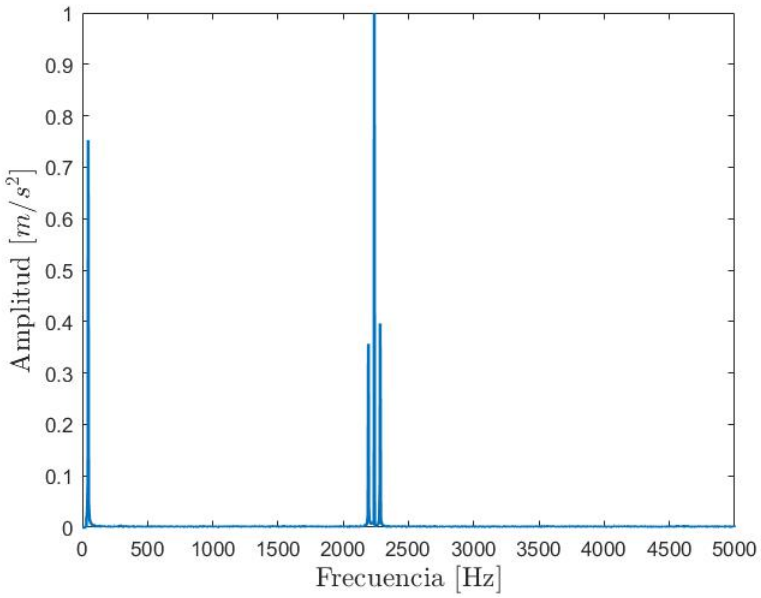

Figura 3.5: Espectro de la señal modulada del planeta con ruido.

Ahora, en la **Figura 3.6** se puede observar una comparación de las tres señales moduladas de cada planeta, mientras que en la **Figura 3.7** se puede observar la suma de las tres señales, la cual parece a simple vista una señal sinusoidal con una componente constante y ruido, sin embargo, en la **Figura 3.8** se puede observar la transformada de Fourier de esta señal, la cual posee una única componente importante ubicada a 2282 [Hz], la cual no corresponde a la frecuencia de engrane de 2237 [Hz] , *y proviene directamente de la suma de las tres señales de los planetas modulados como una banda lateral de la componente a la frecuencia de engrane sumada a la frecuencia de giro del carrier.*

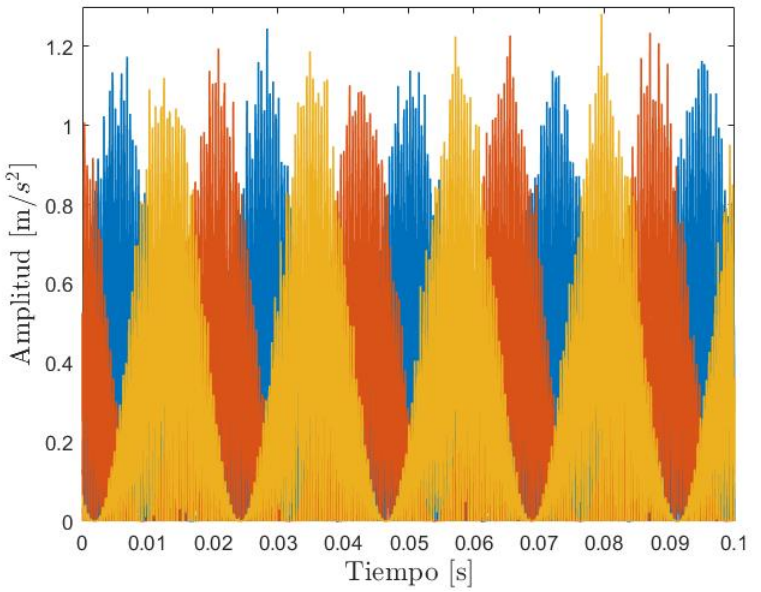

Figura 3.6: Comparación de las 3 señales moduladas.

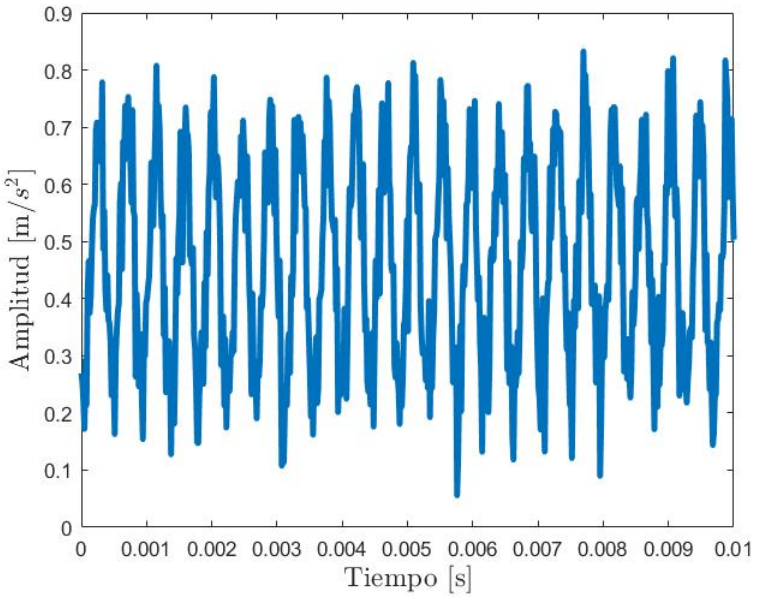

Figura 3.7: Señal que siente el sensor en la carcasa debido a la suma de las señales generadas por los planetas.

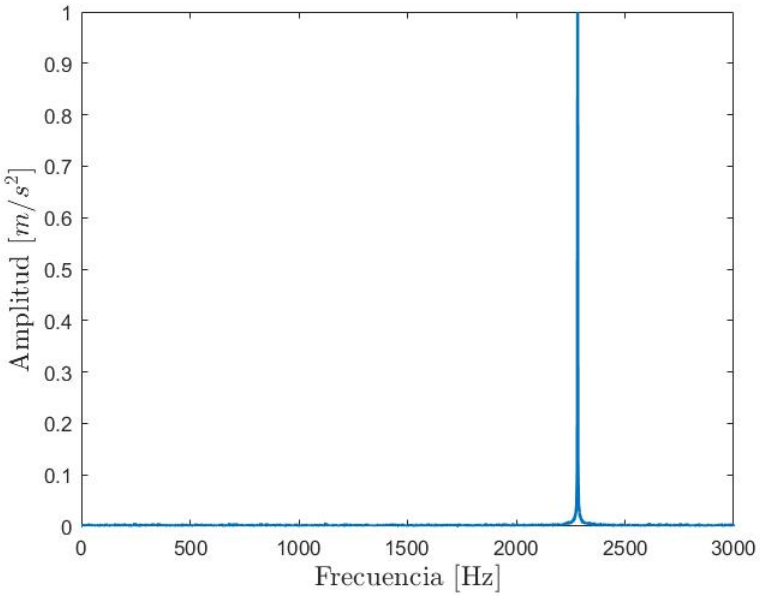

Figura 3.8: Espectro de la suma de las señales moduladas de los planetas.

Componente asociada a las frecuencias de giro de entrada y salida

Para modelar las vibraciones del eje de entrada y de salida se utilizarán funciones sinusoidales a las frecuencias correspondientes a los fenómenos indicados, agregando también una pequeña componente con un error con *Distribución Normal*. Debido a que el portador se encuentra más cerca del sensor que el sol se considera que las vibraciones de este último tienen menor amplitud que el primero, teniendo en consideración también que el camino de las vibraciones a través del planeta pueden atenuar esta componente. En la **Figura 3.9** se puede observar el espectro de las señales de giro de entrada y de salida del reductor, las cuales tienen un error de amplitud igual al 8 % de la amplitud de las señales.

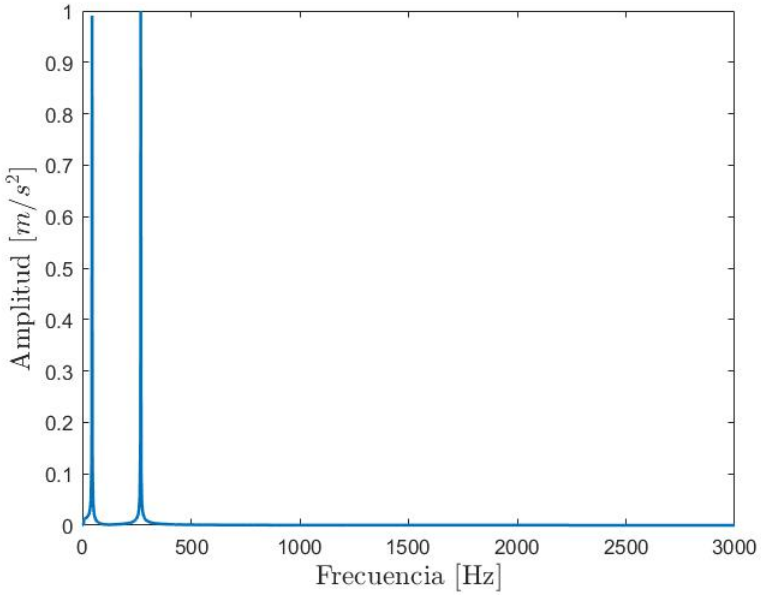

Figura 3.9: Espectro de las señales de entrada y de salida

3.1.3. Inclusión de fallas

Al considerar las fallas en el modelo aparecerán otras componentes asociadas a las frecuencias que se indicaron en las **Ecuaciones 1.3, 1.4 y 1.5**, las que corresponden a las frecuencias de fallas locales. Las fallas locales están usualmente asociadas a grietas de distintos tamaños en la base de los dientes del engranaje, que es normalmente donde se generan debido a que suelen ser las zonas sometidas a mayores esfuerzos. Estas grietas generan variaciones en la rigidez del contacto entre los dientes, lo cual modifica la manera en la que se transmite la potencia entre ellos, y a su vez genera un aumento en la magnitud del desplazamiento, velocidad y aceleración en ese momento. En la **Figura 1.2** se puede observar un gráfico con la variación de la rigidez en función del tiempo, en donde se puede reconocer claramente 3 estados:

1. **Caso 1:** Cuando la rigidez de contacto tiene mayor amplitud es cuando existen dos pares de dientes en contacto. La cantidad de tiempo que esto ocurre depende del *contact ratio*, que es un promedio de la cantidad de dientes en contacto en una transmisión con engranajes.
2. **Caso 2:** Cuando la rigidez disminuye, pero de manera regular, es cuando el hay un solo par de dientes en contacto, y nuevamente la cantidad de tiempo que esto ocurra depende del *contact ratio*.
3. **Caso 3:** Cuando la rigidez disminuye aún más que en el caso 2 entonces se debe a que un diente con problemas de rigidez engranó.

La **Tabla 3.2** indica las frecuencias de las fallas locales para distintos casos.

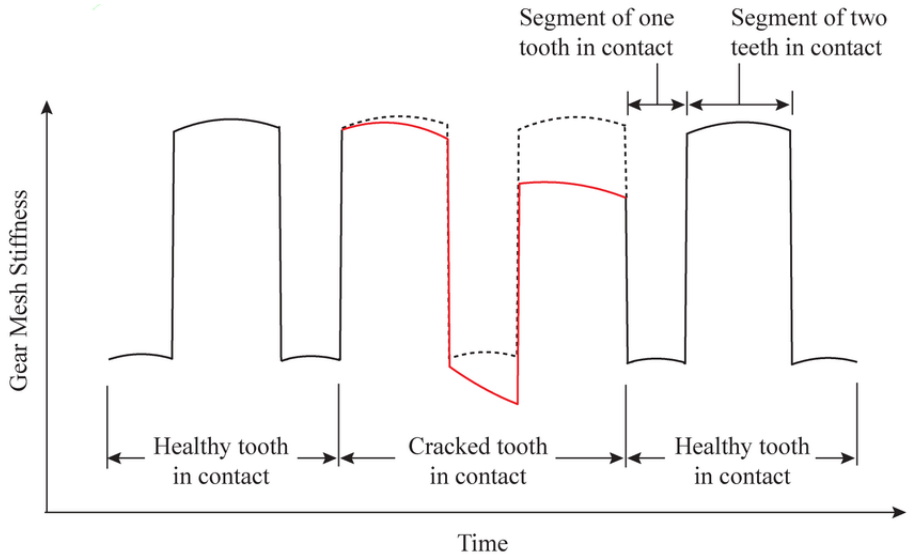

Figura 3.10: Variación en la rigidez de contacto debido a fallas locales en los engranajes.

Simulación de falla local en el planeta

Cuando un planeta tiene una grieta en uno de sus dientes entonces cada vez que realiza contacto con el sol o con el aro exterior se genera un impacto que aumenta la aceleración durante esa interacción. Para simular este tipo de falla se utilizará el mismo modelo descrito en la sección anterior, pero se modificará la señal temporal, aumentando su magnitud durante ese periodo de engranaje, lo cual ocurrirá una vez por cada giro completo que de el planeta. El modelo puede simular más de un engranaje con falla.

Es importante indicar que debido a que los planetas engranan por un lado de un diente con el sol y por el otro lado con el anillo exterior, se considera que una falla local solo interactúa con los otros engranajes una sola vez por ciclo. En la **Figura 3.11** se puede observar un ejemplo de la forma de onda de la aceleración del planeta con un aumento en la magnitud de la vibración durante un ciclo de engrane debido a una falla local. En la **Figura 3.11** también se puede observar la forma de *onda cuadrada utilizada para aumentar la magnitud de la señal en un periodo del engrane*, y debido a que los planetas tienen 20 dientes el aumento en la magnitud del engrane aparece cada 20 ciclos.

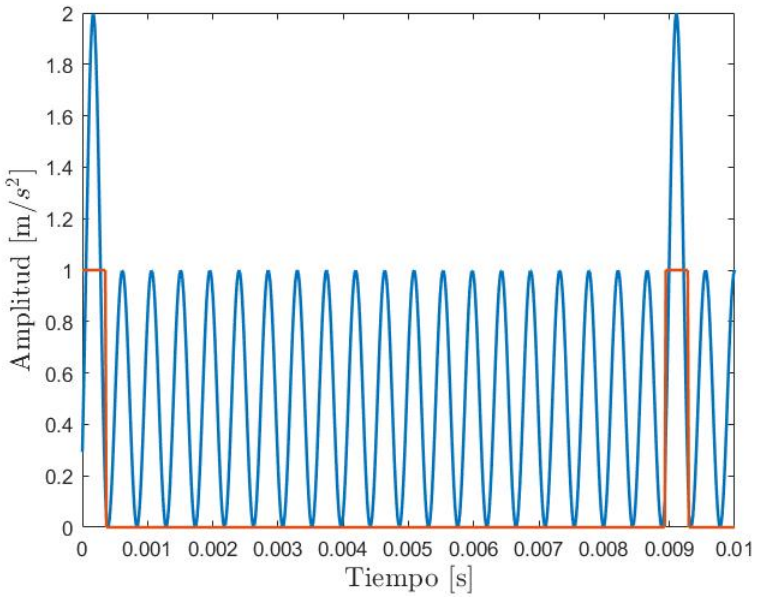

Figura 3.11: Forma de onda del engrane con falla local en el planeta

En la **Figura 3.12** se puede observar la señal del planeta con falla siendo modulada por el movimiento del carrier, donde claramente existe un aumento en la amplitud a intervalos regulares proporcionales a los ciclos de giro del planeta.

En la **Figura 3.13** se puede observar una comparación entre las señales generadas por cada uno de los planetas cuando son moduladas por el carrier, en donde se observa claramente la señal del planeta que tiene problemas. Se suman las señales, se agregan las componentes asociadas a las frecuencias de giro de entrada y de salida, y se utiliza la transformada de Fourier para estudiar el contenido frecuencial de esta señal, el cual se puede ver en la **Figura 3.14**, en donde se puede observar claramente la aparición de varias componentes armónicas de 111.9 [Hz] con bandas laterales a 44.75 [Hz], lo que corresponde a la señal de la falla a una frecuencia de 1/20 de la frecuencia de engrane siendo modulada por la señal del carrier. Es importante indicar que en el espectro se pueden visualizar más de 10 armónicos claramente.

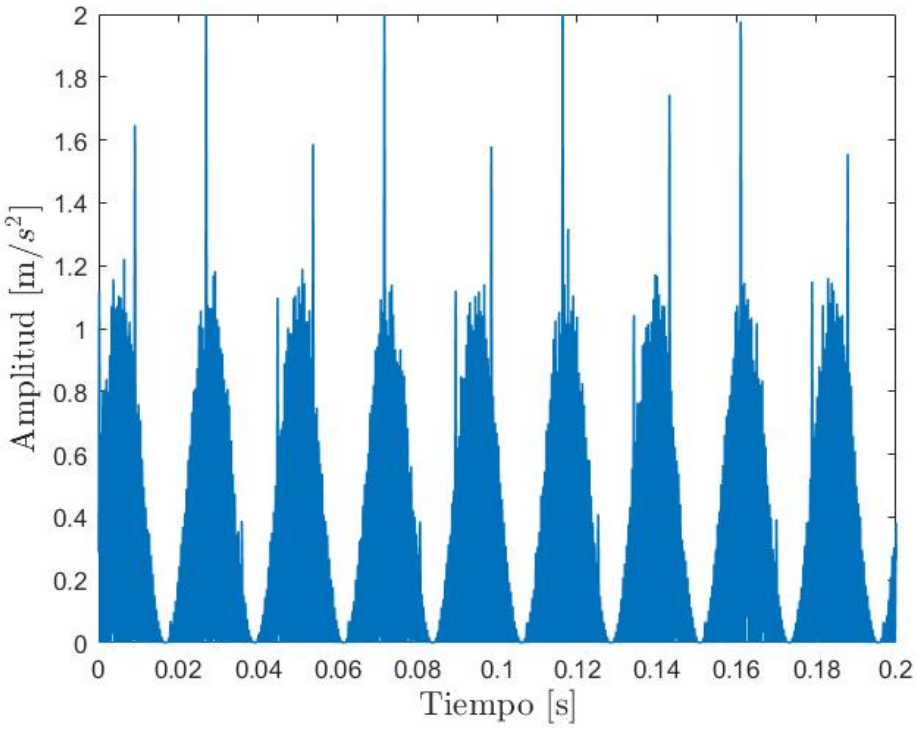

Figura 3.12: Forma de onda del engrane con falla local en el planeta

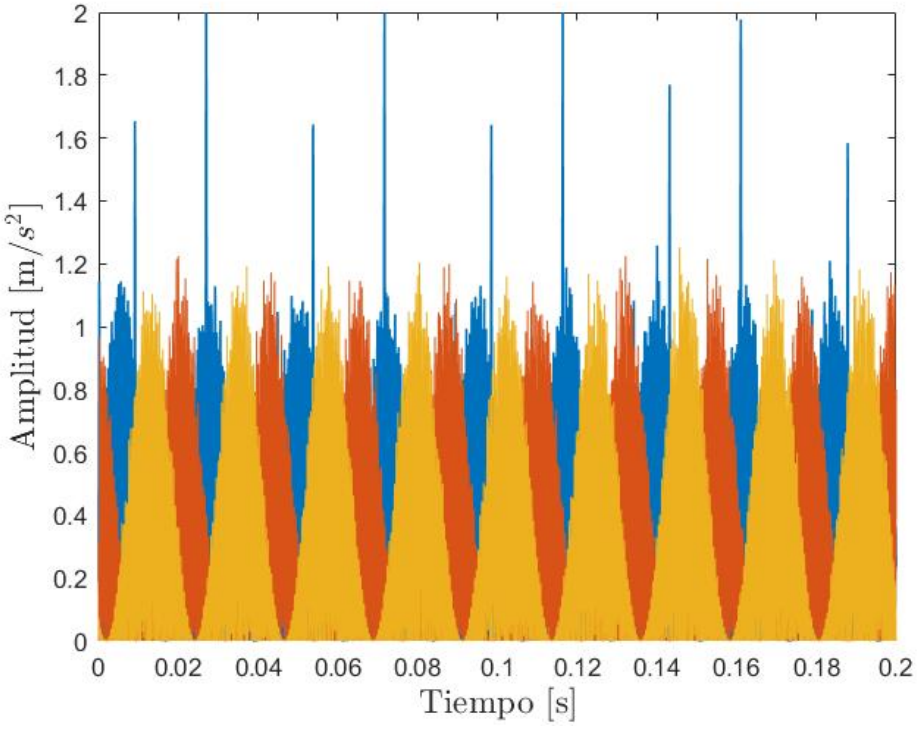

Figura 3.13: Comparación de señales moduladas por el carrier

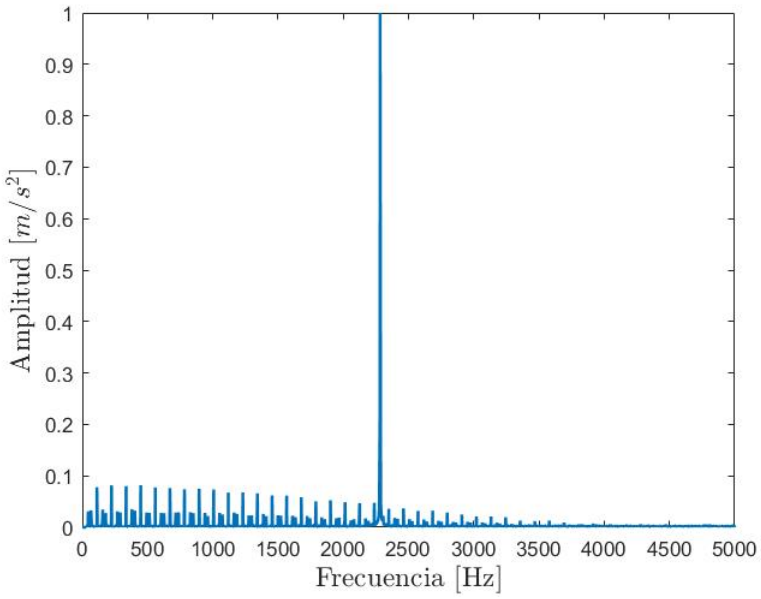

Figura 3.14: Espectro de la señal cuando existe un problema de falla local en uno de los planetas

La amplitud máxima de la aceleración durante el engrane con la grieta en la base en comparación con el equipo sin problemas *debe ser proporcional al tamaño de la grieta*, por lo que a la hora de entrenar el modelo de la *machine learning* y en general al utilizarlo para representar fallas en equipos reales se debe realizar un estudio de cómo se relacionan proporcionalmente las amplitudes de las señales, y también cómo se pueden observar en el *Espectro de Fourier*.

Es importante indicar que *si se cambia la fase de la señal cuadrada se estará cambiando el diente del planeta que está teniendo problemas*, lo cual puede generar variaciones en la señal temporal, sin embargo, el contenido frecuencial de la señal no va a cambiar, *lo que reafirma que el estudio de este tipo de fenómenos desde el dominio de la frecuencia se acomoda bastante bien para su análisis.*

Simulación de falla local en el sol

A la hora de simular una falla local en el sol se tiene que tener en consideración que el sol engrana con los tres planetas, por lo que por cada giro completo que realiza el sol se genera una vez un impacto en cada uno de los planetas, el cual se debe modular a la velocidad de giro del carrier. Utilizando la misma metodología que la mostrada en la sección anterior se utiliza una onda cuadrada para aumentar la amplitud de la señal asociada al engrane de los componentes a las frecuencia de falla del sol. En la **Figura 3.15** se puede observar una comparación del impacto generado en los tres engranajes debido a la falla en un engranaje del sol.

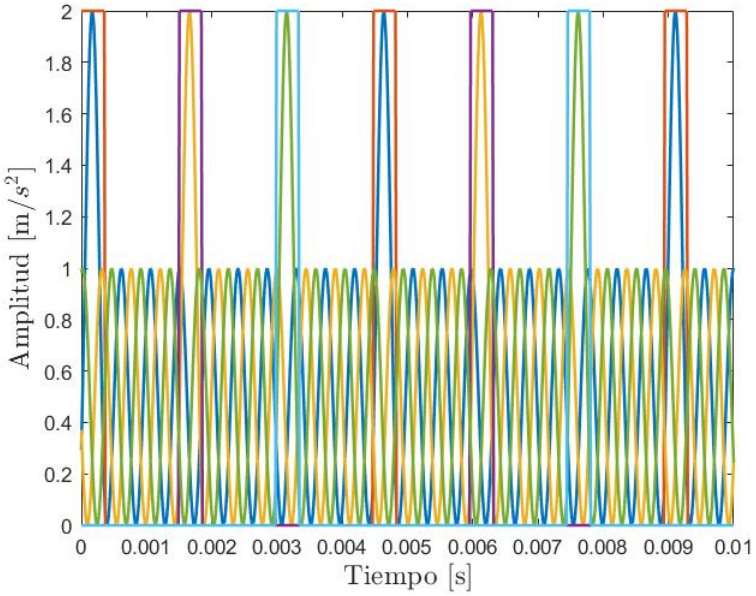

Figura 3.15: Comparación de la vibración de 3 planetas cuando hay una falla local en el sol.

En la **Figura 3.16** se puede observar las tres señales siendo moduladas por el carrier, en donde se ve claramente que a diferencia del caso de falla en un planeta, *en este caso una falla en el sol genera problemas en el engrane de todos los planetas.* En la **Figura 3.17** se puede observar el espectro de la suma de las 3 señales moduladas por el carrier, en donde se ve claramente una componente asociada a la frecuencia de engrane encontrada también en los casos anteriores, correspondiente de 2282 [Hz] y una componente a 671 [Hz] con una gran cantidad de armónicos, el cual aparece en la **Tabla 3.2** asociada a fallas en el sol, y con bandas laterales a 268.5 [Hz], que corresponde a la frecuencia de giro del sol.

En esta última figura aparecen componentes tanto del planeta como del sol y de sus fallas, lo cual es sumamente útil pues se observa claramente la información disponible a través del contenido frecuencial. Tal como se indicó en la sección anterior, cuando se quiera generar los datos para el entrenamiento del *algoritmo ANN se tiene que regular la magnitud de la aceleración en el momento del impacto de manera proporcional al desarrollo de la grieta en el sol.*

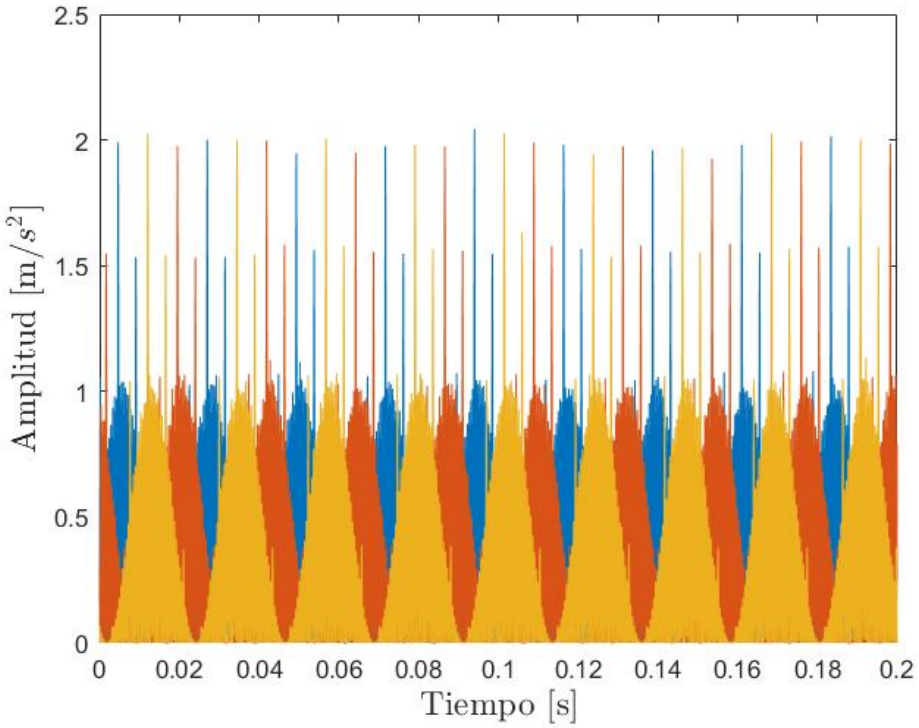

Figura 3.16: Señales de los planetas siendo moduladas con una falla local en el sol.

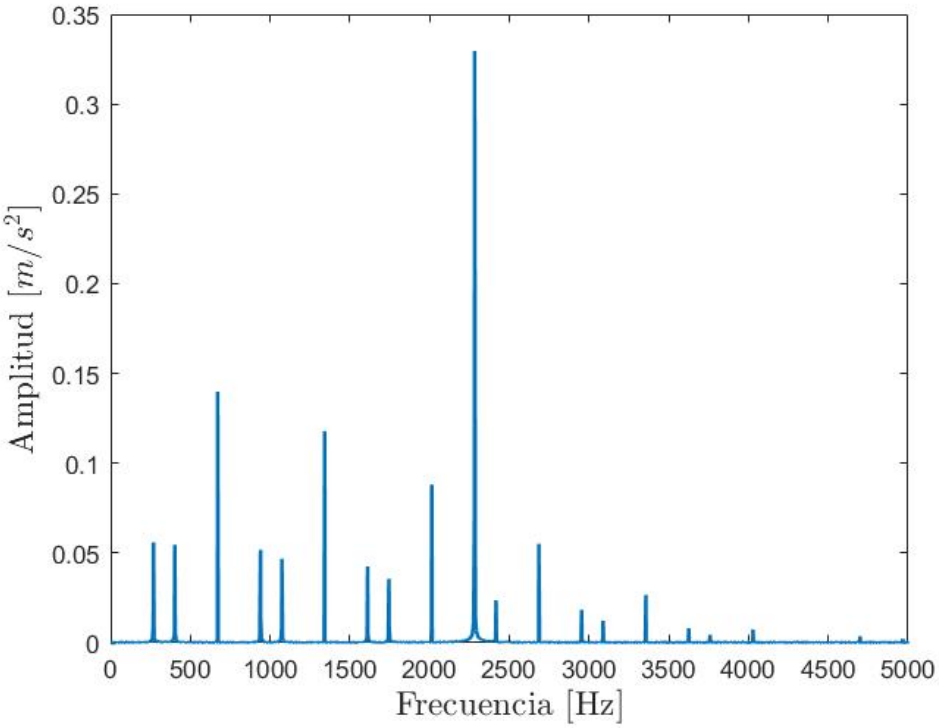

Figura 3.17: Espectro de la suma de las señales moduladas debido a una falla local en el sol.

Simulación de falla local en el anillo exterior

A la hora de simular una falla en el aro exterior se tiene que tener en cuenta que la posición de la falla en el aro está fija respecto a la posición del sensor, por lo que todos los impactos ocurrirán en una misma posición y por lo tanto a una amplitud no modulada. Utilizando la metodología de las secciones anteriores se utiliza una onda cuadrada para aumentar la amplitud de la señal de engrane de los planetas a las frecuencias de falla del sol. En la **Figura 3.18** se puede observar una comparación del impacto generado en un planeta debido a una falla en el aro exterior.

En la **Figura 3.19** se puede observar las tres señales de los planetas siendo moduladas por el carrier, en donde al igual que en el caso del sol, *una falla en el aro exterior genera impactos en todos los planetas*. En la **Figura 3.20** se puede observar el espectro de la suma de las 3 señales moduladas por el carrier, en donde se ve claramente la frecuencia de engrane encontrada en los casos anteriores, correspondiente a 2282 [hz] y una componente a 134.4 [hz] y sus armónicos, la cual aparece en la **Tabla 3.2** asociada a fallas en el aro exterior.

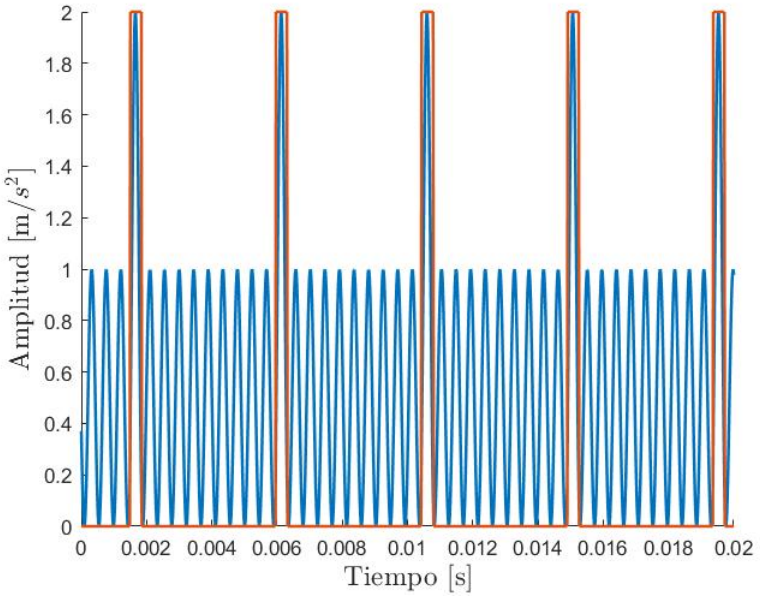

Figura 3.18: Vibración del segundo planeta cuando hay una falla local en el aro exterior.

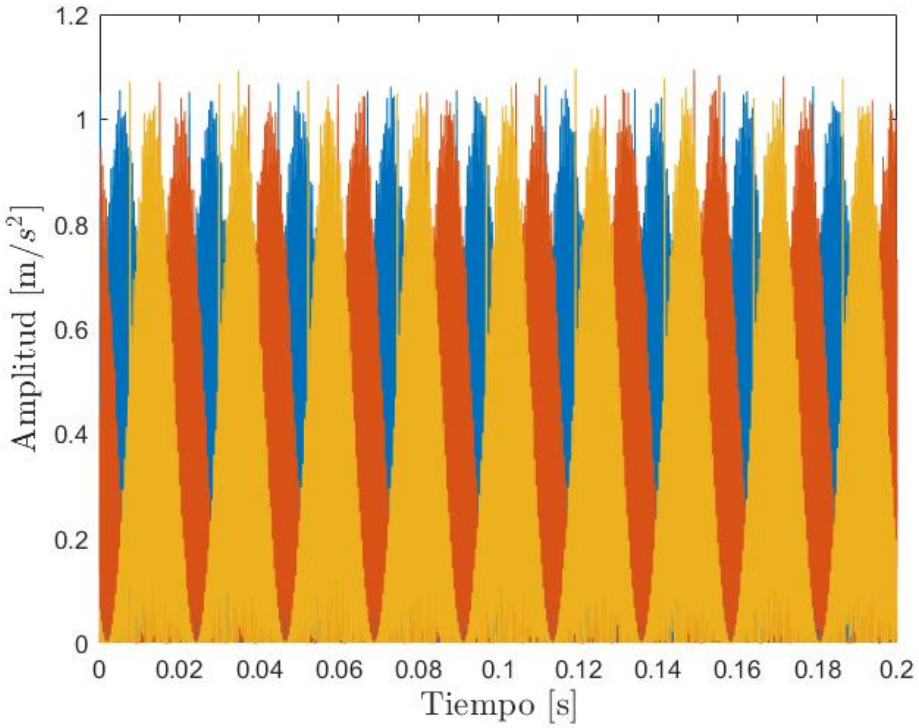

Figura 3.19: Comparación de las señales de los planetas moduladas con falla en el aro exterior.

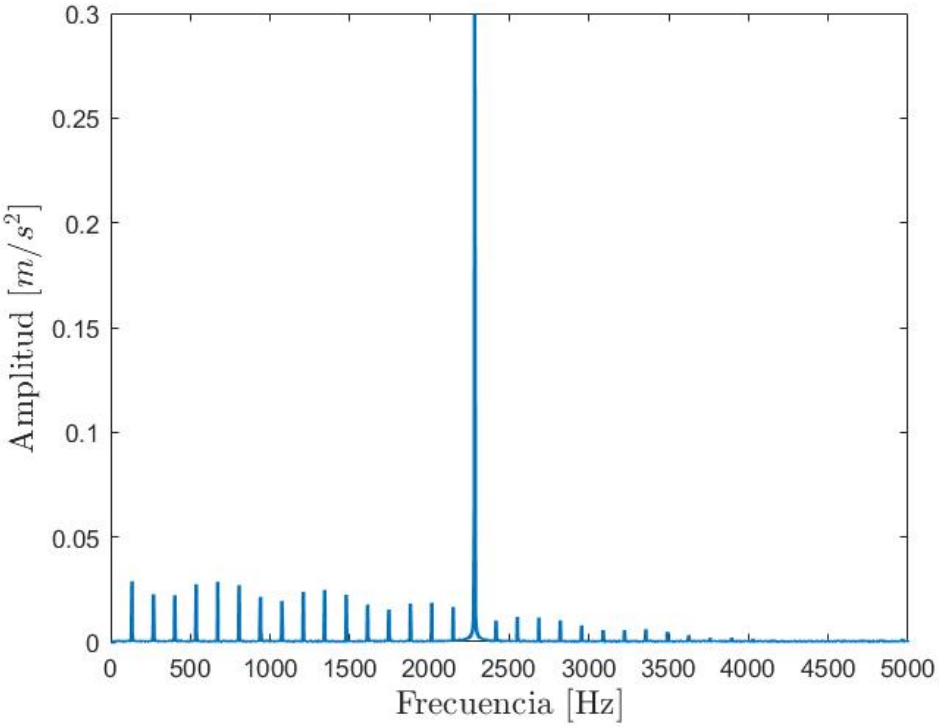

Figura 3.20: Espectro de la suma de las señales moduladas con falla en el aro exterior.

3.1.4. Variabilidad en las frecuencias características y las amplitudes

Durante el funcionamiento de un equipo mecánico es normal encontrar variaciones en la frecuencia de giro o de funcionamiento, ya sea porque los motores eléctricos no siempre proveen una frecuencia de giro constante, porque la carga del sistema varía con el tiempo u otras razones. Es debido a esto que en la simulación se considera el uso de una variable asociada a amplitud de la variación de las frecuencias asociadas a cada uno de los componentes del reductor, en donde una frecuencia característica está descrita según la **Ecuación 3.1**, en donde a la frecuencia teórica se le suma una componente asociada a una *Distribución Normal Estándar*, convirtiéndola en una variable aleatoria.

$$f = f_x + f_{var_x} \cdot N(1,0) \tag{3.1}$$

Los parámetros de amplitud son los valores utilizados para *modificar la amplitud relativa* entre los componentes de las distintas señales que componen al modelo y para regular las amplitudes de la *Transformada de Fourier*. En el modelo existen 3 señales principales, según la siguiente lista:

- Señal del eje de entrada.
- Señal del eje de salida.
- Señal de planeta engranando con el sol y la corona, la cual puede incluir la falla.

Por lo que se utilizarán tres parámetros para regular la amplitud relativa entre cada una de estas componentes y su variación, tal como se indica en la **Ecuación 3.2**. Finalmente, de la misma manera que con las frecuencias, se modifica la amplitud de todas las componentes del *Espectro de Fourier Normalizado*, de tal manera que el máximo de esta no sea siempre 1, según lo que se indica en la **Ecuación 3.3**.

$$\bar{A_x} = A_x + V_x \cdot N(1,0) \tag{3.2}$$

$$T\bar{D}F_x = TDF_x \cdot (TDF_A + TDF_{V_x} \cdot N(1,0)) \tag{3.3}$$

En la **Figura 3.21** se puede observar un ejemplo de simulación en donde se entregan distintos valores de a la amplitud relativa de las componentes de la señal, pero no se modifica la amplitud del *Espectro de Fourier Normalizado*, según la **Tabla 3.5**, mientras que en la **Figura 3.22** se observa la misma señal pero con un factor multiplicando a las componentes del *Espectro de Fourier Normalizado* según lo indicado en la **Tabla 3.6**.

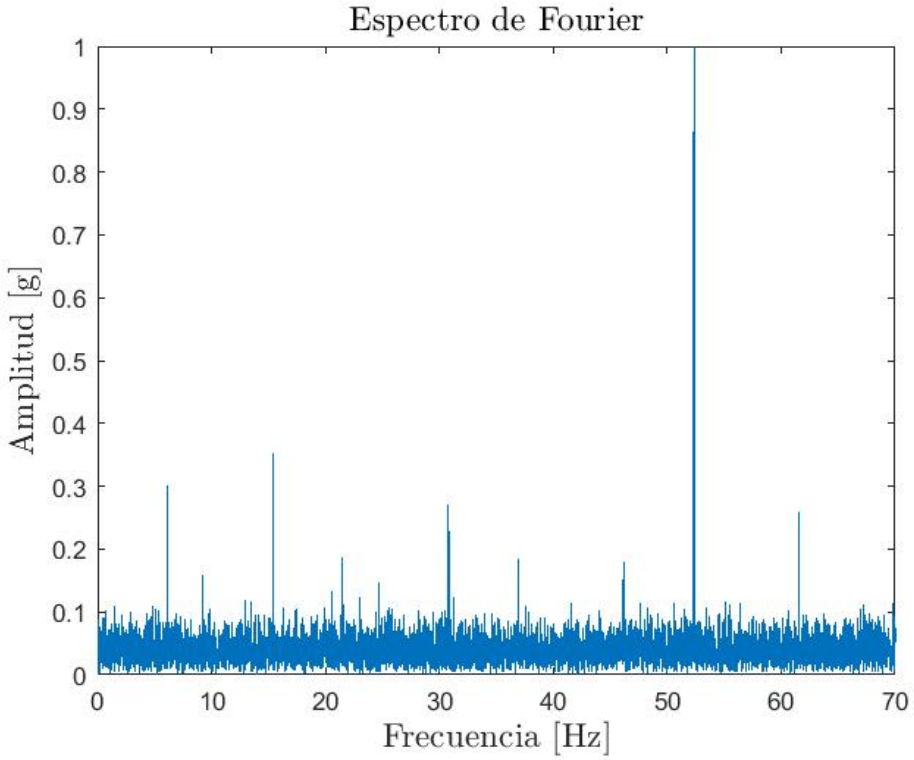

Figura 3.21: Ejemplo del espectro de Fourier Normalizado.

Tabla 3.5: Parámetros de la simulación para el caso falla en planetas

Amplitud de la componente en la señal de giro de entrada [S/U]	A_s	0.2
Amplitud de la componente en la señal de giro de salida [S/U]	A_c	0
Amplitud de la componente en la señal de engrane [S/U]	A_m	3
Amplitud de la variación de la componente en la señal de giro de entrada [S/U]	V_s	0.3
Amplitud de la variación de la componente en la señal de giro de salida [S/U]	V_c	0.05
Amplitud de la variación de la componente en la señal de engrane [S/U]	V_m	0.5
Amplitud generalizada de la TDF [S/U]	TDF_{A_m}	1
Amplitud de la variación generalizada de la TDF [S/U]	TDF_{Vm}	0

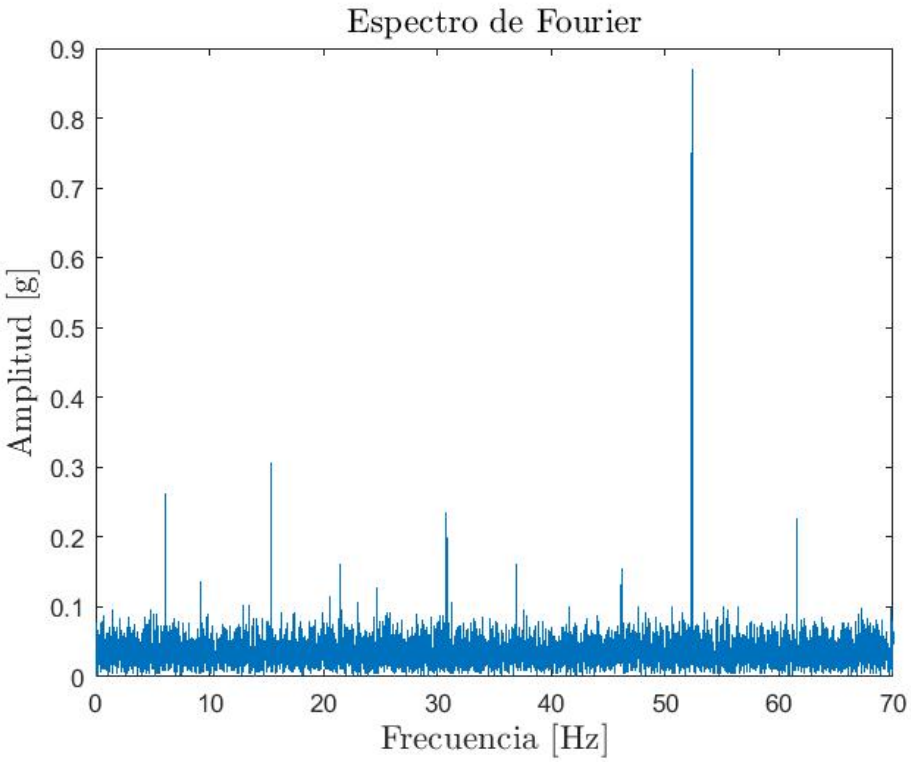

Figura 3.22: Ejemplo del espectro de Fourier Normalizado.

Tabla 3.6: Parámetros de la simulación para el caso falla en planetas

Amplitud de la componente en la señal de giro de entrada [S/U]	A_s	0.2
Amplitud de la componente en la señal de giro de salida [S/U]	A_c	0
Amplitud de la componente en la señal de engrane [S/U]	A_m	3
Amplitud de la variación de la componente en la señal de giro de entrada [S/U]	V_s	0.3
Amplitud de la variación de la componente en la señal de giro de salida [S/U]	V_c	0.05
Amplitud de la variación de la componente en la señal de engrane [S/U]	V_m	0.5
Amplitud generalizada de la TDF [S/U]	TDF_{A_m}	0.85
Amplitud de la variación generalizada de la TDF [S/U]	TDF_{Vm}	0.15

3.2. Normalización a la frecuencia del carrier

Una manera de visualizar los datos de la transformada de Fourier es normalizando las frecuencias por alguna de las frecuencias características del equipo, por ejemplo, *la frecuencia de giro del carrier*, que corresponde a la del eje de salida del equipo. De esta manera las frecuencias características se pueden estudiar en función de los órdenes de la frecuencia de giro del carrier. En la **Figura 3.23** se puede observar una de las simulaciones del reductor con falla en uno de sus planetas y normalizado a la frecuencia de giro del carrier.

Para hacer esto con los equipos reales normalmente se utiliza un equipo para medir las RPM del eje, como un tacómetro o un estroboscopio, que es lo que se utilizará en el montaje experimental. Esto es importante debido a que el equipo puede variar sus RPM durante su funcionamiento, lo que puede llegar a dificultar todo tipo de análisis y comparación, y se puede solucionar con este tipo de procesamiento.

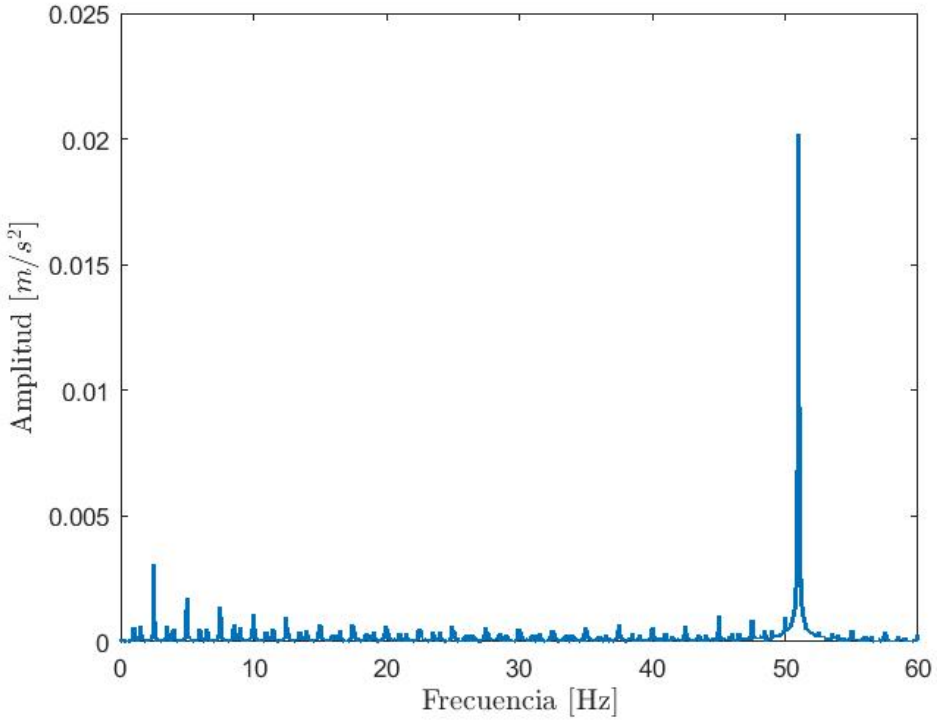

Figura 3.23: Transformada de Fourier normalizada de simulación de engranaje planetario con falla en el planeta

3.3. Resumen y análisis del modelo fenomenológico

En este capítulo se ha presentado un modelo para la generación de vibraciones mecánicas en la carcasa de un reductor planetario de una etapa utilizando un modelo fenomenológico con un programa en MATLAB. El programa permite calcular las frecuencias características del equipo utilizando como datos la frecuencia de giro del portador y la cantidad de dientes de los elementos principales del reductor *correspondiente al sol, el aro exterior y los planetas.* Debido a que la idea del modelo es poder generar datos para entrenar una *red neuronal*, se deja una determinada cantidad de parámetros modificables, para ajustar los datos según se estime conveniente, en función de los resultados de los datos experimentales y los métodos de procesamiento que se les aplique. En la **Tabla 3.7** se pueden observar todos los parámetros que se pueden modificar en el modelo fenomenológico, además, en el **Apéndice A** se pueden observar los códigos para la simulación de los datos y gráficos mostrados en este capítulo, correspondientes a los siguientes casos:

- Caso del equipo sin fallas.
- Caso con falla local en uno de los planetas debido a una grieta.
- Caso de falla local en el sol debido a una grieta.
- Caso de falla local en el aro exterior debido a una grieta.

Tabla 3.7: Parámetros modificables en el modelo fenomenológico.

Parámetro	Descripción	Utilidad
z_r	Número de dientes del anillo exterior	Modifica las frecuencias características de cada modo de falla
z_s	Número de dientes del sol	Modifica las frecuencias características de cada modo de falla
z_p	Número de dientes del planeta	Modifica las frecuencias características de cada modo de falla
f_c	Frecuencia de giro del carrier	Modifica las frecuencias características de cada modo de falla
f_{var_m}	Variación de la frecuencia de engrane	Modifica la variabilidad de la frecuencia de engrane con una distribución normal
f_{var_s}	Variación de la frecuencia de giro del sol	Modifica la variabilidad de la frecuencia de giro del sol con una distribución normal
f_{var_c}	Variación de la frecuencia de giro del carrier	Modifica la variabilidad de la frecuencia de giro del carrier con una distribución normal
f_{var_p}	Variación de la frecuencia de falla local	Modifica la variabilidad de la frecuencia a la que ocurre la falla local
A_p	Amplitud de la falla local	Modifica la amplitud de la señal cuadrada de la falla local
A_m	Amplitud de la señal de engrane	Modifica la amplitud de la señal modulada del planeta
A_s	Amplitud de la frecuencia de giro de entrada	Modifica la frecuencia de la señal de giro de entrada
A_c	Amplitud de la frecuencia de giro de salida	Modifica la frecuencia de la señal de giro de salida
V_m	Variación de la amplitud de la señal de engrane	Modifica la variabilidad de la amplitud de la señal modulada del planeta
V_s	Variación de la amplitud de la señal de giro de entrada	Modifica la variabilidad de la amplitud de la señal de giro de entrada
V_c	Variación de la amplitud de la señal de giro de salida	Modifica la variabilidad de la amplitud de la señal de giro de salida
E_m	Magnitud del ruido en la señal de giro de engrane	Modifica la amplitud del ruido de la señal de engrane
E_s	Magnitud del ruido en la señal de giro de entrada	Modifica la amplitud del ruido de la señal de entrada
E_c	Magnitud del ruido en la señal de giro de salida	Modifica la amplitud del ruido de la señal de salida
TDF_{A_m}	Amplitud generalizada de la TDF	Modifica la amplitud máxima de la TDF
TDF_{V_m}	Variación de la amplitud generalizada de la TDF	Modifica la variaicón de la amplitud máxima de la TDF

Capítulo 4

Mediciones experimentales

En esta sección se explicará cómo se realizaron las mediciones experimentales en el reductor, se describirá el equipo utilizado para generar las vibraciones y para realizar las mediciones, la manera en la que se generaron las fallas, las condiciones de operación durante las mediciones, y los parámetros de las mediciones, además se describirán los procesamientos aplicados a los datos experimentales.

4.1. Descripción del montaje

El equipo utilizado es un motor de partida, el cual se puede observar en la **Figura 4.1**, y que dentro del vehículo cumple con la función de generar el torque inicial para realizar la partida del motor de combustión. Debido a que es necesario un gran torque para comenzar el movimiento del motor a combustión, es normal que en determinados modelos de motores de partida posean reductores con engranajes planetarios, los cuales se caracterizan por tener buenas relaciones de transmisión entre la entrada y la salida, con la ventaja de tener un tamaño relativamente pequeño comparado con otros tipos de reductores, lo que los hace excelentes alternativas para esta aplicación.

El motor de partida consiste básicamente en un *Motor de imanes permanentes, un reductor planetario y un accionador*. En la **Tabla 4.1** se muestran los valores nominales de funcionamiento del motor y las características de los engranajes del reductor fueron descritas en la **Tabla 3.1**. El accionador fue eliminado del equipo debido a que su accionamiento genera un impacto que introduce ruido a la señal.

Figura 4.1: Motor de partida utilizado en el montaje experimental.

Tabla 4.1: Parámetros nominales del motor de partida.

Parámetro	Magnitud	Unidad
Modelo	MP - 801	[N/A]
Resistencia	11.778	$[m\Omega]$
Potencia máxima	1.141	[kW]
Corriente	269	[A]
Torque	8,5	[Nm]
Velocidad de giro	1551	[RPM]

Para hacer funcionar el motor se conectan sus terminales a una batería de auto, la cual entrega un voltaje constante de 12 [V] y tiene una capacidad de 26 [Ah].

Para aplicar carga sobre el eje del motor se utilizó una variante de un *freno Prony* según la foto que se muestra en la **Figura 4.2**, el cual consiste en un marco hecho con perfiles cuadrados de acero laminado, al cual se le apernan un par de abrazaderas con forma de 'U', a las cuales se atan las eslingas que aplicarán una carga sobre el eje de salida del motor de partida. La fuerza aplicada sobre el eje se regula con la posición de las abrazaderas en el marco, y para medir la carga aplicada sobre el eje se conecta un dinamómetro digital entre la abrazadera y la eslinga.

El motor de partida está apernado a una tabla de madera utilizando uniones metálicas en 90°. La tabla de madera tiene piezas de goma en cada una de sus esquinas, las cuales sirven para evitar cualquier tipo de ruido proveniente de otras fuentes. En la **Figura 4.2** se pueden observar tanto el montaje del *freno Prony* como la tabla de madera y la unión con el motor de partida

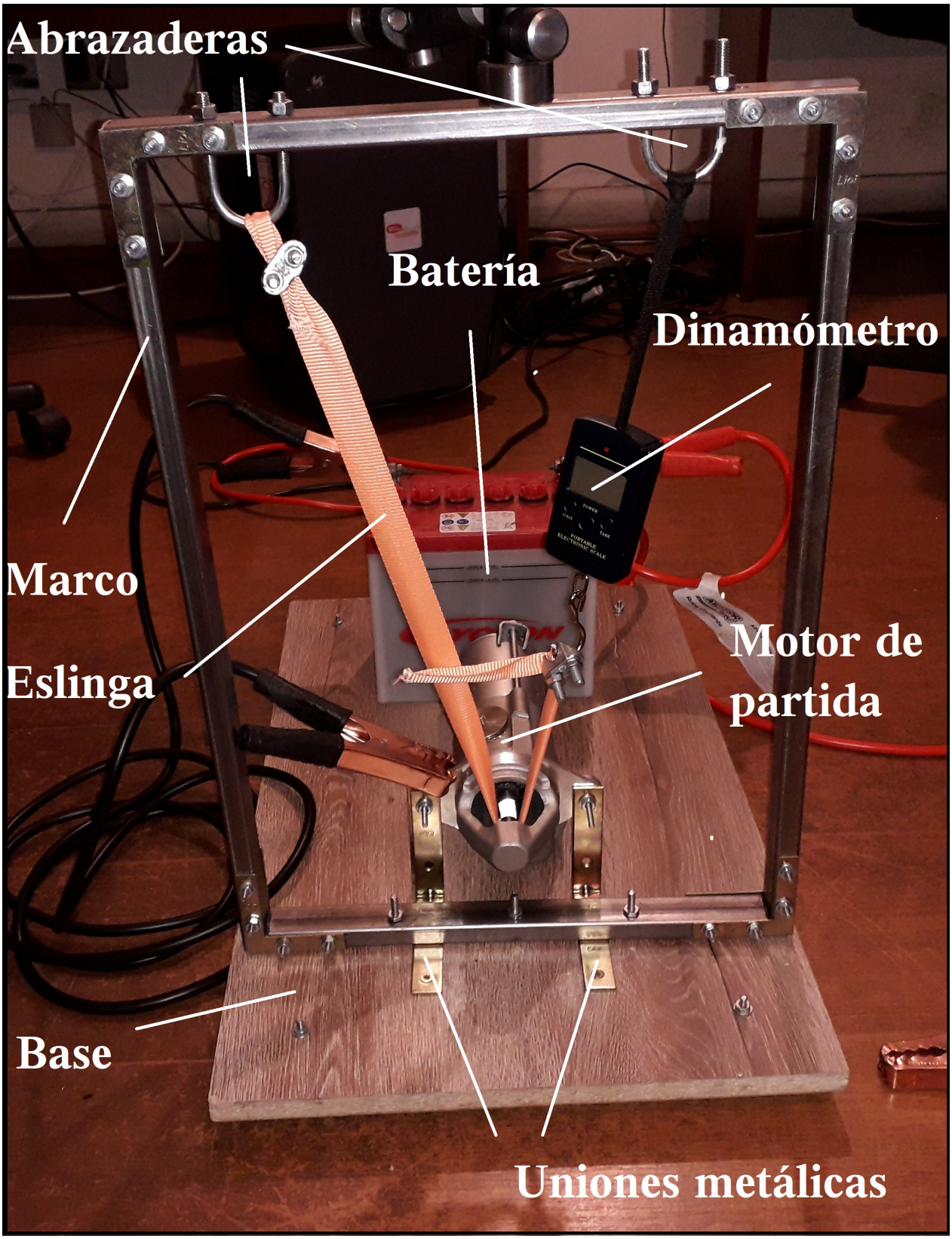

Figura 4.2: Motor de partida utilizado en el montaje experimental.

4.2. Equipo de medición de vibraciones

El equipo de adquisición de datos utilizado para la medición de vibraciones mecánicas es el *FALCON* de la marca francesa *ONEPROD*, el cual se puede observar en la **Figura 4.3**. Este equipo que posee un sensor inalámbrico con las características que se indican en la **Tabla 4.2**. En la carcasa de la sección más cercana al reductor del motor de partida se adhiere una base metálica de acero con el pegamento *Loctite*, tal como se indica en la **Figura 4.4**. La base posee una rosca de medida M6, la cual calza con la base del sensor, formando una unión rígida. Esta base es necesaria debido a que la carcasa del motor de partida está hecha de aluminio, por lo que la base magnética del sensor no se adhiere a la superficie del motor. El sensor es triaxial, sin embargo, para el análisis se considerará únicamente la dirección *radial vertical*, tomando como referencia la base de madera del montaje. Al exportar los datos estos se encuentran en formato UFF (Universal File Format).

Tabla 4.2: Parámetros de funcionamiento del sensor inalámbrico FALCON.

Propiedad	Magnitud	Unidad
Sensibilidad	100	[mV/g]
Diámetro	40	[mm]
Alto	115	[mm]
Rango de medición	0 / 80	[g]
Resistencia a impactos	5000	[g]
Frecuencia de muestreo	51,2	[kHz]
Temperatura de operación	-20 / + 60	[°C]
Montaje	Perforación roscada M6	

Figura 4.3: Equipo inalámbrico FALCON.

Figura 4.4: Montaje del FALCON.

4.3. Inclusión de falla

Para la inclusión de fallas en el modelo se considera imprimir grietas en los engranajes utilizando un *dremel*, una herramienta rotatoria multiuso, con un disco de corte, la cual se puede observar en la **Figura 4.5**. Se consideran 3 niveles de severidad para las fallas de cada equipo, las cuales se describen en la **Tabla 4.3** por lo que en *total la cantidad de casos estudiados corresponden a 10*, 1 caso en donde todos los componentes funcionan adecuadamente sin ningún tipo de grieta o falla, y 3 modos de falla para cada uno de los 3 componentes analizados en este estudio. En la **Figura 4.6** se puede observar el sol del equipo con una grieta de severidad nivel 2.

Figura 4.5: Dremel con disco de corte

Figura 4.6: Sol del reductor con nivel de falla 2.

Nivel de falla	Características
	No hay ninguna grieta o falla
1	La grieta tiene una dimensión de 1/6 del espesor del diente
2	La grieta tiene una dimensión de 1/3 del espesor del diente
3	Se simula desprendimiento de material

Tabla 4.3: Niveles de severidad en las fallas aplicadas a los engranajes

4.4. Procedimiento de la toma de datos

A continuación se describe el procedimiento para la toma de mediciones experimentales:

1. Se abre el motor y se le generan las fallas a los componentes correspondientes.
2. Se cierra el motor y se aseguran todos los pernos fijamente.
3. Se prende el dinamómetro digital y se aprietan los pernos de las abrazaderas hasta que el dinamómetro marque 11 [kg]
4. Se conecta la tierra de la batería a la carcasa del motor.
5. Se prende el sensor y aperna en la base dispuesta en la carcasa del motor.
6. Se comienza la medición apretando el botón correspondiente en el FALCON y se conecta al lado positivo de la batería al contacto para dar inicio al funcionamiento del motor de partida.
7. El contacto se mantiene durante 4 a 5 segundos.

Se debe tener en cuenta que el motor debe reposar a lo menos 2 minutos entre cada medición debido a un rápido aumento en su temperatura durante su uso. También es importante tener en cuenta que el motor no está diseñado para soportar su uso continuo durante demasiado tiempo, por lo que las mediciones no pueden ser más largas que lo indicado, de lo contrario es posible que el equipo se dañe o que su temperatura aumente peligrosamente. *Un aumento excesivo en la temperatura del equipo genera condiciones de operación distintas, lo cual es algo que se desea evitar.*

Se debe tener en consideración que la batería utilizada se descarga, lo que puede afectar a las condiciones de operación de las mediciones, por lo que entre cada conjunto de mediciones asociado a un modo de falla se utilizó un *cargador de baterías para autos de marca Einhell, el cual se puede observar en la Figura 4.7*, con el cual se aseguró que el voltaje de la batería se mantuviera entre 12 [V] y 13 [V] durante todas las mediciones. Para lograr esto se tuvo que cargar la batería entre cada conjunto de mediciones.

Figura 4.7: Cargador de baterías marca Einhell

4.5. Manejo de datos y procesamiento de señales

Para las mediciones se utilizaron los parámetros indicados en la **Tabla 4.4**. El ancho de la eslinga que aplica carga con el *freno Prony* es de *2 [cm]* y la carga utilizada en todas las mediciones es de *10 [kg]*.

Tabla 4.4: Parámetros de adquisición de datos

Propiedad	Magnitud	Unidad
Tiempo de medición	10	[s]
Frecuencia de adquisición	51200	[Hz]
Filtro pasa alta	2	[Hz]

Respecto a las mediciones experimentales se debe decir que el motor de partida no está diseñado para ser utilizado por periodos de tiempo mayores a unos cuantos segundos, por lo tanto, para poder tener suficientes datos se realizaron 15 conjuntos de mediciones de 4 [s] de largo aproximadamente, permitiendo reposar el equipo 10 minutos entre cada medición, principalmente para disminuir su temperatura. Para aprovechar al máximo la cantidad de datos se cortaron las formas de onda tomando *102400 puntos de cada una*, por lo que para cada modo de falla estudiado se terminó con *15 conjuntos de mediciones de 2 [s] aproximadamente*. Dada la frecuencia de adquisición indicada en la **Tabla 4.4** y el tiempo de cada conjunto la *resolución en frecuencia de los espectros va a ser de 0.5 [Hz]*, lo cual es suficiente para realizar el análisis. Estos parámetros son relevantes debido a que para que los datos simulados y experimentales sean comparables las señales deben tener la misma cantidad de puntos y frecuencia de adquisición en el tiempo.

4.5.1. Envolvente e identificación de la frecuencia del carrier

Uno de los problemas del montaje experimental es que no se puede controlar la velocidad de giro del equipo, la cual varía debido al estado de la batería, la cual se fue descargando a medida que se realizaron las mediciones . Para solucionar esto se aprovechó una de las características de los reductores planetarios, los cuales se distinguen porque sus vibraciones están moduladas a la frecuencia del carrier, de tal manera que con la envolvente de la señal es posible encontrar la frecuencia de giro con bastante exactitud. De esta manera se pueden normalizar los espectros respecto a la frecuencia de giro del carrier para que todas las mediciones puedan ser más comparadas más fácilmente, teniendo en cuenta que posteriormente estos datos van a ser la entrada de una *red neuronal*. En caso de no realizar este procedimiento el proceso de analizar dos conjuntos de datos con frecuencias de giro distinta puede ser mucho más complejo a través de la *Transformada de Fourier*, principalmente debido a las frecuencia características del equipo y de las fallas serán distintas. *El método que se utilizará para calcular la envolvente es utilizando la función envelope en MATLAB, la cual utiliza interpolación spline sobre los máximos locales separados por al menos 100 muestras.*

En la **Figura 4.8** se puede observar la forma de onda de uno de los casos de planetas con *nivel de falla 3 según la Tabla 4.3*, y sobre esta la envolvente de la señal, en donde se observa claramente el fenómeno de modulación. En la **Figura 4.9** se puede observar el *espectro de Fourier* de la envolvente de la señal, en donde se marca la frecuencia de giro del carrier cercana a los 37 [Hz] y su primer armónico alrededor de los 74 [Hz]. Para calcular esta frecuencia se realiza un promedio ponderado utilizando la frecuencia del máximo entre los 25 y 45 [Hz] y sus dos componentes colindantes.

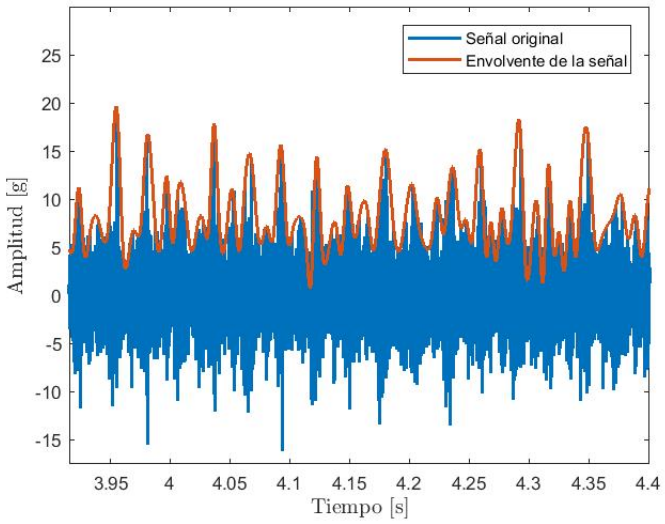

Figura 4.8: Comparación de la señal original y su envolvente

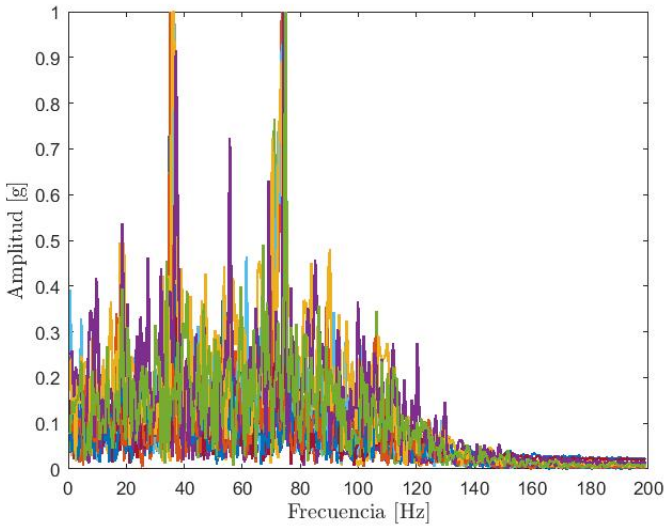

Figura 4.9: Espectros de las envolventes

En el **Apéndice B** se pueden encontrar los códigos utilizados para el procesamiento. Los códigos se organizan de tal manera que en primera instancia extraen los datos desde los archivos en formato UFF y son guardados en el formato de los datos de MATLAB en una carpeta determinada una vez que son procesados según lo que se indicó. Una vez guardada la forma de onda se aplican los códigos para el cálculo de la frecuencia de giro del portador para cada conjunto de mediciones, cuyos valores son guardados en otro archivo, el cual será utilizado en los procesamientos que se describen en los capítulos posteriores **para dividir el eje de las frecuencias de los espectros, pasando de unidades [Hz] a [ordenes], que corresponden a los múltiplos de la frecuencia de giro del carrier.**

Además, debido a que las vibraciones en un reductor planetario están moduladas a la frecuencia del portador, la envolvente no puede utilizarse únicamente como herramienta para encontrar la frecuencia de giro de este, sino que también para la búsqueda de defectos en el equipo. En el **Apéndice J** se pueden observar las *Envolventes* de 3 ejemplos de cada modo de falla comparado el caso sin fallas, además de comparación entre los espectros de un mismo modo de falla pero con distintos niveles de severidad.

4.5.2. Transformada de Fourier

La transformada de Fourier es normalmente el primer procesamiento que se aplica a los datos pues permite estudiar el contenido frecuencial de la señal [**5**], el cual podría tener diferencias ante los distintos modos de falla, pudiendo permitir la identificación de cada uno de los casos. La resolución en frecuencia del *espectro de Fourier* es de 0.5 [Hz], y se calcula utilizando la **Ecuación 4.1**. En el **Apéndice B** se muestran los códigos utilizados para el cálculo de la *transformada de Fourier*, en el cual ser considera la normalización del eje de las frecuencias por la velocidad de giro del carrier, tal como se indicó en la sección anterior, **además, se divide la amplitud del espectro por el máximo del espectro, de tal manera que la amplitud también se encuentre normalizada entre 0 y 1.**

$$F_{resolution} = \frac{F_s}{N_{puntos}} \tag{4.1}$$

En donde

- $F_{resolution}$: Resolución del espectro. [Hz]
- F_s: Frecuencia de adquisición [Hz]
- N_{puntos}: Número de puntos tomados durante la medición [S/U]

En las **Figuras 4.10 y 4.11** se puede observar un ejemplo de *Transformada de Fourier* normalizada tanto en frecuencia como en A amplitud, en la cual se observa claramente como aparecen componentes a múltiplos de la frecuencia de giro del carrier, indicando que la normalización en la frecuencia se llevó a cabo apropiadamente. En el **Apéndice F** se pueden observar los *espectros de Fourier* de 3 ejemplos de cada modo de falla comparado el caso sin fallas, además de comparación entre los espectros de un mismo modo de falla pero con distintos niveles de severidad.

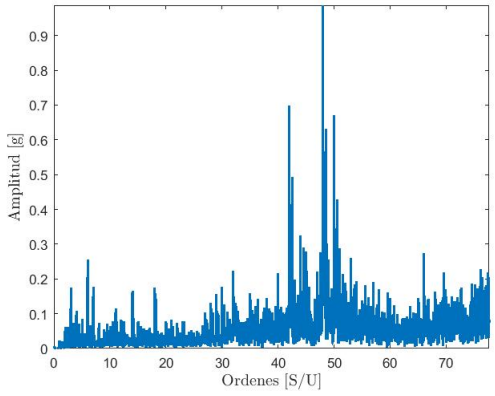

Figura 4.10: Espectro normalizado.

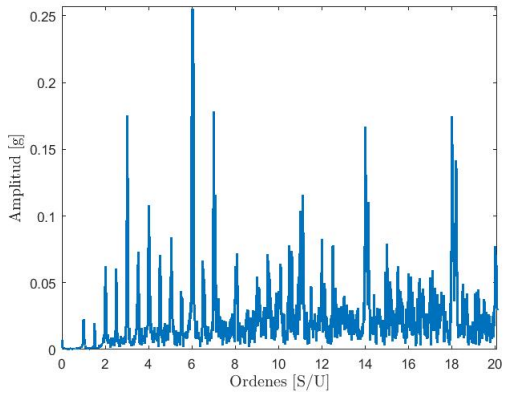

Figura 4.11: Espectro normalizado (Zoom).

4.5.3. Aplicación del filtro AR

En la **Figura 4.12** se puede observar la kurtosis de una señal de ejemplo filtrada en función de los órdenes del modelo AR, en donde el máximo se encuentra en el orden 7 con una Kurtosis de 4.018 . En la **Figura 4.13** se puede observar el contenido frecuencial normalizado por la frecuencia de giro del portador de la señal filtrada, en donde se puede apreciar, en comparación respecto a la **Figura 4.10**, la disminución de algunas componentes a determinadas bandas de frecuencias asociadas a órdenes bajos, mientras que las componentes asociadas a las frecuencias de engrane, cercana al orden 50, se mantiene.

En el **Apéndice C** se muestran los códigos utilizados para la aplicación del *filtro AR*, en el cual también se incluye el uso de la envolvente para normalizar el eje de las frecuencias para cada conjunto de datos. En el **Apéndice G** se puede observar el *espectro de Fourier de los datos con el filtro AR* para cada caso de modo de falla, además de una comparación entre los espectros de un mismo modo de falla pero con distintos niveles de severidad.

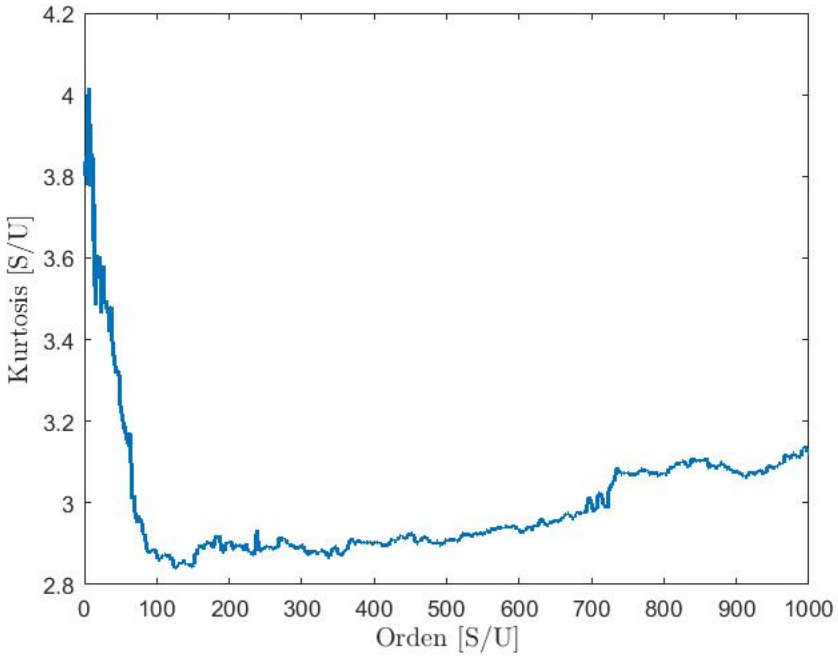

Figura 4.12: Kurtosis en función de los órdenes del modelo AR.

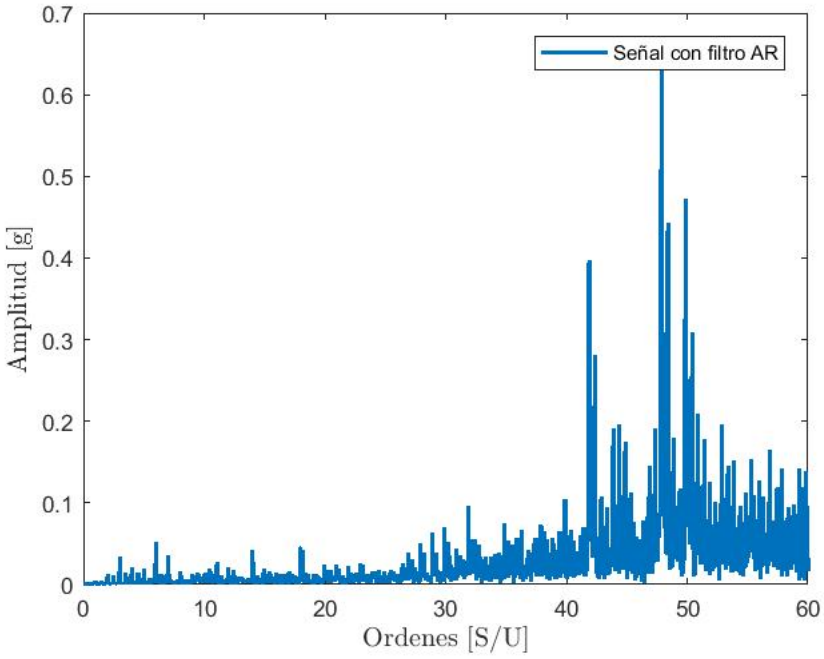

Figura 4.13: Espectro de la señal con filtro AR.

4.5.4. Aplicación del filtro MED

Para aplicar este filtro se utiliza la función *MED2D* en MATLAB, la cual fue creada por *Geoff McDonald* en [8], el cual realiza el procedimiento iterativo descrito anteriormente. En el **Apéndice D** se muestran los códigos utilizados para la aplicación del *filtro MED*, mientras que en el **Apéndice H** se pueden observar resultados de las señales una vez aplicado el *filtro MED* para cada uno de los casos de modo de falla, además de una comparación entre los espectros de un mismo modo de falla pero con distintos niveles de severidad.

4.5.5. Aplicación del filtro ARMED

Los códigos utilizados para la aplicación del *filtro ARMED* son los mismos que los presentados en las dos secciones anteriores. En el **Apéndice E** se muestran los códigos utilizados para la aplicación del *filtro ARMED*, mientras que en el **Apéndice I** se pueden observar resultados de las señales una vez aplicado el *filtro MED* para cada uno de los casos de modo de falla, además de una comparación entre los espectros de un mismo modo de falla pero con distintos niveles de severidad.

4.6. Análisis de los resultados

En esta sección se realizará el análisis de los resultados obtenidos a través de los procesamientos de datos descritos en las secciones anteriores y que se muestran en los Apéndices de este trabajo. El objetivo principal del procesamiento de datos es hacer notar más claramente las fallas en los equipos en donde hay presente algún problema, de tal manera que la *Red Neuronal* pueda identificar más fácilmente los distintos casos.

Para estudiar los resultados se utilizarán los gráficos de los Apéndices correspondientes al procesamiento, en donde se compara cada modo de falla con el caso sin falla y todos los grados de severidad para una misma falla, además, se compararán los resultados de distintos modos de falla para analizar la capacidad de realizar la clasificación de los datos.

En la **Tabla 4.5** se pueden encontrar las frecuencias de falla calculadas según las ecuaciones del **Capítulo 2** en función de órdenes basados en la velocidad de giro del Carrier.

Tabla 4.5: Frecuencias características del reductor.

Características	Magnitud	Unidad
Frecuencia de giro del sol	6	[Orden]
Frecuencia de giro del carrier	1	[Orden]
Frecuencia de engrane	50	[Orden]
Frecuencia de falla local en el sol	15	[Orden]
Frecuencia de falla local del anillo	3	[Orden]
Frecuencia de falla local del planeta	2.5	[Orden]

4.6.1. Análisis de la Transformada de Fourier

Análisis caso sin falla

Para este caso, cuya referencia en el **Apéndice F** es la **Figura F.1**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.6**.

Tabla 4.6: Frecuencias de mayor amplitud - TDF - SIN FALLA

Número	Magnitud	Unidad
1	56	[Orden]
2	51	[Orden]
3	6	[Orden]

La componente de mayor amplitud se encuentra cercana a la frecuencia de engrane a los 51 [Ordenes] que corresponde al máximo en todas las mediciones. Otra componente importante se encuentra a la frecuencia de giro del sol a los 6 [Ordenes], y otra componente a los 56 [Ordenes].

Análisis caso fallas de planetas

Para este caso, cuyas referencias en el **Apéndice F** son las **Figuras F.2, F.3, F.4 y F.11**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.7**.

Tabla 4.7: Frecuencias de mayor amplitud - TDF - FALLA PLANETAS

Número	Magnitud	Unidad
1	54	[Orden]
2	51	[Orden]
3	48	[Orden]
4	6	[Orden]

Respecto al caso sin falla se observa un aumento en la amplitud de la componente a 6 [Ordenes], llegando incluso a magnitudes de 0.7 en la transformada normalizada, además, se puede observar que la componente de 51 [Ordenes] tiene bandas laterales a 3 [Ordenes], lo cual es igual a la frecuencia de falla en el anillo.

Buscando específicamente por la frecuencia de falla del planeta, ubicada en los 2.5 [Ordenes], se puede encontrar un pequeño aumento en los casos de falla, tal como se muestra en la **Figura 4.14**.

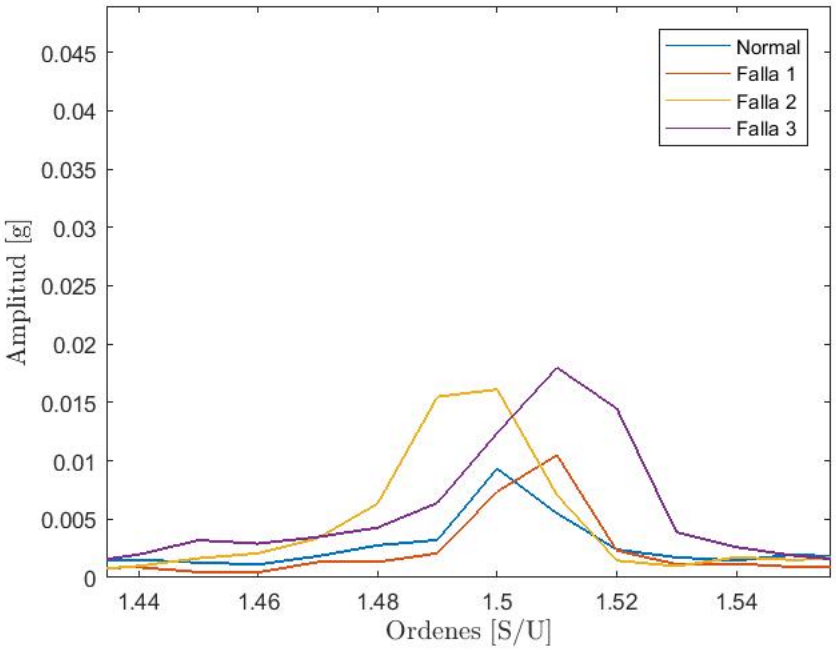

Figura 4.14: Análisis de la frecuencia de falla local en planetas.

Análisis caso fallas de aro

Para este caso, cuyas referencias en el **Apéndice F** son las **Figuras F.5, F.6, F.7 y F.12**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.8**.

Tabla 4.8: Frecuencias de mayor amplitud - TDF - FALLA ARO EXTERIOR

Número	Magnitud	Unidad
1	52	[Orden]
2	51	[Orden]
3	50	[Orden]
4	6	[Orden]

Respecto al caso sin falla se observa un aumento en la amplitud de la componente a 6 [Ordenes], llegando a superar la componente a 51 [Ordenes] en el caso de mayor severidad de fallas, además, esta componente está modulada a la frecuencia del portador.

Buscando específicamente por la frecuencia de falla del aro, ubicada en los 3 [Ordenes], no se observan resultados consistentes, pues tal como se observa en la **Figura 4.15** la componente del caso sin falla es más grande que la del caso con falla nivel 3.

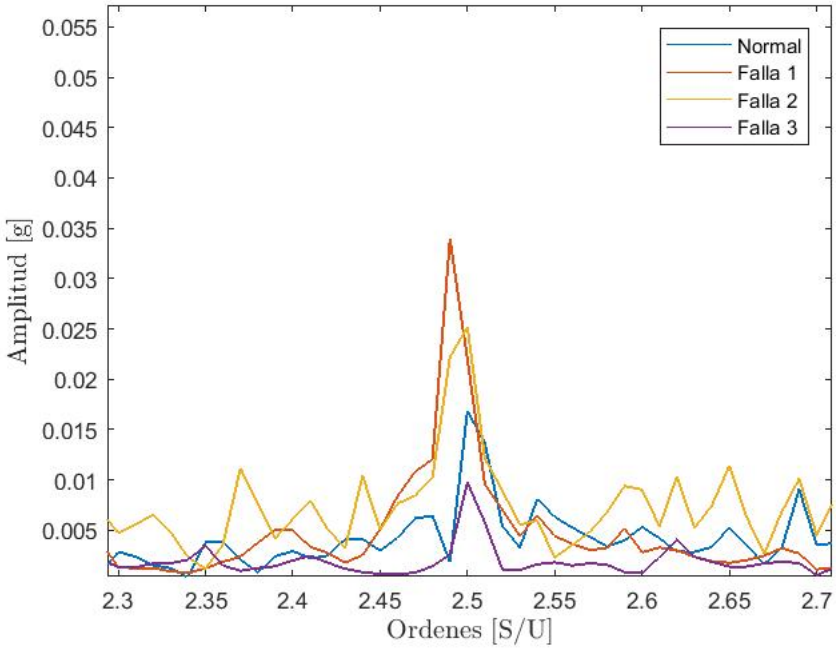

Figura 4.15: Análisis de la frecuencia de falla local en aros.

Análisis caso fallas de sol

Para este caso, cuyas referencias en el **Apéndice F** son las **Figuras F.8, F.9, F.10 y F.13**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.9**.

Tabla 4.9: Frecuencias de mayor amplitud - TDF - FALLA SOL

Número	Magnitud	Unidad
1	58	[Orden]
2	54	[Orden]
3	51	[Orden]
4	6	[Orden]

Respecto al caso sin falla se observa un aumento en la amplitud de la componente a 6 [Ordenes], llegando a superar la componente a 51 [Ordenes] en el caso de mayor severidad de fallas, además, hay otras componentes importantes a 54 [Ordenes] y 58 [Ordenes].

Buscando específicamente por la frecuencia de falla del sol, ubicada en los 15 [Ordenes], se puede observa un aumento en la componente asociada a este modo de falla con el aumento de la severidad de la misma, tal como se puede observar en la **Figura 4.16**.

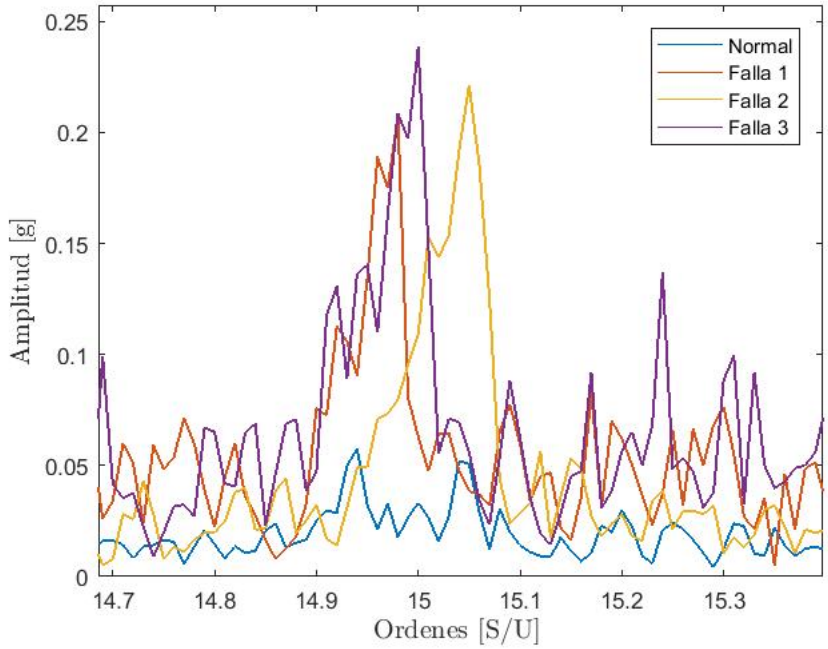

Figura 4.16: Análisis de la frecuencia de falla local en soles.

Comentarios generales [TDF]

De manera general se encontró que al aumentar la severidad de la falla la componente a 6 [Ordenes] aumentó hasta volverse la máxima en el espectro normalizado, sin embargo, esta componente no está asociada a ninguna frecuencia de falla en el equipo, sino que a la frecuencia de giro del sol, por lo que no es ningún indicador directo de falla en el equipo. Otra componente importante que se muestra en todos los espectros es a 51 [Ordenes], a 1 orden de la frecuencia de engrane. Respecto a las frecuencias asociadas a fallas en el equipo, en las comparaciones se encontró que que en general en los casos de falla la magnitud de las componentes aumentaron, lo cual es un buen indicador de que podrían ser utilizados como referencia para la identificación y clasificación de fallas, sin embargo, no son totalmente consistentes, tal como se analizó en el caso de fallas en el aro, en donde las componentes del caso con falla son de menor magnitud de las componentes del caso sin falla.

4.6.2. Análisis de la Envolvente

Análisis caso sin falla

Para este caso, cuya referencia en el **Apéndice J** es la **Figura J.1**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.10**.

Tabla 4.10: Frecuencias de mayor amplitud - ENV - SIN FALLA

Número	Magnitud	Unidad
1	5	[Orden]
2	2.5	[Orden]
3	2	[Orden]
4	1	[Orden]

La componente de mayor amplitud se encuentra cercana a la frecuencia de engrane en los 1, 2 y 2.5 [Ordenes], además, se puede observar una gran cantidad de componentes moduladas a la mitad de la frecuencia de giro del portador, a 0.5 [Ordenes]. Al utilizar la envolvente las componentes de su espectro comienzan a disminuir en amplitud a partir de los 10 [Ordenes], por lo que para el análisis no se consideran mayores ordenes.

Análisis caso fallas de planetas

Para este caso, cuyas referencias son las **Figuras J.2, J.3, J.4 y J.11**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.11**.

Tabla 4.11: Frecuencias de mayor amplitud - ENV - FALLA PLANETAS

Número	Magnitud	Unidad
1	2	[Orden]
2	1.5	[Orden]
3	1	[Orden]
4	0.5	[Orden]

De manera general se puede observar un aumento relativo en las componentes de orden 1 en los casos con falla, así como una disminución en la amplitud de componentes a los 2 y 10 [Ordenes]. Se observan resultados no consistentes asociados a los 6 [Ordenes], en donde para una mayor amplitud en función del grado de severidad de la falla.

Respecto al caso sin falla se puede observar una aumento en la amplitud en las componentes a 0.5 y 1.5 [Ordenes] en los casos con falla, tal como se puede observar en la **Figura 4.17**

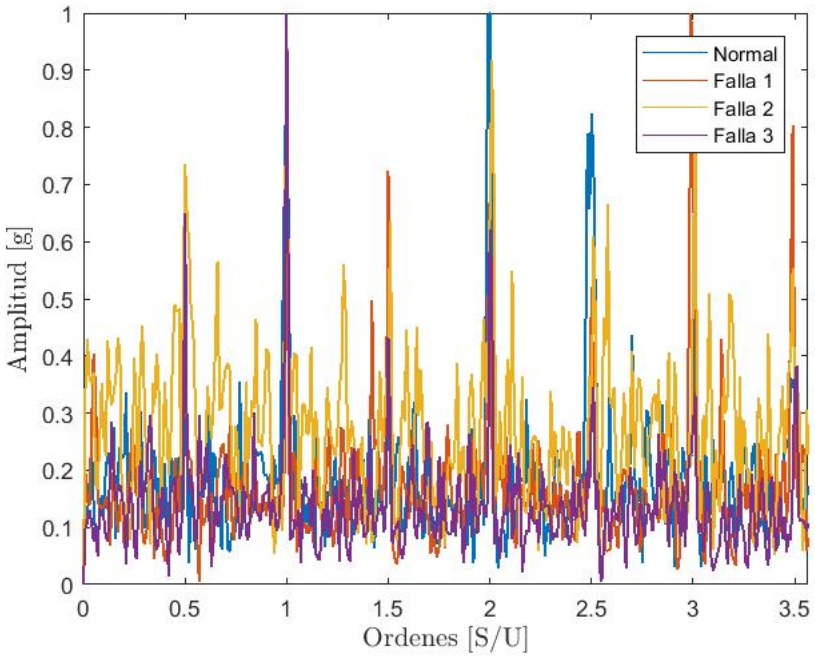

Figura 4.17: Análisis de la frecuencia de falla local en planetas.

Análisis caso fallas de aro

Para este caso, cuyas referencias en el **Apéndice J** son las **Figuras J.5, J.6, J.7 y J.12**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.12**.

Tabla 4.12: Frecuencias de mayor amplitud - ENV - FALLA ARO EXTERIOR

Número	Magnitud	Unidad
1	5	[Orden]
2	3	[Orden]
3	2	[Orden]
4	1	[Orden]

De manera general se puede observar un aumento en las componentes de 1, 2 y 3 [Ordenes] a medida que aumenta en grado de severidad en las fallas, y en algunos casos un aumento en la componente a 5 [Ordenes] en los casos en donde la falla es más grave.

Respecto al caso sin falla se puede observar una disminución en la magnitud de la componente de a 2.5 [Ordenes] y un aumento consistente en la magnitud de la componente a 3 [Ordenes] a medida que aumenta el grado de severidad de la falla, tal como se puede observar en la **Figura 4.18**

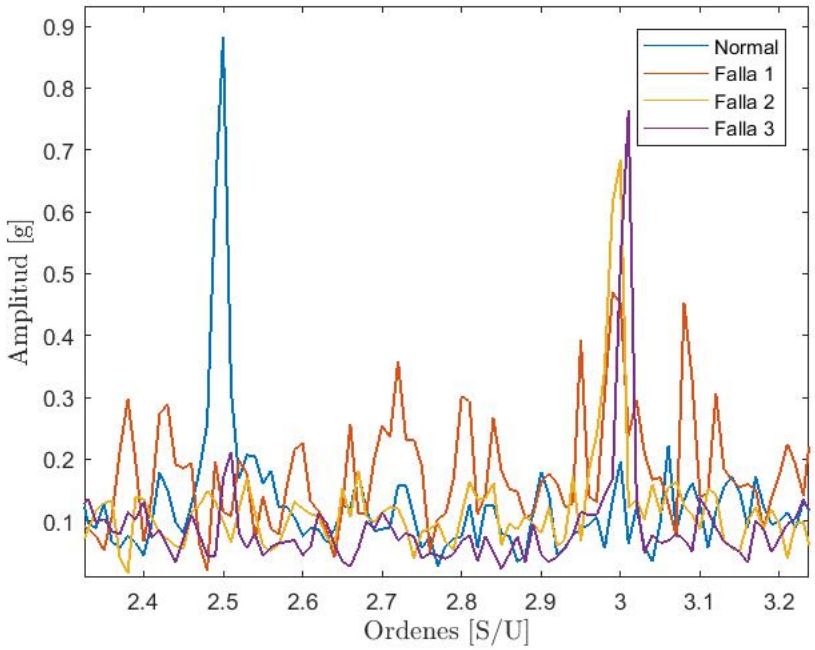

Figura 4.18: Análisis de la frecuencia de falla local en aros.

Análisis caso fallas de sol

Para este caso, cuyas referencias en el **Apéndice J** son las **Figuras J.8, J.9, J.10 y J.13**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.13**.

Tabla 4.13: Frecuencias de mayor amplitud - ENV - FALLA SOL

Número	Magnitud	Unidad
1	6	[Orden]
2	2	[Orden]
3	1.5	[Orden]
4	1	[Orden]

De manera general se puede observar un aumento en las componentes a 6, 2 y 1 [Ordenes] a medida que aumenta en grado de severidad en las fallas.

Respecto al caso sin falla se puede observar una disminución en las componentes a 2 y 2.5 [Ordenes], tal como se puede observar en la **Figura 4.19**

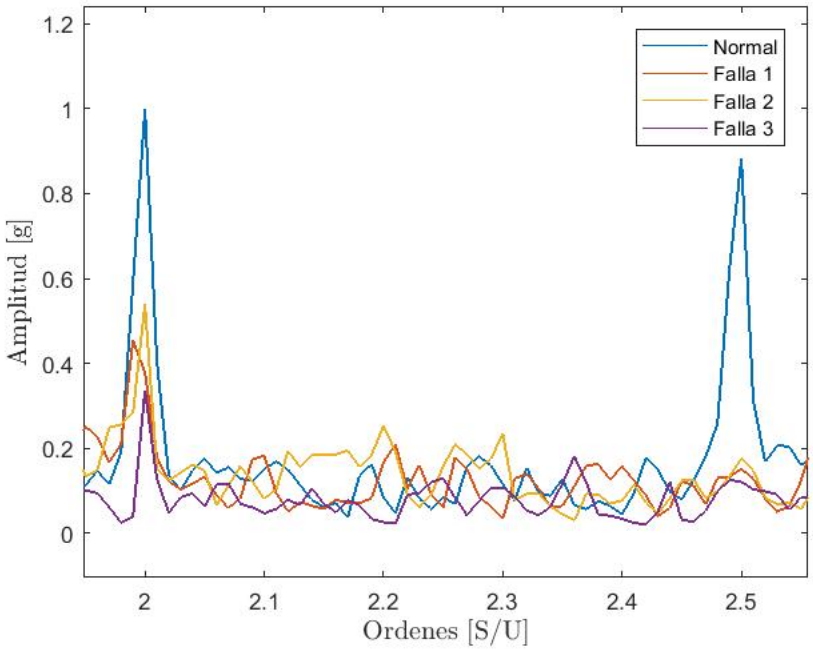

Figura 4.19: Análisis de la frecuencia de falla local en soles.

Comentarios generales [ENV]

De manera general se encontró que al aumentar la severidad de la falla la componente a 2 [Ordenes] disminuyó de manera consistente en todos los casos, y las componentes a 2.5 [Ordenes] también disminuyeron, sin embargo, no de manera tan consistente. A pesar de que hay algunos indicadores de que hay falla, no es posible identificar un aumento o disminución en determinados ordenes y asociarlos a alguna falla en particular, por lo que un trabajo de clasificación podría ser complicado.

4.6.3. Análisis con el Filtro Autoregresivo

Análisis caso sin falla

Para este caso, cuya referencia en el **Apéndice G** es la **Figura G.1**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.14**.

Tabla 4.14: Frecuencias de mayor amplitud - AR - SIN FALLA

Número	Magnitud	Unidad
1	56	[Orden]
2	53	[Orden]
3	51	[Orden]
4	49	[Orden]
5	6	[Orden]

La componente de mayor amplitud se encuentra cercana a la frecuencia de engrane, a 51 [Ordenes], que corresponde al máximo en todas las mediciones, la cual está modulada a 2 [Ordenes], la cual no está asociada a ninguna frecuencia de falla en particular. Además, existen otras componentes importa a los 56 y 6 [Ordenes], este último estando relacionado con la frecuencia de giro del sol.

Análisis caso fallas de planetas

Para este caso, cuyas referencias en el **Apéndice G** son las **Figuras G.2, G.3, G.4 y G.11**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.15**.

Tabla 4.15: Frecuencias de mayor amplitud - AR - FALLA PLANETAS

Número	Magnitud	Unidad
1	54	[Orden]
2	51	[Orden]
3	48	[Orden]
4	6	[Orden]

Respecto al caso sin falla se observa un aumento en la amplitud de la componente a 6 [Ordenes], llegando incluso a magnitudes de 0.8 en la transformada normalizada, además, se puede observar que la componente a 51 [Ordenes] tiene bandas laterales a 3 ordenes, lo cual es igual a la frecuencia de falla en el anillo. Además, las diferencias con el caso sin falla es que las bandas laterales están a 3 [Ordenes] y no a 2 [Ordenes].

Buscando específicamente por la frecuencia de falla del planeta, ubicada en los 2.5 [Ordenes], se puede encontrar un pequeño aumento en los casos de falla, tal como se muestra en la **Figura 4.20**, sin embargo, los resultados no son consistentes pues prácticamente no existe diferencia entre algunos grados de severidad y el caso sin falla.

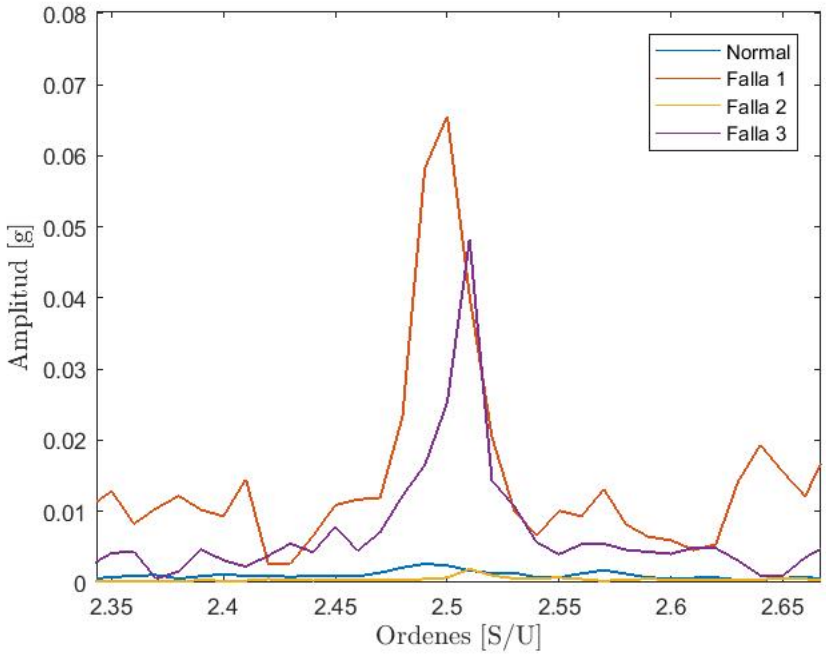

Figura 4.20: Análisis de la frecuencia de falla local en planetas.

Análisis caso fallas de aro

Para este caso, cuyas referencias en el **Apéndice G** son las **Figuras G.5, G.6, G.7 y G.12**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.16**.

Tabla 4.16: Frecuencias de mayor amplitud - AR - FALLA ARO EXTERIOR

Número	Magnitud	Unidad
1	56	[Orden]
2	51	[Orden]
3	6	[Orden]

Respecto al caso sin falla se observa un aumento en la amplitud de la componente a 6 [Ordenes], llegando a ser el máximo, superando en magnitud a la componente de 51 [Ordenes], la cual es normalmente la más grande en todos los casos estudiados.

Buscando específicamente por la frecuencia de falla del aro, ubicada en los 3 [Ordenes], se observa un aumento en la magnitud de estas componentes en los casos de falla, de tal manera que a mayor grado de severidad mayor es su magnitud, lo cual es bastante consistente, tal como se puede observar en la **Figura 4.21**

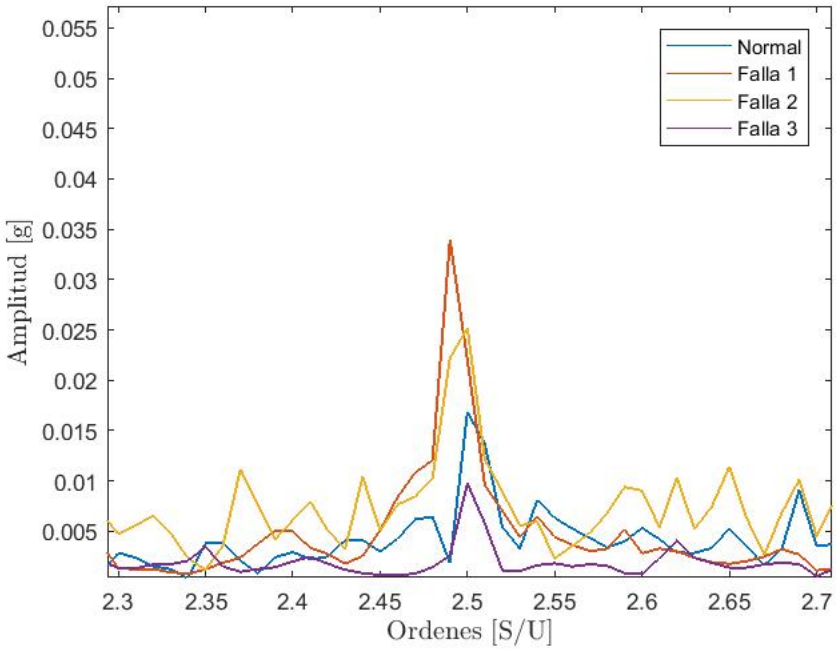

Figura 4.21: Análisis de la frecuencia de falla local en aros.

Análisis caso fallas de sol

Para este caso, cuyas referencias en el **Apéndice G** son las **Figuras G.8, G.9, G.10 y G.13**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.17**.

Tabla 4.17: Frecuencias de mayor amplitud - AR - FALLA SOL

Número	Magnitud	Unidad
1	51	[Orden]
2	36	[Orden]
3	6	[Orden]

Respecto al caso sin falla se observa un aumento en la amplitud de la componente a 6 [Ordenes], llegando a superar la componente a 51 [Ordenes] en el caso de mayor severidad de fallas, además, aparece una componente importante a los 36 [Ordenes], separado 15 [Ordenes] de la frecuencia de engrane, lo cual podría ser una banda lateral a la frecuencia de falla del sol.

Buscando específicamente por la frecuencia de falla del sol, ubicada en los 15 [Ordenes], se puede observa un aumento en la componente asociada a este modo de falla con el aumento de la severidad de la misma, tal como se puede observar en la **Figura 4.22**.

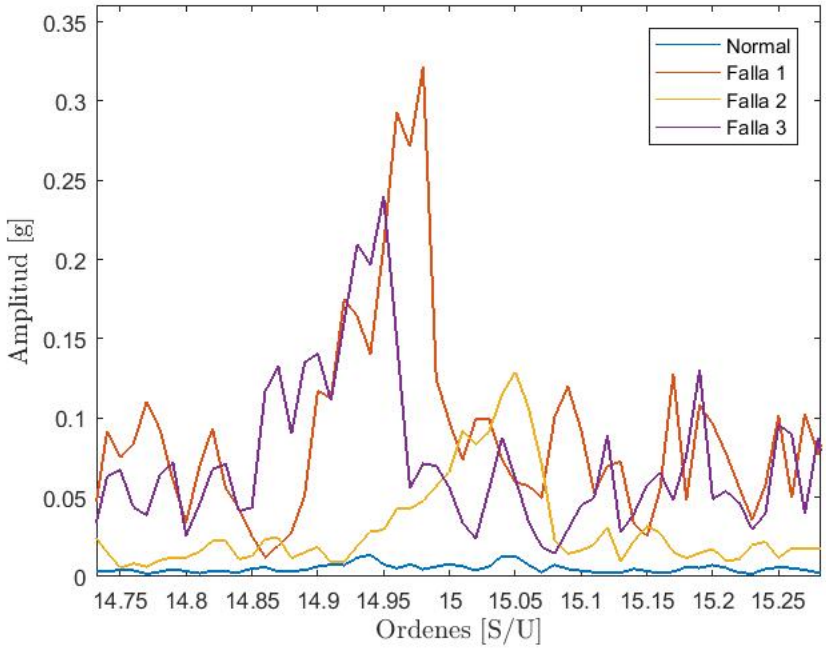

Figura 4.22: Análisis de la frecuencia de falla local en soles.

Comentarios generales [AR]

De manera general se encontró que al aumentar la severidad de la falla la componente a 6 [Ordenes] aumentó hasta volverse la máxima en el espectro normalizado, sin embargo, esta componente no está asociada a ninguna frecuencia de falla en el equipo, sino que a la frecuencia de giro del sol, por lo que no es ningún indicador directo de falla en el equipo. Otra componente importante que se muestra en todos los espectros es a 51 [Ordenes], a 1 [Ordenes] de la frecuencia de engrane. Respecto a las frecuencias asociadas a fallas en el equipo, en las comparaciones se encontró que que en general en los casos de falla la magnitud de las componentes aumentaron, lo cual es un buen indicador de que podrían ser utilizados como referencia para la identificación y clasificación de fallas, sin embargo, no son totalmente consistentes, tal como se analizó en el caso de fallas en los planetas.

4.6.4. Análisis con el Filtro de Deconvolución de Mínima Entropía

Análisis caso sin falla

Para este caso, cuya referencia en el **Apéndice H** es la **Figura H.1**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.18**.

Tabla 4.18: Frecuencias de mayor amplitud - MED - SIN FALLA

Número	Magnitud	Unidad
1	56	[Orden]
2	51	[Orden]
3	31	[Orden]
4	21	[Orden]
5	6	[Orden]

La componente de mayor amplitud se encuentra cercana a la frecuencia de engrane, a 51 [Ordenes], que corresponde al máximo en todas las mediciones, la cual está modulada a 5 [Ordenes], lo cual no corresponde a ninguna frecuencia de falla en particular, sin embargo, lo más destacable de los espectros sin falla es la amplitud de las componentes, las cuales rara vez superan 0.3 o 0.4 en magnitud.

Análisis caso fallas de planetas

Para este caso, cuyas referencias en el **Apéndice H** son las **Figuras H.2, H.3, H.4 y H.11**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.19**.

Tabla 4.19: Frecuencias de mayor amplitud - MED - FALLA PLANETAS

Número	Magnitud	Unidad
1	51	[Orden]
2	21	[Orden]
3	6	[Orden]

Respecto al caso sin falla en primera instancia se observa que las componentes indicadas en la **Tabla 4.19** poseen siempre amplitud 1 en cualquiera de los casos de falla, sin embargo, no se observa consistencia en los máximos para distintas mediciones, pudiendo ser cualquiera de los 3.

Buscando específicamente por la frecuencia de falla del planeta, ubicada en los 2.5 [Ordenes], se puede encontrar un pequeño aumento en los casos de falla, tal como se muestra en la **Figura 4.23**, sin embargo, los resultados no son consistentes pues prácticamente no existe diferencia entre algunos grados de severidad y el caso sin falla.

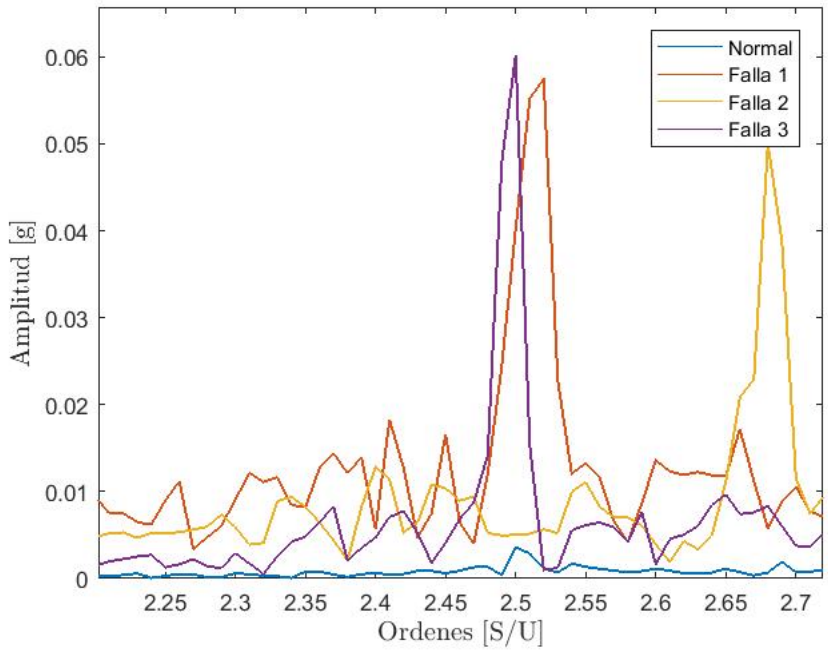

Figura 4.23: Análisis de la frecuencia de falla local en planetas.

Análisis caso fallas de aro

Para este caso, cuyas referencias en el **Apéndice H** son las **Figuras H.5, H.6, H.7 y H.12**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.20**.

Tabla 4.20: Frecuencias de mayor amplitud - MED - FALLA ARO EXTERIOR

Número	Magnitud	Unidad
1	58	[Orden]
2	42	[Orden]
3	21	[Orden]
4	15	[Orden]
5	6	[Orden]

Al igual que en caso con los planetas, las magnitudes de los casos con falla a los 51 y 6 [Ordenes] son siempre los máximos, por lo que se diferencian bastante del caso sin falla. Es importante indicar que a medida que la severidad de la falla aumenta el orden 6 tiende a tener mayor amplitud que la componente a 51 [Ordenes], sin embargo, esto no es consistente en todos los casos.

Buscando específicamente por la frecuencia de falla del aro, ubicada en los 3 [Ordenes], se observa un aumento en la magnitud de estas componentes en los casos de falla, sin embargo, en general no existe consistencia entre la severidad de la falla y la amplitud de esta componente, lo cual es bastante consistente, tal como se puede observar en la **Figura 4.24**

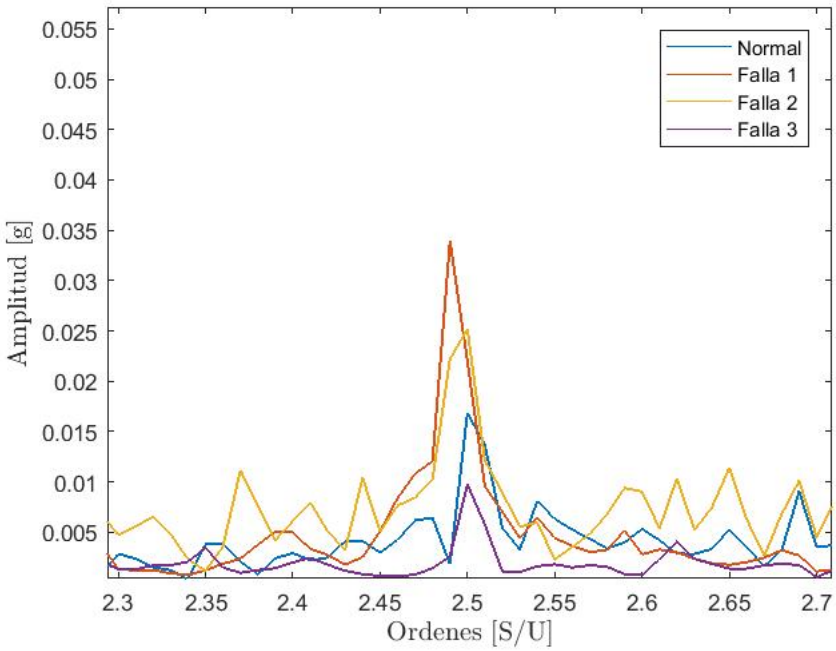

Figura 4.24: Análisis de la frecuencia de falla local en aros.

Análisis caso fallas de sol

Para este caso, cuyas referencias en el **Apéndice H** son las **Figuras H.8, H.9, H.10 y H.13**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.21**.

Tabla 4.21: Frecuencias de mayor amplitud - MED - FALLA SOL

Número	Magnitud	Unidad
1	51	[Orden]
2	21	[Orden]
3	6	[Orden]

Al igual que en caso con los planetas, las magnitudes de los casos con falla a los ordenes 51 y 6 son siempre los máximos, por lo que se diferencian bastante del caso sin falla. Es importante indicar que a medida que la severidad de la falla aumenta a los 6 [Ordenes] tiende a tener mayor amplitud que la componente de 51 [Ordenes], sin embargo, esto no es consistente en todos los casos.

Buscando específicamente por la frecuencia de falla del sol, ubicada en los 15 [Ordenes], se puede observa un aumento en la componente asociada a este modo de falla con el aumento de la severidad de la misma, tal como se puede observar en la **Figura 4.25**, sin embargo, no existe consistencia entre la severidad de la falla y la amplitud de la componente en todos los casos.

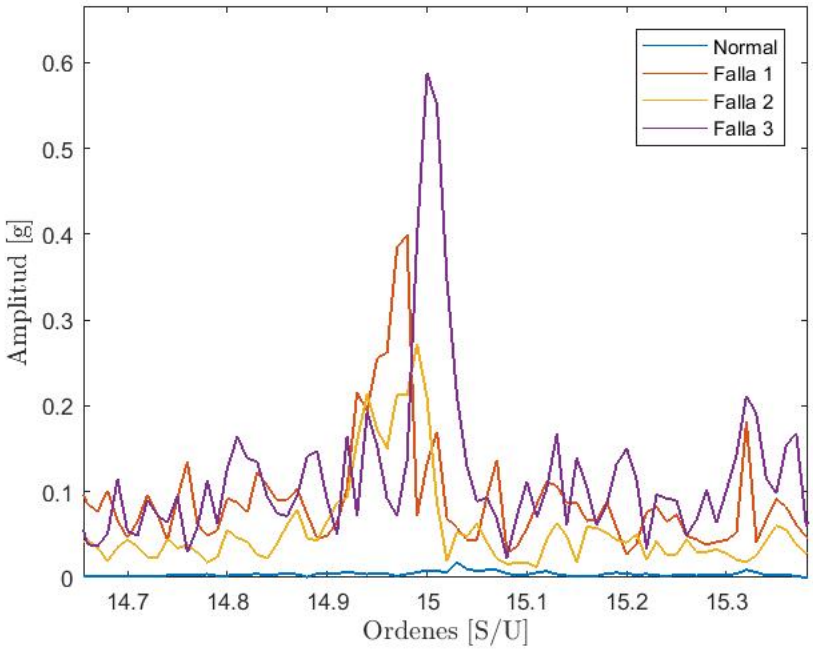

Figura 4.25: Análisis de la frecuencia de falla local en soles.

Comentarios generales [MED]

De manera general se encontró que en los casos con falla las componentes a 51, 21 y 6 [Ordenes] aumentaron en amplitud, siendo siempre los máximos en la transformada normalizada, además, el hecho de que para los casos sin falla la amplitud de todas las componentes hasta 70 [Ordenes] sea en general de amplitud pequeña, menores a 0.5, **indica que este es un excelente método para la identificación de fallas en el equipo**. Además de esto, a pesar de que no existe consistencia entre la amplitud de las componentes asociadas a las frecuencias de falla y el grado de severidad de cada caso, sería posible identificar el tipo de falla a partir de una variación en la amplitud de esas componentes.

4.6.5. Análisis con el Filtro ARMED

Análisis caso sin falla

Para este caso, cuya referencia en el **Apéndice I** es la **Figura I.1**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.22**.

Tabla 4.22: Frecuencias de mayor amplitud - ARMED - SIN FALLA

Número	Magnitud	Unidad
1	56	[Orden]
2	53	[Orden]
3	51	[Orden]
4	6	[Orden]

La componente de mayor amplitud se encuentra cercana a la frecuencia de engrane, a 51 [Ordenes], que corresponde al máximo en todas las mediciones, la cual está modulada a 5 y a 2 [Ordenes], lo cual no corresponde a ninguna frecuencia de falla en particular. La componente a 6 [Ordenes] siempre aparece, pero nunca es de mayor amplitud que la de 51 [Ordenes].

Análisis caso fallas de planetas

Para este caso, cuyas referencias en el **Apéndice I** son las **Figuras I.2, I.3, I.4 y I.11**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.23**.

Tabla 4.23: Frecuencias de mayor amplitud - ARMED - FALLA PLANETAS

Número	Magnitud	Unidad
1	54	[Orden]
2	51	[Orden]
3	48	[Orden]
4	21	[Orden]
5	6	[Orden]

Respecto al caso sin falla se observa que las componentes a 6 y 51 [Ordenes] siempre son las máximas, con amplitud 1, sin embargo, no existe consistencia respecto a cuál de ellas es la más relevante para los distintos grados de severidad para este caso.

Buscando específicamente por la frecuencia de falla del planeta, ubicada en los 2.5 [Ordenes], se puede encontrar un pequeño aumento en los casos de falla, tal como se muestra en la **Figura 4.26**, sin embargo, los resultados no son consistentes pues no hay relación entre el grado de severidad de la falla y la amplitud de la componente a esa frecuencia.

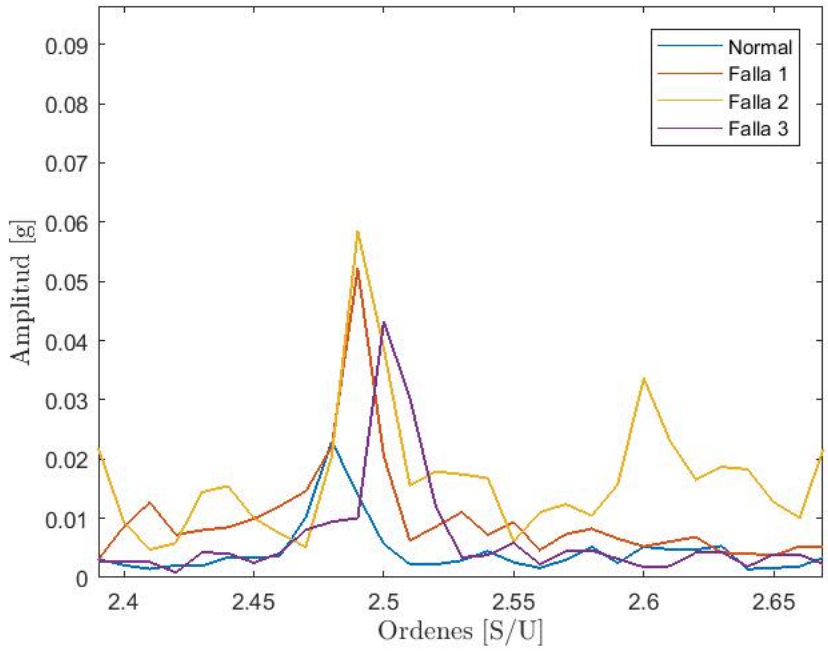

Figura 4.26: Análisis de la frecuencia de falla local en planetas.

Análisis caso fallas de aro

Para este caso, cuyas referencias en el **Apéndice I** son las **Figuras I.5, I.6, I.7 y I.12**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.24**.

Tabla 4.24: Frecuencias de mayor amplitud - ARMED - FALLA ARO EXTERIOR

Número	Magnitud	Unidad
1	51	[Orden]
2	15	[Orden]
3	6	[Orden]

En este caso la componente de mayor amplitud es casi siempre a 6 [Ordenes]. Es importante indicar que a medida que la severidad de la falla aumenta el orden 6 tiende a tener mayor amplitud que la componente a 51 [Ordenes], sin embargo, esto no es consistente en todos los casos.

Buscando específicamente por la frecuencia de falla del aro, ubicada en los 3 [Ordenes], se observa un aumento en la magnitud de estas componentes en los casos de falla, sin embargo, en general no existe consistencia entre la severidad de la falla y la amplitud de esta componente, lo cual es bastante consistente, además, en algunos casos la amplitud de los casos con falla a esta frecuencia es semejante a la del caso sinfalla, tal como se puede observar en la **Figura 4.27**.

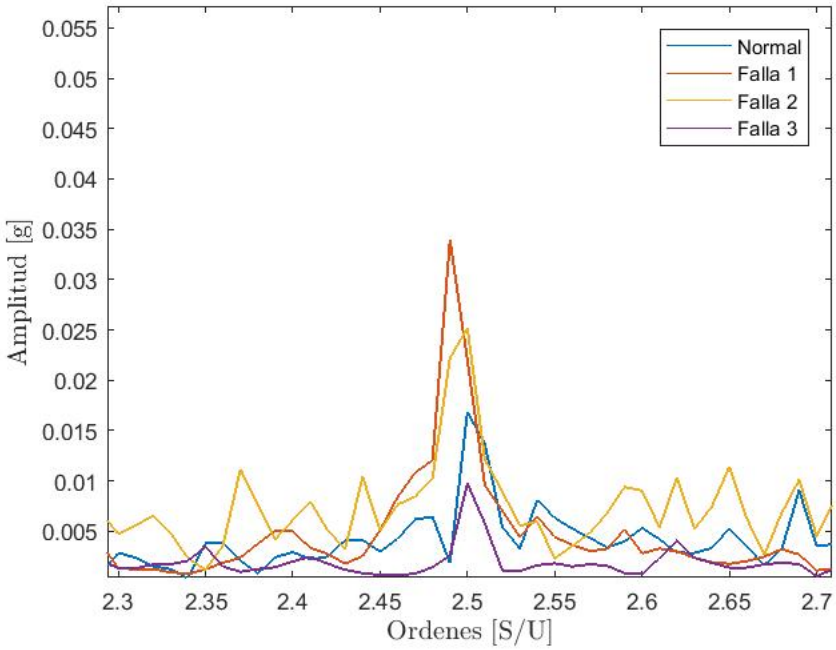

Figura 4.27: Análisis de la frecuencia de falla local en aros.

Análisis caso fallas de sol

Para este caso, cuyas referencias en el **Apéndice I** son las **Figuras I.8, I.9, I.10 y I.13**, las componentes de mayor amplitud se pueden observar en la **Tabla 4.25**.

Tabla 4.25: Frecuencias de mayor amplitud - ARMED - FALLA SOL

Número	Magnitud	Unidad
1	51	[Orden]
2	36	[Orden]
3	6	[Orden]

En este caso la componente de mayor amplitud es casi siempre a 5 [Ordenes]. Es importante indicar que a medida que la severidad de la falla aumenta el orden 6 tiende a tener mayor amplitud que la componente de a 51 [Ordenes], sin embargo, esto no es consistente en todos los casos.

Buscando específicamente por la frecuencia de falla del sol, ubicada en los 15 [Ordenes], se puede observa un aumento en la componente asociada a este modo de falla con el aumento de la severidad de la misma, tal como se puede observar en la **Figura 4.28**, sin embargo, no existe consistencia entre la severidad de la falla y la amplitud de la componente en todos los casos.

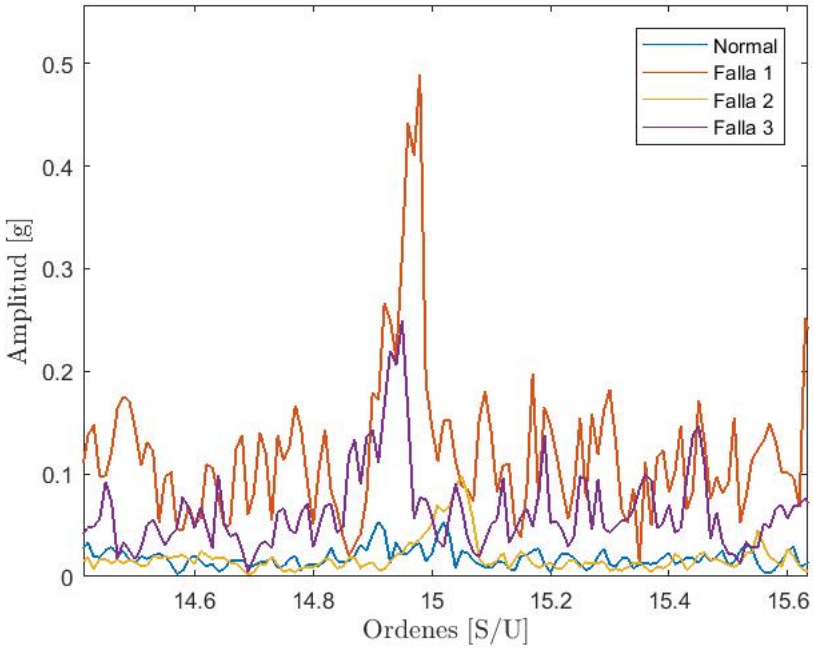

Figura 4.28: Análisis de la frecuencia de falla local en soles.

Comentarios generales [ARMED]

De manera general se encontró que en los casos con falla las componentes a 51 y 6 [Ordenes] siempre fueron los de mayor amplitud, por lo que no se podría realizar ningún tipo de identificación ni clasificación. A diferencia del caso del filtro MED, en donde las amplitudes de las componentes menores a 60 [Ordenes] son en general menores a 0.5 en la amplitud normalizada.

4.7. Análisis de las frecuencias de fallas

Es importante indicar de que en la mayoría de las ocasiones, a pesar de que se pueda hacer una relación entre el grado de severidad y la amplitud de la componente para un determinado modo de falla, en ese mismo conjunto de datos las componentes asociados a otros modos de falla también son visibles, por ejemplo, en las **Figuras 4.29, 4.30 y 4.31**, las cuales corresponden a Espectros de Fourier de datos procesados con el filtro MED para el caso de falla en planetas, se puede observar un acercamiento a cada uno de los ordenes asociados a cada modo de falla estudiado en este capítulo, y se puede observar que todos los grados de severidad poseen componentes más grandes que el caso sin falla para todos los modos de falla, a pesar de que este sea solo el caso de falla en planetas. Dicho esto, y entendiendo que este comportamiento es recurrente y regular en todos los conjuntos de datos, resultaría complejo realizar la clasificación de los tipos de falla según las únicamente basándose en las amplitudes a las frecuencias de falla.

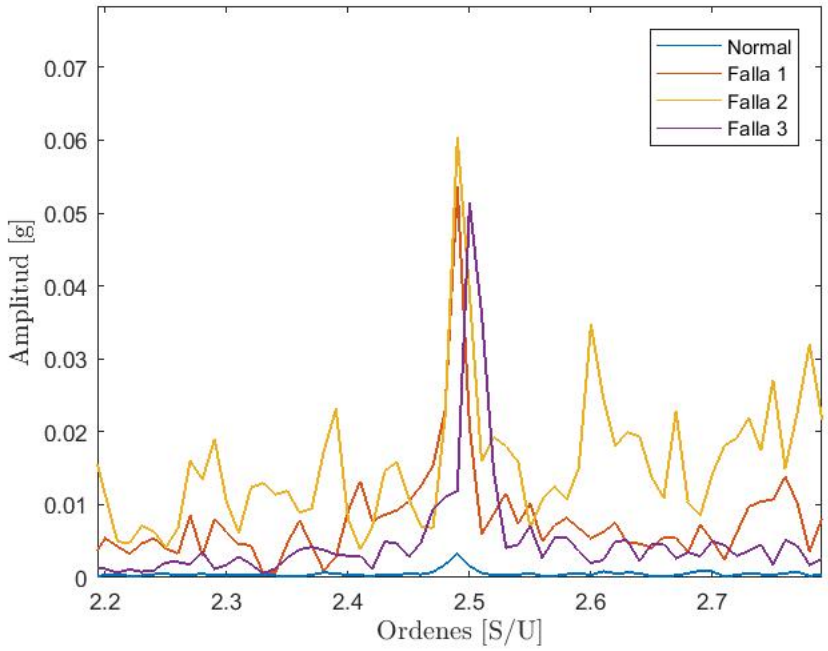

Figura 4.29: Análisis de la frecuencia de falla local en planetas.

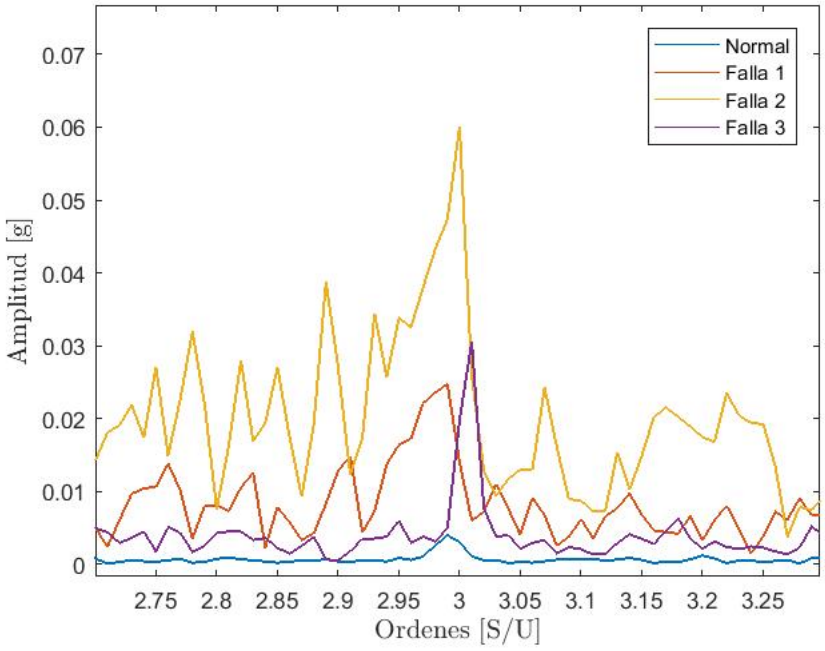

Figura 4.30: Análisis de la frecuencia de falla local en el anillo.

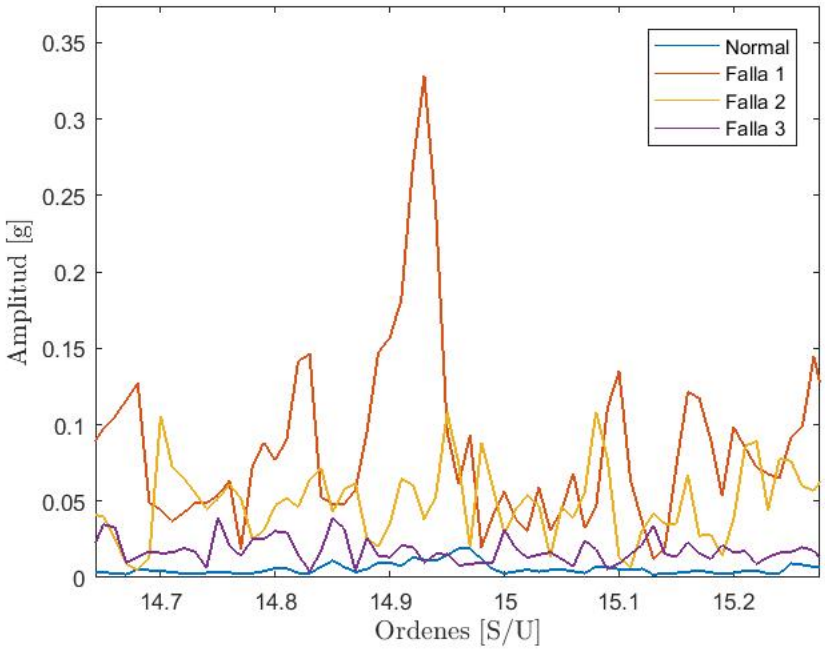

Figura 4.31: Análisis de la frecuencia de falla local en soles.

4.8. Análisis general de los métodos de procesamiento

De manera general en los resultados obtenidos a través de todos los procesamientos se pudo encontrar que las componentes que mayor amplitud tuvieron fueron a 6 y 51 [Ordenes], las cuales están asociadas a la frecuencia de giro del sol y la frecuencia de engrane. En algunos de los casos la componente de mayor amplitud fue a 21 [Ordenes]. También en algunos casos hubo presencia de bandas laterales, a 2 y 3 [Ordenes] de la frecuencia de engrane, sin embargo, ninguno de estos resultados es útil para la clasificación de fallas, pues los resultados no son consistentes y no se podría diferenciar entre los distintos modos de fallas utilizando como referencia solamente las amplitudes de estas componentes. Para lo cual podría resultar útil estas componentes es la identificación de las fallas, pues existe una tendencia a que la componente de orden 6 aumente en amplitud a medida que el grado de severidad de la falla aumenta.

Con ninguno de los procesamientos aplicados alguna de las componentes de mayor amplitud fueron las que están asociadas a falla local en el equipo, indicadas en la **Tabla 4.5**, lo cual complica la clasificación de los modos de falla. Tal como se indicó en el capítulo anterior la clasificación puede ser complicada con los métodos estudiados porque para un determinado modo de falla, las componentes asociadas a las frecuencias de falla de todos los otros modos están presentes, y debido a estas frecuencias cruzadas es difícil diferenciar una falla de otra. Lo que si sería posible con el análisis de estas componentes sería la identificación de fallas, pues en los casos sin falla la amplitud de estas componentes son bastante más pequeñas que en los casos con falla.

De todos los métodos estudiados el filtro de Deconvolución de Mínima Entropia (MED) es el que mejor puede diferenciar entre los casos con y sin falla, debido a que en estos últimos las amplitudes son mucho más pequeñas comparadas con los otros casos, lo cual es sumamente útil. Debido a esto para el siguiente capítulo, en el cual se buscará entrenar un algoritmo de *Redes Neuronales Artificiales* con los datos simulados para identificar falla en los datos reales, se utilizaran los datos con el filtro MED, y se buscará principalmente identificar las fallas.

Capítulo 5

Entrenamiento de Red Neuronal

En este capítulo se describen los parámetros utilizados para la creación de una *Red Neuronal* de tipo *Artificial Neuronal Network (ANN)*, para lo cual se utilizó la librería *Keras*, disponible en el software *Python*. Para entrenar la red se utilizarán datos simulados con el modelo descrito en los capítulos anteriores, para lo cual se tienen que calibrar los parámetros del modelo para que se ajusten lo mejor posible a los datos experimentales, de tal manera que exista coherencia entre los datos con los que se entrena la red y los datos que se pretenden clasificar.

5.1. Parámetros de la simulación

Para poder entrenar apropiadamente la *ANN* se deben seleccionar cuidadosamente los parámetros de las simulaciones, de tal manera que se parezcan a los datos experimentales para los distintos casos. Hay que tener en cuenta que, tal como se indicó en el capítulo del modelo fenomenológico, los parámetros del modelo son limitados y no pueden representar todas las componentes observadas en el capítulo de mediciones experimentales, en particular aquellas que no están relacionadas con alguna frecuencia característica o frecuencia de falla.

Para poder seleccionar los parámetros de las simulaciones se tiene en consideración los resultados de la **sección 4.6.4**, en particular las **Tablas 4.18, 4.19, 4.20 y 4.21**. Se pueden encontrar tres tipos de espectros con características similares, los cuales se describen a continuación:

1. En el caso sin falla los componentes principales se encuentran a los ordenes 6 y 51, con amplitudes que varían entre los 0.05 a los 0.5 en el espectro normalizado. Aparecen otras componentes a otras frecuencias características, pero de menor amplitud.
2. En uno de los casos son falla el componente principal es de orden 51, de amplitud 1 en el espectro normalizado, y el segundo componente de mayor amplitud es de orden 6, pudiendo llegar a amplitudes de hasta 0.8 en el espectro normalizado. Es posible encontrar otras componentes a frecuencias características del equipo.
3. En uno de los casos son falla el componente principal es de orden 6, de amplitud 1 en el espectro normalizado, y el segundo componente de mayor amplitud es de orden

51, pudiendo llegar a amplitudes de hasta 0.8 en el espectro normalizado. Es posible encontrar otras componentes a frecuencias características del equipo.

Dicho lo anterior, se crean simulaciones para poder generar conjuntos de datos con las características descritas en el punteo anterior, para lo cual se tiene en considerar variar los parámetros que se muestran en el **Anexo K, en las Tablas K.1, K.2, K.3 y K.4**,en donde se puede observar los valores con los que se probó y el valor final seleccionado con los que se obtuvieron los mejores resultados. Las variables que se consideraron para la modificación de las simulaciones y poder obtener mejores resultados con la red se indican a continuación:

- Largo de los datos.
- Amplitudes de las componentes en la transformada.
- Variación en las frecuencias de giro.

Se encontró que utilizando el código del caso con falla para planetas para simular los casos sin falla se obtuvieron mejores resultados para la exactitud de la red a la hora de identificar las fallas. En las **Figuras 5.1, 5.2, 5.3 y 5.4** se puede observar ejemplos de los datos simulados para cada uno de los casos descritos anteriormente.

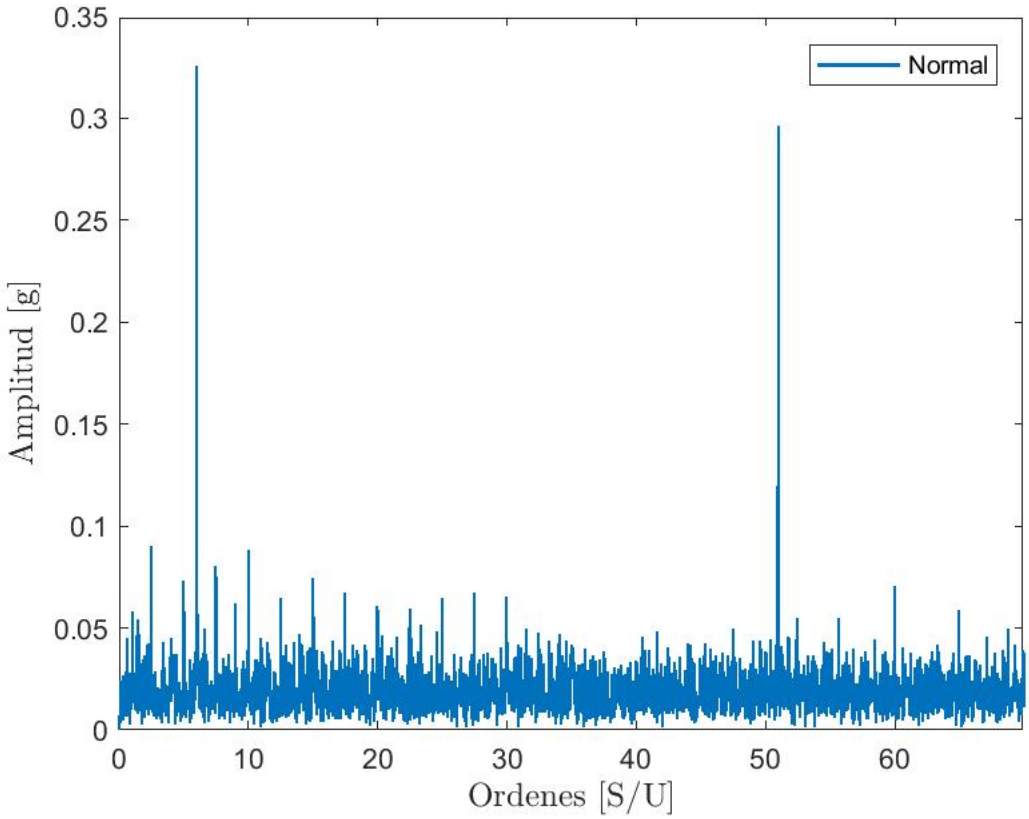

Figura 5.1: TDF de la simulación del caso sin falla.

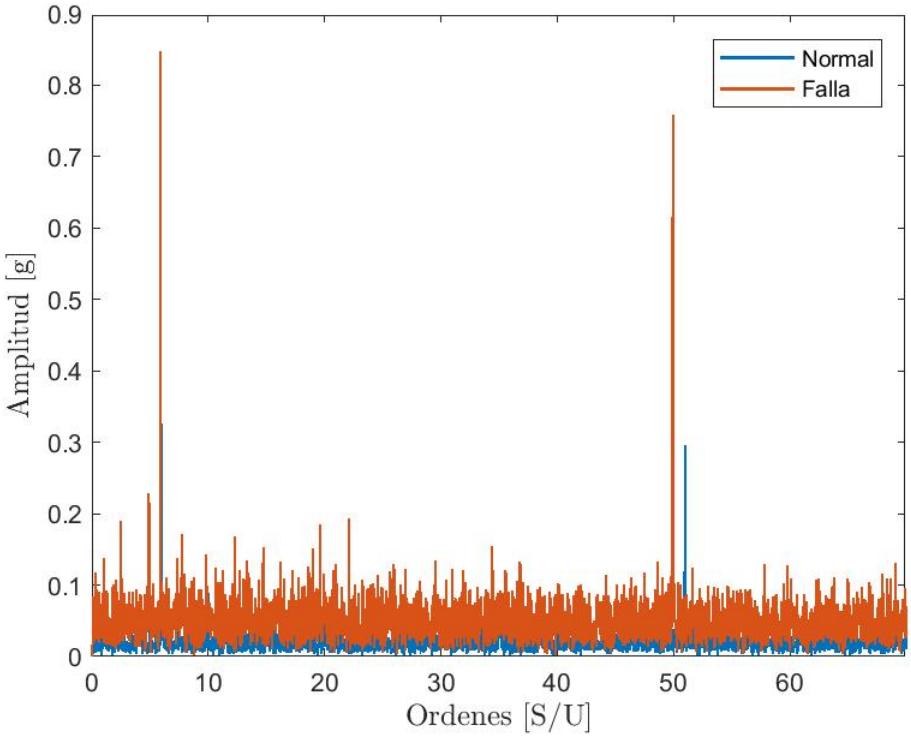

Figura 5.2: TDF de la simulación del caso falla en planeta.

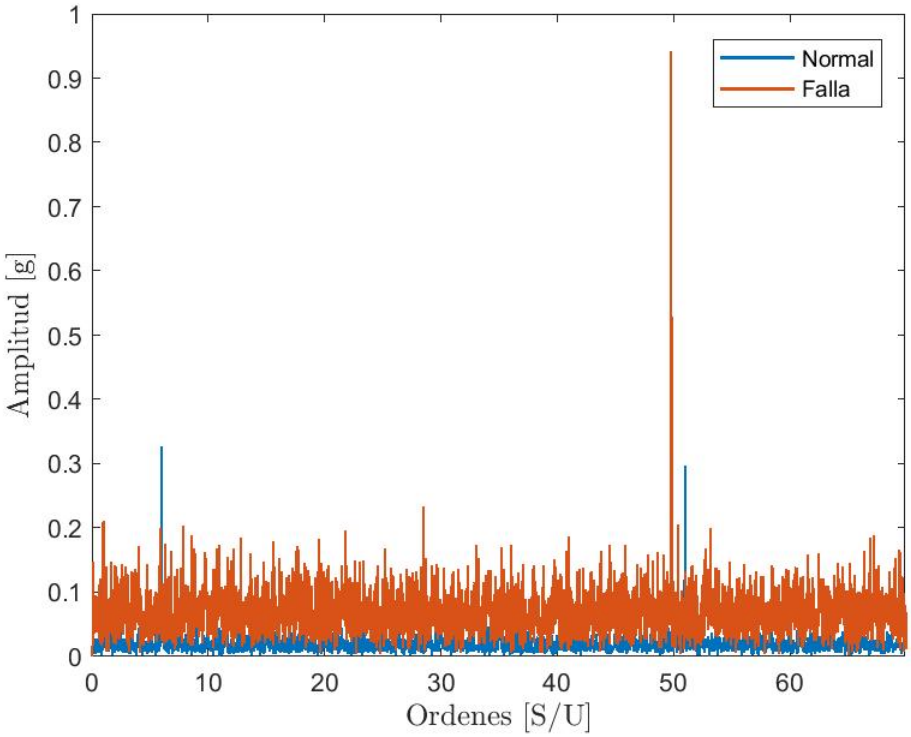

Figura 5.3: TDF de la simulación del caso falla en aro.

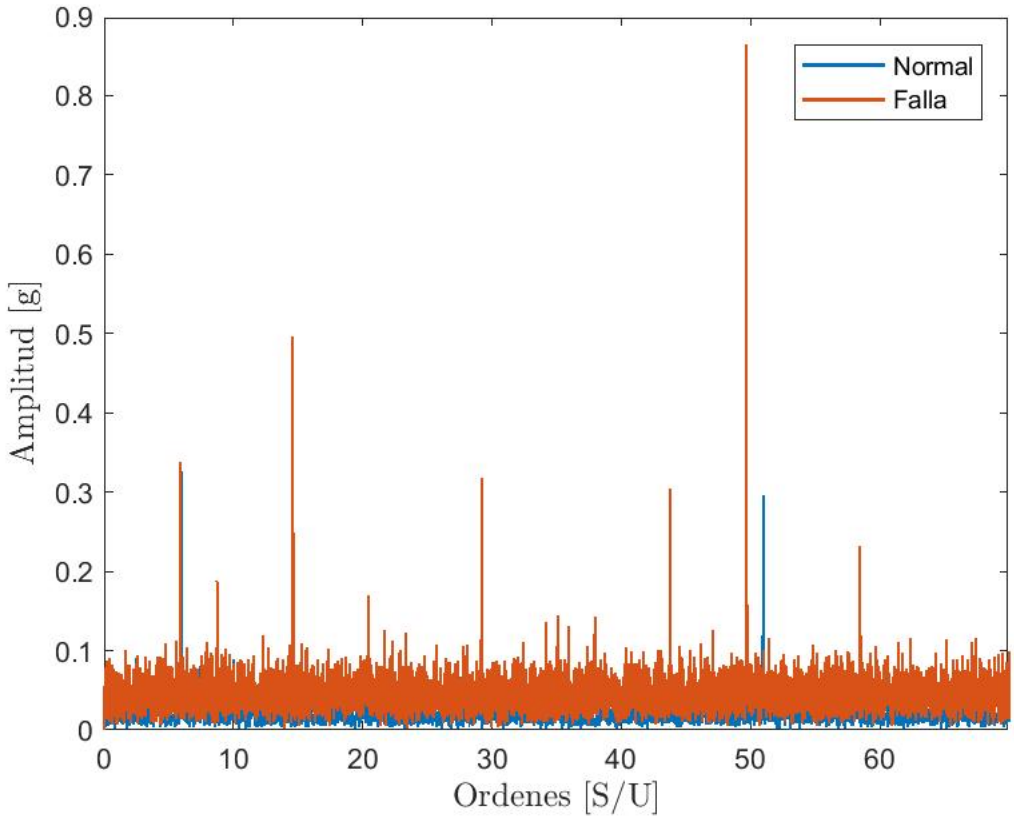

Figura 5.4: TDF de la simulación del caso falla en sol.

Se decide utilizar una *Artificial Neural Network (ANN)* para realizar la clasificación de los datos, principalmente porque se pueden configurar para que requieran de pocos recursos computacionales para realizar los entrenamientos, lo cual es escencial, ya que debido a la gran cantidad de iteraciones que se deben hacer para los distintos casos obtenidos con los parámetros de las simulaciones, el entrenamiento debe demorarse la menor cantidad de tiempo posible.

5.2. Parámetros de la red y resultados

El propósito de la red propuesta es la identificación de fallas, por lo que se utilizará *Binary Crossentropy* como función de pérdida. Los parámetros de la red son indicados en la **Tabla 5.1**, se debe indicar que para su entrenamiento se utilizaron 1800 datos de casos con falla y 1800 datos de casos sin falla. Además de esto se utilizó *Dropout*, el cual consiste en un método que desactiva de manera aleatoria un porcentaje de las neuronas durante el entrenamiento, obligando a la red a ser lo más general posible. En el **Anexo L** se puede observar el código utilizado para entrenar la ANN

Con los parámetros recién indicados se obtuvo una matriz de confusión como la que se muestra en la **Figura 5.5**, además, en la Figura 5.6 se puede observar el gráfico de la función de pérdida en función de la época. La exactitud en las predicciones obtenidas son del 98.9 % para el total de los datos simulados, lo cual es bastante alto y aceptable. De manera general no se tuvo muchas complicaciones para poder hacer llegar la red simulada a obtener exactitudes bastante altas, cercanas al 100 %.

En la **Figura 5.7** se puede observar la matriz de confusión de los datos experimentales, en donde se observa que la exactitud de la predicción para casos sin falla es del 86,6 % y para el caso con falla es del 97,4 %

Tabla 5.1: Parámetros utilizados para el entrenamiento de la ANN

Datos de las capas			
Capa N°	N° de neuronas	Función de activación	Dropout
1	7000	relu	∼
2	4000	relu	0,2
3	500	relu	0,2
4	200	relu	0,2
5	50	relu	0,2
6	1	sigmoid	∼
Datos del entrenamiento			
Parámetro		Valor asignado	
Dimensión de entrada		7000	
Optimizador		adam	
Función de pérdida		Binary Crossentropy	
Batch Size		400	
N° de épocas		80	
Validation Split		20 %	

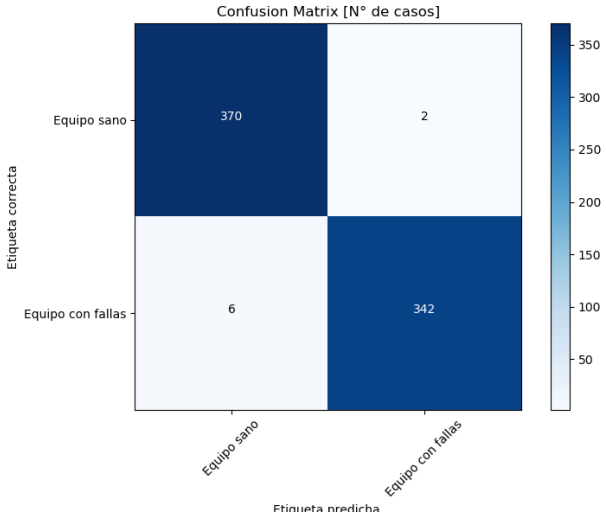

Figura 5.5: Matriz de confusión para el entrenamiento de los datos.

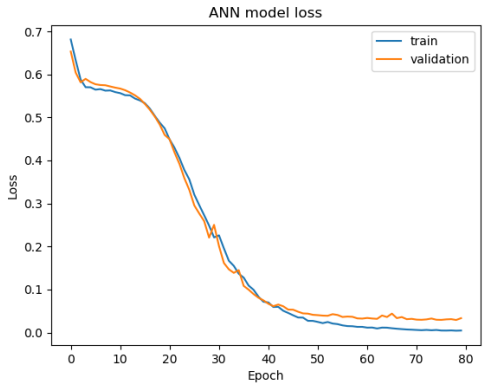

Figura 5.6: Gráfico de la función de pérdida del entrenamiento.

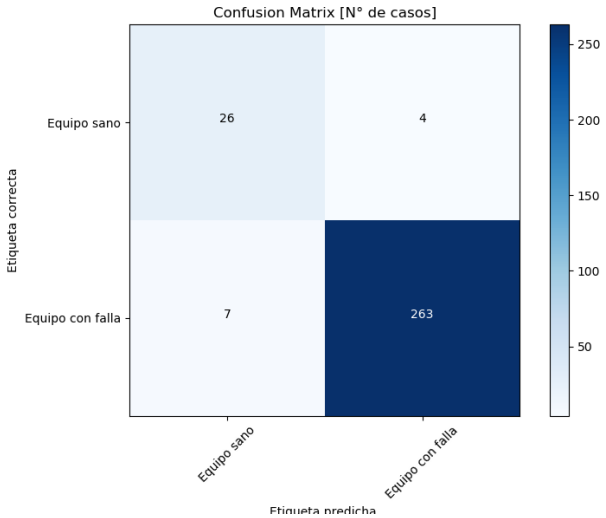

Figura 5.7: Gráfico de la función de pérdida del entrenamiento.

Para llegar los resultados indicados en la **Tabla 5.1** se fue probando variando con los siguientes parámetros de la red, utilizando los valores que se indican en la **Tabla 5.2**.

Tabla 5.2: Selección de par´metros para la Red Neuronal

Parámetro	Valor mínimo	Intervalos	Valor máximo
N° de capas	3	1	8
Dropout	0	0.05	0.3
Batch Size	200	100	500
N° de épocas	80	20	200

5.3. Análisis de los resultados

Respecto al proceso para la generación de los datos simulados y el entrenamiento de la red es importante indicar que todas las redes entrenadas tuvieron resultados con un *accuracy* superior al 97 % sin tener que realizar mayores modificaciones en la arquitectura de la red. Esto puede ser debido a que los parámetros de las simulaciones están dispuestos de tal manera que se existan diferencias notables entre cada caso, lo que facilita el entrenamiento de la ANN. En la **Figura 5.5** se puede observar la función de pérdida del conjunto de entrenamiento y validación, entre los cuales existe poca diferencia, por lo que está claro que la red no tiene problemas de *overfitting*.

Lo que no resulta trivial es la selección de los parámetros de la simulaciones para obtener un *accuracy* alto analizando los datos experimentales. El proceso de la modificación de los parámetros de la simulación, el entrenamiento de la red con los datos simulados y la clasificación de los datos experimentales es largo, pues existe una gran cantidad de combinaciones que van aumentando rápidamente con la cantidad de parámetros de la simulación.

Respecto a los resultados obtenidos se puede indicar que la red tiende identificar más fácilmente las fallas que los equipos en buen estado, lo cual es preferible, pues si en caso de aplicarse en algún equipo industrial, si existe una falla y la red indica que no hay problemas existe la probabilidad de que se genere una falla catastrófica. Ahora, en caso de ser aplicar esta red en equipos reales se pueden utilizar distintas metodologías que faciliten el diagnóstico del equipo, por ejemplo, se pueden tomar varias mediciones y hacer que la red clasifique todas ellas, para finalmente concluir a partir del resultado según la clasificación de todos los datos y no solamente una medición.

Capítulo 6

Discusión

Respecto a las simulaciones con el modelo fenomenológico se puede decir que el estudio del equipo a través del modelo fenomenológico se basa casi completamente en la cinemática del equipo, por lo que es importante tener claro de qué manera se relacionan los engranajes principalmente a partir de la cantidad de dientes de cada uno. Respecto a la inclusión de fallas el hecho de incluir variaciones en las frecuencias de engrane y ruido entregó al modelo la capacidad de acercarse a los datos experimentales, pues sin estas características los espectros habrían sido mucho más simples y por lo tanto la red entrenada con estos datos podría no haber clasificado tan bien. Respecto a esto es importante indicar que todos los parámetros utilizados en el modelo tienen que ser calibrados en función de las mediciones experimentales, por lo que se debe conocer las distribuciones de estos, por ejemplo, en este caso el estudio se basó principalmente en las componentes del *Espectro de Fourier*, sin embargo, si se quisieran agregar otros parámetros, por ejemplo la amplitud RMS de la velocidad o la aceleración, se debería hacer un estudio estadístico de la distribución de este parámetro en mediciones experimentales para poder agregarlo de alguna manera a la simulación. Un problema respecto al ejemplo anterior es que la amplitud RMS de la vibración depende de las características y dimensiones del equipo, y de otros parámetros que no se tuvieron en consideración en la simulación, mientras que con la metodología aplicada el *Espectro de Fourier* está normalizado, lo que facilita la comparación entre mediciones. Respecto a esto se debe indicar que un modelo más completo, por ejemplo, un modelo dinámico de los engranajes planetarios tiene en consideración la masa de los componentes, por lo que no tendría que ser calibrado en función de los parámetros experimentales,y por lo tanto sería mucho más útil a la hora de simular los datos para el entrenamiento de la red.

Respecto a la simulación de las fallas en el equipo se debe tener en consideración que uno de los elementos más importantes es la variabilidad de algunos parámetros, pues eso le podría permitir a la red el no ser tan rígida y adaptarse frente a distintas situaciones. Hay que tener en consideración que la simulación no tiene en cuenta una gran cantidad de parámetros:

- Masa de los engranajes
- Carga aplicada a la salida.
- Torque entregado por el motor.
- Temperatura del equipo
- Material de los engranajes

Respecto a las mediciones experimentales se puede decir que el equipo utilizado no fue ideal, pues tiene varios factores que dificultan el análisis de los datos. Uno de estos factores es el hecho de que al estar conectado a una batería, las condiciones de operación van cambiando a medida que esta se descarga, modificando la potencia del motor y la velocidad de giro del eje de salida, en cambio, si se utilizara un motor que se pudiera conectar a la red no se tendrían tantas complicaciones en este aspecto. Ahora, a pesar de que esto resulte ser una complicación, existe una gran variedad de equipos en la industria que funcionan con carga y velocidad variable, por lo que las metodologías de análisis desarrolladas en este trabajo podrían ser aplicadas a su estudio, en particular el análisis de la frecuencia del carrier a partir de la envolvente.

Otro de los problemas con las mediciones experimentales es que la temperatura del motor aumenta rápidamente con su uso, pues no está diseñado para funcionar durante mucho tiempo, lo que hace que las condiciones de operación también varíen, en particular debido a la dilatación de los componentes debido al aumento en la temperatura probablemente existan cambios respecto en las formas de onda respecto a un caso en que el equipo se encuentre a una misma temperatura. Esto también podría solucionare utilizando un montaje experimental con un motor que funcione en estado estacionario. Debido a esto es también que la cantidad de mediciones que se tomaron fue limitada, pudiendo disponer de una mayor cantidad de datos.

Se debe tener en cuenta que en este estudio se utilizó una carga constante para todas las mediciones, sin embargo, es posible que al variar la carga aplicada sobre el eje de salida se obtengan distintos resultados, por lo que también es un parámetro que se debería estudiar.

Respecto a las mediciones experimentales y sus procesamientos se puede decir que se debe tener en cuenta que corresponden a un caso en particular, y sería de interés estudiar si qué tanto cambian la distribución de frecuencias y amplitudes en la transformada normalizada si se utiliza un equipo de las mismas proporciones pero de distintas dimensiones. En caso de que un equipo de distintas dimensiones posea un *Espectro de Fourier normalizado* con distintas características se debería estudiar qué parámetros afectan a estas características.

Es importante indicar que dentro de las componentes estudiadas en los espectros estaban la de 51 [Ordenes], asociada a la frecuencia de engrane, y de 6 [Ordenes], asociado a la frecuencia de giro del sol, pero en ningún caso se encontró una componente de gran amplitud a alguna de las frecuencias de falla con las que se realizó el modelo fenomenológico, lo cual dificultó el análisis de los datos y la clasificación de los datos según su modo de falla. Tal como se observó en el capítulo de procesamiento de los datos experimentales se encontró que a pesar de que existe un pequeño aumento en las componentes asociadas a la frecuencias de falla para el caso de los equipos con distintos grados de severidad de falla, no siempre existen una relación directa entre la severidad de la falla en sus distintos niveles, lo que dificulta enormemente la clasificación de los datos.

Respecto a los resultados obtenidos con todos los procesamientos aplicados se debe comentar que de manera general es fácil identificar las fallas en el equipo utilizando el método MED y con ninguno de los otros métodos se obtuvieron resultados tan concluyentes. Una razón de esto podría ser que el filtro MED busca resaltar los impactos de la señal, sin embargo, el método ARMED, el cual utiliza un filtro AR antes del filtro MED no obtuvo resultados tan claro en la diferenciación de los casos con y sin falla, lo cual podría ser debido a que el filtro AR aplicando antes quizás esté eliminando algunas componentes importantes al tratar de solamente buscar un aumento en la kurtosis de la señal.

Respecto a la metodología aplicada en este estudio se debe indicar que se realizó la simulación con el modelo fenomenológico antes que las mediciones de los datos experimentales y su análisis, sin embargo, para poder determinar los parámetros más importantes del modelo se recomendaría realizar primero las mediciones experimentales y el análisis de sus parámetros, siguiendo una metodología como la indicada a continuación:

- Estudiar estadísticamente los datos experimentales para colocar la mayor cantidad de información posible con los distintos parámetros y procedimientos.
- Generar un modelo de simulaciones que tenga en consideración una manera de poder generar todos esos parámetros
- Entrenar una red neuronal con los datos sacados del modelo y corroborar el funcionamiento de la red entrenada con datos simulados en el modelo.
- Finalmente debe estudiar como varían las estadísticas para distintos reductores planetarios semejantes, para distintas dimensiones, y conocer si hay una correlación o no.

Respecto a la ANN se puede decir que el foco principal de esta sección del trabajo estuvo en la modificación de los parámetros de la simulación y de la red para poder obtener los mejores resultados posibles en el análisis de los datos experimentales, aunque es el primero en el que se tuvo que realizar más casos para mejorar los resultados, pues los parámetros de la red no tuvieron que ser modificados para obtener altos *accuracy* con los datos simulados. Este proceso es bastante tedioso pues requiere seguir los siguientes pasos para cada parámetro que se quiera modificar:

1. Simular las formas de onda con los parámetros seleccionados.
2. Aplicar los procesamientos a las formas de onda procesadas.
3. Entrenar la ANN con los datos de los procesamientos simulados.
4. Clasificar los datos experimentales con la red entrenada con datos simulados.
5. Estudiar si el resultado es mejor o peor que otros casos y modificar los parámetros de la simulación para repetir nuevamente todos los pasos.

Debido a la gran cantidad de pasos que toma observar el efecto en la modificación de 1 solo parámetro es que se recomendaría que para poder realizar un análisis completo y mucho más rápido, se debería programar todo el procedimiento para analizar los resultados frente a un vector de parámetros.

Es importante indicar que el hecho de poder generar una gran cantidad de datos para el entrenamiento con la simulación es sumamente útil porque requiere únicamente de recursos computacionales una vez que todo está hecho, además, entregar una gran cantidad de datos siempre es una buena práctica en el entrenamiento de una ANN.

Conclusión

El objetivo de la memoria fue logrado, ya que se logró generar un algoritmo de *Aprendizaje Profundo* entrenado con datos simulados de un *modelo fenomenológico*, el cual permite la identificación de falla por grietas en la base de los dientes. Respecto al desarrollo del trabajo se pueden realizar las siguientes conclusiones:

- Es posible programar una simulación de vibraciones mecánicas con semejanza a datos experimentales utilizando parámetros normalizados.
- Para poder agregar parámetros a la simulación se debe conocer la distribución de sus valores según los distintos casos estudiados a partir de datos experimentales.
- El montaje experimental utilizado dificultó el análisis de los datos debido a la potencia y la velocidad de giro variable del equipo, así como la temperatura del equipo.
- En un motor de partida las frecuencias que mayor amplitud tienen en el espectro son las de engrane y de giro del sol, ya sea con o son procesamientos aplicados.
- Con los procesamientos aplicados es muy difícil poder realizar la clasificación de los distintos modos de falla o de la severidad de las falla debido a que no existen características que los diferencien a unos de otros.
- El procesamiento MED es excelente para la identificación de fallas en reductores planetarios pues filtran la señal de tal manera que se observan más claramente los impactos asociados a fallas.
- Ajustar los parámetros de la simulación el entrenamiento de la ANN y su uso como clasificador es un proceso largo debido a la cantidad de tiempo que requiere cada proceso.
- Una buen método de procesamiento para identificar la frecuencia de modulación en reductores planetarios es a través del análisis de la envolvente.
- El hecho de poder simular los datos no entrega límites respecto a la cantidad de mediciones con la que se entrena la ANN.

Existen varias maneras de poder mejorar el trabajo acá presentado, las cuales se van a indicar a continuación:

- Se puede modificar el tipo de simulación utilizada a una simulación dinámica, la cual tiene mucho más parámetros para introducir y por lo tanto generar casos más variados. Entre los parámetros que se pueden considerar está la masa del equipo, la dimensión de una grieta y la posición de esta, la carga apicada al eje de salida, el torque del motor de entrada y el material de los engranajes.
- Se puede utilizar un montaje experimental en donde se puedan regular una variables que en este estudio no se pudieron manejar libremente, como la velocidad de giro del motor y el torque aplicado por el mismo.
- Se podrían utilizar otras arquitecturas de redes o métodos de *Aprendizaje profundo*, como por ejemplo una CNN, para estudiar su rendimiento a la hora de realizar la clasificación de los datos.

Estos puntos pueden ser considerados para poder generar propuestas de trabajos que continuén la línea de lo acá expuesto.

Capítulo 7

Bibliografía

[1] Jon Boman and Gustav Stock. *Evaluation of methods for fault detection of planetary gears for nutrunners*. KTH Industrial tenik och management, 2014.

[2] Omar Dawood Mohammed. *Dynamic Modelling and Vibration Analysis for Gear Tooth Crack Detection*. Operation and Maintenance Engineering, Lulea University of Technology, 2015.

[3] Anand Deshpande and Manish Kumar. *Artificial Intelligence for Big Data: Complete Guide to Automating Big Data Solutions Using Artificial Intelligence Techniques*. Packt Publishing Ltd, 2018.

[4] Nicolás Gutiérrez. *Analisis de vibraciones de un reductor de velocidad planetario a través de modelos analíticos*. Departamento de Ingeniería Mecánica, Facultad de Ingeniería, Universidad de Chile, 2017.

[5] Power Mi Cloud Condition Monitoring. *Manual de análisis de vibraciones*. Power - Mi, 2018.

[6] Javier Parra. *Caracterización de síntomas vibratorios producidos por fallas en transmisiones planetarias*. Departamento de Ingeniería Mecánica , Facultad de Ingeniería, Universidad de Concepción, 2016.

[7] Wiggins R. Minimum entropy deconvolution. *Wester Geophysical Company*, 1978, 1977.

[8] Santhana Raj. Ar filter + minimum entropy deconvolution for bearing fault diagnosis.

[9] Nicolás Reyes. *Estudio y análisis dinámicro de señales prevenientes de cajas de engranajes*. Departamento de Ingeniería Mecánica, Facultad de Ingeniería, Universidad del Bío - Bío, 2014.

[10] Randall R.B. Sawalhi N and Endo H. The enhancement of fault detection and diagnosis in rolling element bearings using minimum entropy deconvolution combined with spectral kurtosis. *Mechanical Systems and Signal Processing*, 2007, 2007.

[11] Juan Torregrosa. *Diagnóstico de fallas en reductores planetarios mediante el análisis de vibraciones*. Departamento de Ingeniería Mecánica , Facultad de Ingeniería, Universidad de Concepción, 2011.

Apéndice A

SIMULACIONES

Simulación engranaje planetario [Caso Sin falla]

```
%% SIMULACION CON RUIDO Y DISTRUCION DE FRECUENCIAS [OK]

clc
clear all

fs = 51200;      % Frecuencia de adquisicion [Hz]

z_r = 50;        % Numero de dientes del ring gear [S/U]
z_s = 10;        % Numero de dientes del sol [S/U]
z_p = 20;        % Numero de dientes del planeta [S/U]
f_c = 44.75;     % Frecuencia de giro del Carrier [Hz]

f_s = (z_r + z_s) * f_c / z_s;          % Frecuencia de giro del sol [Hz]
f_m = (z_r * z_s * f_s)/(z_r + z_s);    % Frecuencia de engrane [Hz]

L = 100000;                     % Largo del vector de aceleracion.

t = 0:1/fs:(L-1)/fs;            % Vector de tiempo [s]

psi_m1 = - pi / 4;              % Fase de engrane del planeta 1.
psi_m2 = 4 * pi/3 - pi / 4;     % Fase de engrane del planeta 2.
psi_m3 = 2 * pi/3 - pi / 4;     % Fase de engrane del planeta 3.

psi_c1 = 0;             % Fase del planeta 1 [rad]
psi_c2 = 2 * pi/3;      % Fase del planeta 2 [rad]
psi_c3 = 4 * pi/3;      % Fase del planeta 3 [rad]

f_sampling = 1/mean(diff(t));    % Se calcula la frecuencia de muestreo [Hz]

n = 100;                        % Numero de senales que se generan  [*]
m = n/100;                      % Numero de senales finales despues de promediar
ALL = zeros(n,length(t));       % Se crea un vector para guardar las senales
F_AR_NORMAL = zeros(1,n);       % Se guardan las frecuencias de giro para cada caso [Hz]

f_var_m = 1;     % Variacion de la frecuencia de engrane [Hz] [*]
f_var_s = 1;     % Variacion de la frecuencia de entrada [Hz] [*]
f_var_c = 1;     % Variacion de la frecuencia de salida [Hz] [*]
f_var_fp = 1;    % Variacion en la frecuencia de falla [Hz] [*]

E_s = 0.05;      % Magnitud del ruido en la senal de giro de entrada [*]
E_c = 0.05;      % Magnitud del ruido en la senal de giro de salida [*]
E_m = 0.05;      % Magnitud del ruido en la senal de engrane [*]

amp = 0;         % Le doy amplitud a la senal cuadrada [g] [*]

for i = 1:n

    X =['Porcentaje de avance NORMAL (1 de 2): ',num2str(i*100/n),' %];     % Se determina el porcentaje de
        avance del proceso [%]
    disp(X) % Se indica en la pantalla el porcentaje de avance [%]

    var_1 = normrnd(0,1,1,1);       % Variacion en la frecuencia de engrane [Hz]
    f_f1 = f_m + f_var_m * var_1;   % Frecuencia final de engrane [Hz]

    var_2 = normrnd(0,1,1,1);       % Variacion en la frecuencia del sol [Hz]
    f_f2 = f_s + f_var_s * var_2;   % Frecuencia de giro de entrada [Hz]

    var_3 = normrnd(0,1,1,1);                   % Variacion en la frecuencia del carrier [Hz]
    f_f3 = f_c + f_var_c * var_3;   % Frecuecia de giro de salida [Hz]

    F_AR_NORMAL(1,i) = f_f3;     % Se guarda la frecuencia de giro del Carrier

```

```
62      P_eng1 = 0.5 * (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m1));  % Senal del planeta 1 sin
            modulacion [g]
63      P_eng2 = 0.5 * (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m2));  % Senal del planeta 2 sin
            modulacion [g]
64      P_eng3 = 0.5 * (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m3));  % Senal del planeta 3 sin
            modulacion [g]
65
66      P_eng1 =  P_eng1 + E_m * normrnd(0,1,1,length(t));  % Se agrega ruido a la senal original.
67      P_eng2 =  P_eng2 + E_m * normrnd(0,1,1,length(t));  % Se agrega ruido a la senal original.
68      P_eng3 =  P_eng3 + E_m * normrnd(0,1,1,length(t));  % Se agrega ruido a la senal original.
69
70      freq = f_f1/z_p;                    % Determino que la frecuencia de la senal cuadrada es la del planeta [
            Hz]
71
72      var_4 = normrnd(0,1,1,1);           % Variacion en la frecuencia de falla [Hz]
73      f_f4 = freq + f_var_fp * var_4;     % Frecuecia de falla del planeta[Hz]
74
75      offset = amp;                       % Le doy un offset a la senal cuadrada [g]
76      duty = 4;                           % Indico la duracion del ciclo de trabajo.
77
78      sq_wav = offset + amp*square(2 * pi * f_f4 .*t,duty);     % Genero la senal cuadrada con los parametros
            indicados [g]
79
80      P_eng1_mod = P_eng1 + sq_wav.*P_eng1;      % Genero la senal con falla.
81
82      MOD_1 = (sin(2 * pi * f_f3 * t + psi_c1)+1);    % Senal que modula al planeta 1.
83      MOD_2 = (sin(2 * pi * f_f3 * t + psi_c2)+1);    % Senal que modula al planeta 2.
84      MOD_3 = (sin(2 * pi * f_f3 * t + psi_c3)+1);    % Senal que modula al planeta 3.
85
86      S_1 = MOD_1 .* P_eng1_mod;   % Senal modulada del planeta 1.
87      S_2 = MOD_2 .* P_eng2;       % Senal modulada del planeta 2.
88      S_3 = MOD_3 .* P_eng3;       % Senal modulada del planeta 3.
89
90      S_mod = (S_1 + S_2 + S_3);    % Se suman las senales desde el punto de vista del sensor.
91
92      S_in = sin(2 * pi * (f_f2 * ones(1,length(t)) .* t)) + E_s * normrnd(0,1,1,length(t));      % Senal de
            giro de entrada.
93      S_out = sin(2 * pi * (f_f3 * ones(1,length(t)) .* t)) + E_c * normrnd(0,1,1,length(t));     % Senal de
            giro de salida.
94
95      VAR_IN =  0 + 0 * rand(1);
96      VAR_OUT = 0 + 0 * rand(1);
97      VAR_F = 1 + 0 * rand(1) ;
98
99      S = VAR_IN * S_in/max(S_in) + VAR_OUT * S_out/max(S_out) + VAR_F * S_mod/max(S_mod);    % Suma de todas
            las senales.
100
101     ALL(i,:) = S;
102 end
103
104  A = zeros(m,L);          % Para guardar el promedio de las senales en el tiempo
105  F = zeros(m,1);          % Para guardar el promedio de las frecuencias de giro
106
107  %PROMEDIO DE A 100 SENALES
108
109  for i = 100:100:n
110      X =['Porcentaje de avance PLANETA (2 de 2): ',num2str(i*100/n),' %'];     % Se determina el porcentaje
             de avance del proceso [%]
111      disp(X)
112      w = mean(ALL(i-99:i,:));
113      f = mean(F_AR_NORMAL(1,i-99:i));
114      A(i/100,:) = w/max(w);
115      F(i/100,:) = f;
116  end
117
118 %% GRAFICOS DE LOS DATOS
119
120 % SE GENERAN SENALES SIN MODIFICACIONES
121
122 P_eng1_norm = 0.5* (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m1));  % Senal del planeta 1 sin
        modulacion [g]
123 P_eng2_norm = 0.5* (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m2));  % Senal del planeta 2 sin
        modulacion [g]
124 P_eng3_norm = 0.5* (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m3));  % Senal del planeta 3 sin
        modulacion [g]
125
126 % SE GRAFICA LA COMPARACION DE LA SENAL DEL PLANETA CON Y SIN RUIDO
127
128 figure
129 hold on
130 plot(t,P_eng1,'LineWidth',1.5);
131 plot(t,P_eng1_norm,'LineWidth',1.5);
132 xlim([0  0.002])
133 xlabel('Tiempo [Hz]', 'Interpreter', 'Latex', 'FontSize', 10);
134 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
135 title('Senales de los planetas','Interpreter','Latex','FontSize',10)
136 legend('Planeta 1 con ruido','Planeta 1 sin ruido')
137
138 %% SE GRAFICAN LAS SENALES DE LOS 3 PLANETAS
139
140 figure
141 hold on
142 plot(t,P_eng1,'LineWidth',1.5);
143 plot(t,P_eng2,'LineWidth',1.5);
144 plot(t,P_eng3,'LineWidth',1.5);
145 xlim([0  0.002])
```

```
xlabel('Tiempo [Hz]', 'Interpreter', 'Latex', 'FontSize', 10);
ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
title('Senales de los planetas','Interpreter','Latex','FontSize',10)
legend('Planeta 1','Planeta 2','Planeta 3')

%% SE GRFICA LA SENAL QUE MODULA AL PLANETA 1

figure
hold on
plot(t,MOD_1,'LineWidth',1.5);
xlim([0 0.07])
xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
title('Senal que modula al planeta 1','Interpreter','Latex','FontSize',10)

%% SE GRAFICA LA SENAL DEL PLANETA CON Y SIN FALLA LOCAL

%En este caso no hay falla.

figure
hold on
plot(t,P_eng1,'LineWidth',1.5);
plot(t,P_eng1_mod,'LineWidth',1.5);
plot(t, sq_wav,'LineWidth',1.5);
xlim([0 0.005])
xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
title('Falla en el planeta 1','Interpreter','Latex','FontSize',10)
legend('Planeta 1','Falla en el planeta 1')

%% SE GRAFICA LA SENAL DEL PLANETA 1 MODULADA A LA FRECUENCIA DEL CARRIER

figure
hold on
plot(t,S_1,'LineWidth',1.5);
xlim([0 0.05])
xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
title('Aceleracion del planeta 1 modulada a la frecuencia del carrier','Interpreter','Latex','FontSize',10)

%% SE GRAFICAN TODOS LOS PLANETAS MODULADOS A LA FRECUENCIA DEL CARRIER

figure
hold on
plot(t,S_1,'LineWidth',1.5);
plot(t,S_2,'LineWidth',1.5);
plot(t,S_3,'LineWidth',1.5);
xlim([0 0.035])
xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
    ,10)

%% SE GRAFICA LA SUMA DE LAS SENALES DE LOS PLANETAS

figure
hold on
plot(t,S,'LineWidth',1.5);
xlim([0 0.035])
xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
ylabel('Amplitud  $[m/s^2$]','Interpreter', 'Latex','FontSize', 10)
title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
    ,10)

%% SE CALCULA LA TDF DE UNO DE LOS PLANETAS MODULADOS

[F_S1,X_S1,Z_S1] = TDF(S_1,fs);
X_S1(1,1) = zeros(1,1);

figure
hold on
plot(F_S1, X_S1,'LineWidth',1.5);
xlim([0 2500])
xlabel('Frecuencia [s]', 'Interpreter', 'Latex', 'FontSize', 10);
ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
    ,10)

%% SE CALCULA LA TDF DE TODA LA SENAL

[F_A,X_A,Z_A] = TDF(A,fs);
X_A(1,1) = zeros(1,1);

[F_ALL,X_ALL,Z_ALL] = TDF(ALL(1,:),fs);
X_ALL(1,1) = zeros(1,1);

figure
hold on
plot(F_A, 5 * X_A,'LineWidth',1.5);
plot(F_ALL, X_ALL,'LineWidth',1.5);
xlim([2250 2320])
xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 12);
ylabel('Amplitud  $[m/s^2]$','Interpreter', 'Latex','FontSize', 12)
box on
% title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize
    ',10)
```

```
238  legend('Modelo con variacion de frecuencia','Modelo sin variacion de frecuencia')
```

Simulación engranaje planetario [Caso Falla de Planeta]

```
%% SIMULACION CON RUIDO Y DISTRUCION DE FRECUENCIAS [OK]

clc
clear all

fs = 51200;      % Frecuencia de adquisicion [Hz]

z_r = 50;        % Numero de dientes del ring gear [S/U]
z_s = 10;        % Numero de dientes del sol [S/U]
z_p = 20;        % Numero de dientes del planeta [S/U]
f_c = 44.75;     % Frecuencia de giro del Carrier [Hz]

f_s = (z_r + z_s) * f_c / z_s;          % Frecuencia de giro del sol [Hz]
f_m = (z_r * z_s * f_s)/(z_r + z_s);    % Frecuencia de engrane [Hz]

L = 100000;                      % Largo del vector de aceleracion.

t = 0:1/fs:(L-1)/fs;             % Vector de tiempo [s]

psi_m1 = - pi / 4;               % Fase de engrane del planeta 1.
psi_m2 = 4 * pi/3 - pi / 4;      % Fase de engrane del planeta 2.
psi_m3 = 2 * pi/3 - pi / 4;      % Fase de engrane del planeta 3.

psi_c1 = 0;              % Fase del planeta 1 [rad]
psi_c2 = 2 * pi/3;       % Fase del planeta 2 [rad]
psi_c3 = 4 * pi/3;       % Fase del planeta 3 [rad]

f_sampling = 1/mean(diff(t));    % Se calcula la frecuencia de muestreo [Hz]

n = 100;                         % Numero de senales que se generan  [*]
m = n/100;                       % Numero de senales finales despues de promediar
ALL = zeros(n,length(t));        % Se crea un vector para guardar las senales
F_AR_NORMAL = zeros(1,n);        % Se guardan las frecuencias de giro para cada caso [Hz]

f_var_m = 1;          % Variacion de la frecuencia de engrane [Hz] [*]
f_var_s = 1;          % Variacion de la frecuencia de entrada [Hz] [*]
f_var_c = 1;          % Variacion de la frecuencia de salida [Hz] [*]
f_var_fp = 0.3;       % Variacion en la frecuencia de falla [Hz] [*]

E_s = 0.05;       % Magnitud del ruido en la senal de giro de entrada [*]
E_c = 0.05;       % Magnitud del ruido en la senal de giro de salida [*]
E_m = 0.05;       % Magnitud del ruido en la senal de engrane [*]

amp = 0.5;     % Le doy amplitud a la senal cuadrada [g] [*]

for i = 1:n

    X =['Porcentaje de avance NORMAL (1 de 2): ',num2str(i*100/n),' %];     % Se determina el porcentaje de
        avance del proceso [%]
    disp(X) % Se indica en la pantalla el porcentaje de avance [%]

    var_1 = normrnd(0,1,1,1);        % Variacion en la frecuencia de engrane [Hz]
    f_f1 = f_m + f_var_m * var_1;    % Frecuencia final de engrane [Hz]

    var_2 = normrnd(0,1,1,1);        % Variacion en la frecuencia del sol [Hz]
    f_f2 = f_s + f_var_s * var_2;    % Frecuencia de giro de entrada [Hz]

    var_3 = normrnd(0,1,1,1);        % Variacion en la frecuencia del carrier [Hz]
    f_f3 = f_c + f_var_c * var_3;    % Frecuecia de giro de salida [Hz]

    F_AR_NORMAL(1,i) = f_f3;     % Se guarda la frecuencia de giro del Carrier

    P_eng1 = 0.5 * (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m1));  % Senal del planeta 1 sin
        modulacion [g]
    P_eng2 = 0.5 * (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m2));  % Senal del planeta 2 sin
        modulacion [g]
    P_eng3 = 0.5 * (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m3));  % Senal del planeta 3 sin
        modulacion [g]

    P_eng1 =  P_eng1 + E_m * normrnd(0,1,1,length(t));  % Se agrega ruido a la senal original.
    P_eng2 =  P_eng2 + E_m * normrnd(0,1,1,length(t));  % Se agrega ruido a la senal original.
    P_eng3 =  P_eng3 + E_m * normrnd(0,1,1,length(t));  % Se agrega ruido a la senal original.

    freq = f_f1/z_p;                 % Determino que la frecuencia de la senal cuadrada es la del planeta [
        Hz]

    var_4 = normrnd(0,1,1,1);        % Variacion en la frecuencia de falla [Hz]
    f_f4 = freq + f_var_fp * var_4;  % Frecuecia de falla del planeta[Hz]

    offset = amp;                    % Le doy un offset a la senal cuadrada [g]
    duty = 4;                        % Indico la duracion del ciclo de trabajo.

    sq_wav = offset + amp * square(2 * pi * f_f4 .*t,duty);     % Genero la senal cuadrada con los
        parametros indicados [g]

    P_eng1_mod = P_eng1 + sq_wav.*P_eng1;      % Genero la senal con falla.

    MOD_1 = (sin(2 * pi * f_f3 * t + psi_c1)+1);    % Senal que modula al planeta 1.
    MOD_2 = (sin(2 * pi * f_f3 * t + psi_c2)+1);    % Senal que modula al planeta 2.
    MOD_3 = (sin(2 * pi * f_f3 * t + psi_c3)+1);    % Senal que modula al planeta 3.

    S_1 = MOD_1 .* P_eng1_mod;   % Senal modulada del planeta 1.
    S_2 = MOD_2 .* P_eng2;       % Senal modulada del planeta 2.
    S_3 = MOD_3 .* P_eng3;       % Senal modulada del planeta 3.
```

```
89
90      S_mod = (S_1 + S_2 + S_3);    % Se suman las senales desde el punto de vista del sensor.
91
92      S_in = sin(2 * pi * (f_f2 * ones(1,length(t)) .* t)) + E_s * normrnd(0,1,1,length(t));      % Senal de
            giro de entrada.
93      S_out = sin(2 * pi * (f_f3 * ones(1,length(t)) .* t)) + E_c * normrnd(0,1,1,length(t));     % Senal de
            giro de salida.
94
95      VAR_IN =  0 + 0 * rand(1);
96      VAR_OUT = 0 + 0 * rand(1);
97      VAR_F = 1 + 0 * rand(1) ;
98
99      S = VAR_IN * S_in/max(S_in) + VAR_OUT * S_out/max(S_out) + VAR_F * S_mod/max(S_mod);    % Suma de todas
            las senales.
100
101     ALL(i,:) = S;
102 end
103
104  A = zeros(m,L);          % Para guardar el promedio de las senales en el tiempo
105  F = zeros(m,1);          % Para guardar el promedio de las frecuencias de giro
106
107  %PROMEDIO DE A 100 SENALES
108
109  for i = 100:100:n
110      X =['Porcentaje de avance PLANETA (2 de 2): ',num2str(i*100/n),' %];     % Se determina el porcentaje
             de avance del proceso [%]
111      disp(X)
112      w = mean(ALL(i-99:i,:));
113      f = mean(F_AR_NORMAL(1,i-99:i));
114      A(i/100,:) = w/max(w);
115      F(i/100,:) = f;
116  end
117
118 %% GRAFICOS DE LOS DATOS
119
120
121 P_eng1_norm = 0.5* (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m1));  % Senal del planeta 1 sin
        modulacion [g]
122 P_eng2_norm = 0.5* (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m2));  % Senal del planeta 2 sin
        modulacion [g]
123 P_eng3_norm = 0.5* (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m3));  % Senal del planeta 3 sin
        modulacion [g]
124
125 % SE GRAFICA LA COMPARACION DE LA SENAL DEL PLANETA CON Y SIN RUIDO
126
127 figure
128 hold on
129 plot(t,P_eng1,'LineWidth',1.5);
130 plot(t,P_eng1_norm,'LineWidth',1.5);
131 xlim([0  0.001])
132 xlabel('Tiempo [Hz]', 'Interpreter', 'Latex', 'FontSize', 10);
133 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
134 title('Senales de los planetas','Interpreter','Latex','FontSize',10)
135 legend('Planeta 1 con ruido','Planeta 1 sin ruido')
136
137 %% SE GRAFICAN LAS SENALES DE LOS 3 PLANETAS
138
139 figure
140 hold on
141 plot(t,P_eng1,'LineWidth',1.5);
142 plot(t,P_eng2,'LineWidth',1.5);
143 plot(t,P_eng3,'LineWidth',1.5);
144 xlim([0  0.002])
145 xlabel('Tiempo [Hz]', 'Interpreter', 'Latex', 'FontSize', 10);
146 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
147 title('Senales de los planetas','Interpreter','Latex','FontSize',10)
148 legend('Planeta 1','Planeta 2','Planeta 3')
149
150 %% SE GRFICA LA SENAL QUE MODULA AL PLANETA 1
151
152 figure
153 hold on
154 plot(t,MOD_1,'LineWidth',1.5);
155 xlim([0  0.07])
156 xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
157 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
158 title('Senal que modula al planeta 1','Interpreter','Latex','FontSize',10)
159
160 %% SE GRAFICA LA SENAL DEL PLANETA CON Y SIN FALLA LOCAL
161
162 figure
163 hold on
164 plot(t,P_eng1,'LineWidth',1.5);
165 plot(t,P_eng1_mod,'LineWidth',1.5);
166 plot(t, sq_wav,'LineWidth',1.5);
167 xlim([0  0.02])
168 xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
169 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
170 title('Falla en el planeta 1','Interpreter','Latex','FontSize',10)
171 legend('Planeta 1','Falla en el planeta 1')
172
173 %% SE GRAFICA LA SENAL DEL PLANETA 1 MODULADA A LA FRECUENCIA DEL CARRIER
174
175 figure
176 hold on
177 plot(t,S_1,'LineWidth',1.5);
```

```
xlim([0 0.035])
xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
ylabel('Amplitud $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
title('Aceleracion del planeta 1 modulada a la frecuencia del carrier','Interpreter','Latex','FontSize',10)

%% SE GRAFICAN TODOS LOS PLANETAS MODULADOS A LA FRECUENCIA DEL CARRIER

figure
hold on
plot(t,S_1,'LineWidth',1.5);
plot(t,S_2,'LineWidth',1.5);
plot(t,S_3,'LineWidth',1.5);
xlim([0 0.035])
xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
ylabel('Amplitud $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
    ,10)

%% SE GRAFICA LA SUMA DE LAS SENALES DE LOS PLANETAS

figure
hold on
plot(t,S,'LineWidth',1.5);
xlim([0 0.035])
xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
ylabel('Amplitud $[m/s^2$]','Interpreter', 'Latex','FontSize', 10)
title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
    ,10)

%% SE CALCULA LA TDF DE UNO DE LOS PLANETAS MODULADOS

[F_S1,X_S1,Z_S1] = TDF(S_1,fs);
X_S1(1,1) = zeros(1,1);

figure
hold on
plot(F_S1, X_S1,'LineWidth',1.5);
xlim([0 2500])
xlabel('Frecuencia [s]', 'Interpreter', 'Latex', 'FontSize', 10);
ylabel('Amplitud $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
    ,10)

%% SE CALCULA LA TDF DE TODA LA SENAL

[F_A,X_A,Z_A] = TDF(A,fs);
X_A(1,1) = zeros(1,1);

[F_ALL,X_ALL,Z_ALL] = TDF(ALL(1,:),fs);
X_ALL(1,1) = zeros(1,1);

figure
hold on
plot(F_A, 5 * X_A,'LineWidth',1.5);
plot(F_ALL, X_ALL,'LineWidth',1.5);
xlim([0 2320])
xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 10);
ylabel('Amplitud $[m/s^2]$','Interpreter', 'Latex','FontSize', 10)
% title('Comparacion de espectro con y sin variacion en frecuencia','Interpreter','Latex','FontSize',10)
legend('Modelo con variacion de frecuencia','Modelo sin variacion de frecuencia')
box on

%% GRAFICOS NORMALIZADOS

[F_A,X_A,Z_A] = TDF(A,fs);
X_A(1,1) = zeros(1,1);

figure
hold on
plot(F_A/f_c, X_A,'LineWidth',1.5);
xlim([0 60])
xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 12);
ylabel('Amplitud $[m/s^2]$','Interpreter', 'Latex','FontSize', 12)
% title('Equipo con falla en el planeta con eje de frecuencias normalizado','Interpreter','Latex','FontSize
    ',10)
% legend('Modelo con variacion de frecuencia','Modelo sin variacion de frecuencia')
box on
```

Simulación engranaje planetario [Caso Falla del Aro]

```
%% SIMULACION CON RUIDO Y DISTRUCION DE FRECUENCIAS [OK]

clc
clear all

fs = 51200;      % Frecuencia de adquisicion [Hz]

z_r = 50;        % Numero de dientes del ring gear [S/U]
z_s = 10;        % Numero de dientes del sol [S/U]
z_p = 20;        % Numero de dientes del planeta [S/U]
f_c = 44.75;     % Frecuencia de giro del Carrier [Hz]

f_s = (z_r + z_s) * f_c / z_s;          % Frecuencia de giro del sol [Hz]
f_m = (z_r * z_s * f_s)/(z_r + z_s);    % Frecuencia de engrane [Hz]

L = 20000;                      % Largo del vector de aceleracion.

t = 0:1/fs:(L-1)/fs;            % Vector de tiempo [s]

psi_m1 = - pi / 4;              % Fase de engrane del planeta 1.
psi_m2 = 4 * pi/3 - pi / 4;     % Fase de engrane del planeta 2.
psi_m3 = 2 * pi/3 - pi / 4;     % Fase de engrane del planeta 3.

psi_c1 = 0;             % Fase del planeta 1 [rad]
psi_c2 = 2 * pi/3;      % Fase del planeta 2 [rad]
psi_c3 = 4 * pi/3;      % Fase del planeta 3 [rad]

f_sampling = 1/mean(diff(t));   % Se calcula la frecuencia de muestreo [Hz]

n = 100;                        % Numero de senales que se generan  [*]
m = n/100;                      % Numero de senales finales despues de promediar
ALL = zeros(n,length(t));       % Se crea un vector para guardar las senales
F_AR_ARO_1 = zeros(1,n);        % Se guardan las frecuencias de giro para cada caso [Hz]

f_var_m = 1;    % Variacion de la frecuencia de engrane [Hz] [*]
f_var_s = 1;     % Variacion de la frecuencia de entrada [Hz] [*]
f_var_c = 1;     % Variacion de la frecuencia de salida [Hz] [*]
f_var_fp = 1;    % Variacion en la frecuencia de falla [Hz] [*]

E_s = 0.05;       % Magnitud del ruido en la senal de giro de entrada [*]
E_c = 0.05;       % Magnitud del ruido en la senal de giro de salida [*]
E_m = 0.05;       % Magnitud del ruido en la senal de engrane [*]

amp = 0.5;        % Le doy amplitud unitaria a la senal cuadrada [g][*]

for i = 1:n

    X =['Porcentaje de avance ARO (1 de 2): ',num2str(i*100/n),' %'];     % Se determina el porcentaje de
        avance del proceso [%]
    disp(X)                         % Se indica en la pantalla el porcentaje de avance [%]

    var_1 = normrnd(0,1,1,1);       % Variacion en la frecuencia de engrane [Hz]
    f_f1 = f_m + f_var_m * var_1;   % Frecuencia final de engrane [Hz]

    var_2 = normrnd(0,1,1,1);       % Variacion en la frecuencia del sol [Hz]
    f_f2 = f_s + f_var_s * var_2;   % Frecuencia de giro de entrada [Hz]

    var_3 = normrnd(0,1,1,1);       % Variacion en la frecuencia del carrier [Hz]
    f_f3 = f_c + f_var_c * var_3;   % Frecuecia de giro de salida [Hz]
    F_AR_ARO_1(1,i) = f_f3;

    P_eng1 = 0.5 * (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m1));  % Senal del planeta 1 sin
        modulacion [g]
    P_eng2 = 0.5 * (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m2));  % Senal del planeta 2 sin
        modulacion [g]
    P_eng3 = 0.5 * (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m3));  % Senal del planeta 3 sin
        modulacion [g]

    P_eng1 =  P_eng1 + E_m * normrnd(0,1,1,length(t));  % Se agrega ruido a la senal original.
    P_eng2 =  P_eng2 + E_m * normrnd(0,1,1,length(t));  % Se agrega ruido a la senal original.
    P_eng3 =  P_eng3 + E_m * normrnd(0,1,1,length(t));  % Se agrega ruido a la senal original.

    freq = f_m/z_r;                 % Determino que la frecuencia de la senal cuadrada es la del aro [Hz]

    var_4 = normrnd(0,1,1,1);        % Variacion en la frecuencia de falla [Hz]
    f_f4 = freq + f_var_fp * var_4;  % Frecuecia de falla del aro [Hz]

    offset = amp;                   % Le doy un offset a la senal cuadrada [g]
    duty = 1.5;                     % Indico la duracion del ciclo de trabajo.

    sq_wav1 = offset + amp*square(2 * pi * f_f4 .*t,duty) * 1;    % Genero la senal cuadrada con los
        parametros indicados [g]
    sq_wav2 = offset + amp*square(2 * pi* f_f4 .*t + 2* pi/3,duty) * 1;    % Genero la senal cuadrada con
        los parametros indicados [g]
    sq_wav3 = offset + amp*square(2 * pi* f_f4 .*t + 4* pi/3,duty) * 1;    % Genero la senal cuadrada con
        los parametros indicados [g]

    P_eng1_mod = P_eng1 + sq_wav1.* P_eng1;     % Genero la senal con falla.
    P_eng2_mod = P_eng2 + sq_wav2.* P_eng2;     % Genero la senal con falla.
    P_eng3_mod = P_eng3 + sq_wav3.* P_eng3;     % Genero la senal con falla.

    MOD_1 = (sin(2 * pi * f_f3 * t + psi_c1)+1);    % Senal que modula al planeta 1.
    MOD_2 = (sin(2 * pi * f_f3 * t + psi_c2)+1);    % Senal que modula al planeta 2.
    MOD_3 = (sin(2 * pi * f_f3 * t + psi_c3)+1);    % Senal que modula al planeta 3.
```

```
88
89      S_1 = MOD_1 .* P_eng1_mod;   % Senal modulada del planeta 1.
90      S_2 = MOD_2 .* P_eng2_mod;   % Senal modulada del planeta 2.
91      S_3 = MOD_3 .* P_eng3_mod;   % Senal modulada del planeta 3.
92
93      S_mod = (S_1 + S_2 + S_3);    % Se suman las senales desde el punto de vista del sensor.
94
95      S_in = sin(2 * pi * (f_f2 * ones(1,length(t)) .* t)) + E_s * normrnd(0,1,1,length(t));      % Senal de
            giro de entrada.
96      S_out = sin(2 * pi * (f_f3 * ones(1,length(t)) .* t)) + E_c * normrnd(0,1,1,length(t));     % Senal de
            giro de salida.
97
98      VAR_IN =  0 + 0 * rand(1);
99      VAR_OUT = 0 + 0 * rand(1);
100     VAR_F = 1 + 0 * rand(1) ;
101
102     S = VAR_IN * S_in/max(S_in) + VAR_OUT * S_out/max(S_out) + VAR_F * S_mod/max(S_mod);    % Suma de todas
            las senales.
103
104     ALL(i,:) = S;
105 end
106
107  A = zeros(m,L);           % Para guardar el promedio de las senales en el tiempo
108  F = zeros(m,1);           % Para guardar el promedio de las frecuencias de giro
109
110  %PROMEDIO DE A 100 SENALES
111
112  for i = 100:100:n
113      X =['Porcentaje de avance ARO (2 de 2): ',num2str(i*100/n),' %];     % Se determina el porcentaje de
            avance del proceso [%]
114      disp(X)
115      w = mean(ALL(i-99:i,:));
116      f = mean(F_AR_ARO_1(1,i-99:i));
117      A(i/100,:) = w/max(w);
118      F(i/100,:) = f;
119  end
120
121  %% GRAFICOS DE LOS DATOS
122
123
124 P_eng1_norm = 0.5* (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m1));  % Senal del planeta 1 sin
        modulacion [g]
125 P_eng2_norm = 0.5* (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m2));  % Senal del planeta 2 sin
        modulacion [g]
126 P_eng3_norm = 0.5* (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m3));  % Senal del planeta 3 sin
        modulacion [g]
127
128 % SE GRAFICA LA COMPARACION DE LA SENAL DEL PLANETA CON Y SIN RUIDO
129
130 figure
131 hold on
132 plot(t,P_eng1,'LineWidth',1.5);
133 plot(t,P_eng1_norm,'LineWidth',1.5);
134 xlim([0  0.001])
135 xlabel('Tiempo [Hz]', 'Interpreter', 'Latex', 'FontSize', 10);
136 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
137 title('Senales de los planetas','Interpreter','Latex','FontSize',10)
138 legend('Planeta 1 con ruido','Planeta 1 sin ruido')
139
140 %% SE GRAFICAN LAS SENALES DE LOS 3 PLANETAS
141
142 figure
143 hold on
144 plot(t,P_eng1,'LineWidth',1.5);
145 plot(t,P_eng2,'LineWidth',1.5);
146 plot(t,P_eng3,'LineWidth',1.5);
147 xlim([0  0.002])
148 xlabel('Tiempo [Hz]', 'Interpreter', 'Latex', 'FontSize', 10);
149 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
150 title('Senales de los planetas','Interpreter','Latex','FontSize',10)
151 legend('Planeta 1','Planeta 2','Planeta 3')
152
153 %% SE GRFICA LA SENAL QUE MODULA AL PLANETA 1
154
155 figure
156 hold on
157 plot(t,MOD_1,'LineWidth',1.5);
158 xlim([0  0.07])
159 xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
160 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
161 title('Senal que modula al planeta 1','Interpreter','Latex','FontSize',10)
162
163 %% SE GRAFICA LA SENAL DEL PLANETA CON Y SIN FALLA LOCAL
164
165 figure
166 hold on
167 % plot(t,P_eng1_mod,'LineWidth',1.5);
168 % plot(t,P_eng1_norm,'LineWidth',1.5);
169 % plot(t,P_eng2_mod,'LineWidth',1.5);
170 % plot(t,P_eng2_norm,'LineWidth',1.5);
171 plot(t,P_eng3_mod,'LineWidth',1.5);
172 plot(t,P_eng3_norm,'LineWidth',1.5);
173 % plot(t, sq_wav1,'LineWidth',1.5);
174 % plot(t, sq_wav2,'LineWidth',1.5);
175 plot(t, sq_wav3,'LineWidth',1.5);
176 xlim([0  0.04])
```

```
177 xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
178 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
179 title('Falla en el planeta 1','Interpreter','Latex','FontSize',10)
180
181 %% SE GRAFICA LA SENAL DEL PLANETA 1 MODULADA A LA FRECUENCIA DEL CARRIER
182
183 figure
184 hold on
185 plot(t,S_1,'LineWidth',1.5);
186 xlim([0 0.035])
187 xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
188 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
189 title('Aceleracion del planeta 1 modulada a la frecuencia del carrier','Interpreter','Latex','FontSize',10)
190
191 %% SE GRAFICAN TODOS LOS PLANETAS MODULADOS A LA FRECUENCIA DEL CARRIER
192
193 figure
194 hold on
195 plot(t,S_1,'LineWidth',1.5);
196 plot(t,S_2,'LineWidth',1.5);
197 plot(t,S_3,'LineWidth',1.5);
198 xlim([0 0.035])
199 xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
200 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
201 title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
        ,10)
202 legend('Planeta 1 modulado con falla en sol','Planeta 2 modulado con falla en sol','Planeta 3 modulado con
        falla en sol')
203
204 %% SE GRAFICA LA SUMA DE LAS SENALES DE LOS PLANETAS
205
206 figure
207 hold on
208 plot(t,S,'LineWidth',1.5);
209 xlim([0 0.035])
210 xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
211 ylabel('Amplitud  $[m/s^2$]','Interpreter', 'Latex','FontSize', 10)
212 title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
        ,10)
213
214 %% SE CALCULA LA TDF DE UNO DE LOS PLANETAS MODULADOS
215
216 [F_S1,X_S1,Z_S1] = TDF(S_1,fs);
217 X_S1(1,1) = zeros(1,1);
218
219 figure
220 hold on
221 plot(F_S1, X_S1,'LineWidth',1.5);
222 xlim([0 2500])
223 xlabel('Frecuencia [s]', 'Interpreter', 'Latex', 'FontSize', 10);
224 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
225 title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
        ,10)
226
227 %% SE CALCULA LA TDF DE TODA LA SENAL
228
229 [F_S,X_S,Z_S] = TDF(S,fs);
230 X_S(1,1) = zeros(1,1);
231
232 figure
233 hold on
234 plot(F_S, X_S,'LineWidth',1.5);
235 xlim([0 2500])
236 xlabel('Frecuencia [s]', 'Interpreter', 'Latex', 'FontSize', 10);
237 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
238 title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
        ,10)
```

Simulación engranaje planetario [Caso Falla del Sol]

```
%% SIMULACION CON RUIDO Y DISTRUCION DE FRECUENCIAS [OK]

clc
clear all

fs = 51200;      % Frecuencia de adquisicion [Hz]

z_r = 50;        % Numero de dientes del ring gear [S/U]
z_s = 10;        % Numero de dientes del sol [S/U]
z_p = 20;        % Numero de dientes del planeta [S/U]
f_c = 44.75;     % Frecuencia de giro del Carrier [Hz]

f_s = (z_r + z_s) * f_c / z_s;          % Frecuencia de giro del sol [Hz]
f_m = (z_r * z_s * f_s)/(z_r + z_s);    % Frecuencia de engrane [Hz]

L = 20000;                      % Largo del vector de aceleracion.

t = 0:1/fs:(L-1)/fs;            % Vector de tiempo [s]

psi_m1 = - pi / 4;              % Fase de engrane del planeta 1.
psi_m2 = 4 * pi/3 - pi / 4;     % Fase de engrane del planeta 2.
psi_m3 = 2 * pi/3 - pi / 4;     % Fase de engrane del planeta 3.

psi_c1 = 0;             % Fase del planeta 1 [rad]
psi_c2 = 2 * pi/3;      % Fase del planeta 2 [rad]
psi_c3 = 4 * pi/3;      % Fase del planeta 3 [rad]

f_sampling = 1/mean(diff(t));    % Se calcula la frecuencia de muestreo [Hz]

n = 100;                        % Numero de senales que se generan  [*]
m = n/100;                      % Numero de senales finales despues de promediar
ALL = zeros(n,length(t));       % Se crea un vector para guardar las senales
F_AR_SOL_1 = zeros(1,n);        % Se guardan las frecuencias de giro para cada caso [Hz]

f_var_m = 1;     % Variacion de la frecuencia de engrane [Hz] [*]
f_var_s = 1;     % Variacion de la frecuencia de entrada [Hz] [*]
f_var_c = 1;     % Variacion de la frecuencia de salida [Hz] [*]
f_var_fp = 1;    % Variacion en la frecuencia de falla [Hz] [*]

E_s = 0.05;       % Magnitud del ruido en la senal de giro de entrada [*]
E_c = 0.05;       % Magnitud del ruido en la senal de giro de salida [*]
E_m = 0.05;       % Magnitud del ruido en la senal de engrane [*]

amp = 0.5;        % Le doy amplitud unitaria a la senal cuadrada [g]

for i = 1:n

    X =['Porcentaje de avance SOL (1 de 2): ',num2str(i*100/n),' %];     % Se determina el porcentaje de
        avance del proceso [%]
    disp(X)                                          % Se indica en la pantalla el porcentaje de avance [%]

    var_1 = normrnd(0,1,1,1);       % Variacion en la frecuencia de engrane [Hz]
    f_f1 = f_m + f_var_m * var_1;   % Frecuencia final de engrane [Hz]

    var_2 = normrnd(0,1,1,1);       % Variacion en la frecuencia del sol [Hz]
    f_f2 = f_s + f_var_s * var_2;   % Frecuencia de giro de entrada [Hz]

    var_3 = normrnd(0,1,1,1);       % Variacion en la frecuencia del carrier [Hz]
    f_f3 = f_c + f_var_c * var_3;   % Frecuecia de giro de salida [Hz]

    F_AR_SOL_1(1,i) = f_f3;

    P_eng1 = 0.5 * (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m1));  % Senal del planeta 1 sin
        modulacion [g]
    P_eng2 = 0.5 * (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m2));  % Senal del planeta 2 sin
        modulacion [g]
    P_eng3 = 0.5 * (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m3));  % Senal del planeta 3 sin
        modulacion [g]

    P_eng1 =  P_eng1 + E_m * normrnd(0,1,1,length(t));  % Se agrega ruido a la senal original.
    P_eng2 =  P_eng2 + E_m * normrnd(0,1,1,length(t));  % Se agrega ruido a la senal original.
    P_eng3 =  P_eng3 + E_m * normrnd(0,1,1,length(t));  % Se agrega ruido a la senal original.

    freq = f_m/z_s;                 % Determino que la frecuencia de la senal cuadrada es la del sol [Hz]

    var_4 = normrnd(0,1,1,1);        % Variacion en la frecuencia de falla [Hz]
    f_f4 = freq + f_var_fp * var_4;  % Frecuecia de falla del sol [Hz]

    offset = amp;                   % Le doy un offset a la senal cuadrada [g]
    duty = 8;                       % Indico la duracion del ciclo de trabajo.

    sq_wav1 = offset + amp*square(2 * pi * f_f4 .*t,duty) * 1;              % Genero la senal cuadrada con
        los parametros indicados [g]
    sq_wav2 = offset + amp*square(2 * pi* f_f4 .*t + 4* pi/3,duty) * 1;     % Genero la senal cuadrada con
        los parametros indicados [g]
    sq_wav3 = offset + amp*square(2 * pi* f_f4 .*t + 2* pi/3,duty) * 1;     % Genero la senal cuadrada con
        los parametros indicados [g]

    P_eng1_mod = P_eng1 + sq_wav1.* P_eng1;      % Genero la senal con falla.
    P_eng2_mod = P_eng2 + sq_wav2.* P_eng2;      % Genero la senal con falla.
    P_eng3_mod = P_eng3 + sq_wav3.* P_eng3;      % Genero la senal con falla.

    MOD_1 = (sin(2 * pi * f_f3 * t + psi_c1)+1);    % Senal que modula al planeta 1.
    MOD_2 = (sin(2 * pi * f_f3 * t + psi_c2)+1);    % Senal que modula al planeta 2.
```

```
88      MOD_3 = (sin(2 * pi * f_f3 * t + psi_c3)+1);    % Senal que modula al planeta 3.
89
90      S_1 = MOD_1 .* P_eng1_mod;   % Senal modulada del planeta 1.
91      S_2 = MOD_2 .* P_eng2_mod;   % Senal modulada del planeta 2.
92      S_3 = MOD_3 .* P_eng3_mod;   % Senal modulada del planeta 3.
93
94      S_mod = (S_1 + S_2 + S_3);    % Se suman las senales desde el punto de vista del sensor.
95
96      S_in = sin(2 * pi * (f_f2 * ones(1,length(t)) .* t)) + E_s * normrnd(0,1,1,length(t));      % Senal de
            giro de entrada.
97      S_out = sin(2 * pi * (f_f3 * ones(1,length(t)) .* t)) + E_c * normrnd(0,1,1,length(t));     % Senal de
            giro de salida.
98
99      VAR_IN =  0 + 0 * rand(1);
100     VAR_OUT = 0 + 0 * rand(1);
101     VAR_F = 1 + 0 * rand(1) ;
102
103     S = VAR_IN * S_in/max(S_in) + VAR_OUT * S_out/max(S_out) + VAR_F * S_mod/max(S_mod);    % Suma de todas
            las senales.
104     ALL(i,:) = S;
105 end
106
107  A = zeros(m,L);          % Para guardar el promedio de las senales en el tiempo
108  F = zeros(m,1);          % Para guardar el promedio de las frecuencias de giro
109
110  % PROMEDIO DE A 100 SENALES
111
112  for i = 100:100:n
113      X =['Porcentaje de avance SOL (2 de 2): ',num2str(i*100/n),' %];     % Se determina el porcentaje de
             avance del proceso [%]
114      disp(X)
115      w = mean(ALL(i-99:i,:));
116      f = mean(F_AR_SOL_1(1,i-99:i));
117      A(i/100,:) = w/max(w);
118      F(i/100,:) = f;
119  end
120
121 %% GRAFICOS DE LOS DATOS
122
123
124 P_eng1_norm = 0.5* (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m1));  % Senal del planeta 1 sin
        modulacion [g]
125 P_eng2_norm = 0.5* (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m2));  % Senal del planeta 2 sin
        modulacion [g]
126 P_eng3_norm = 0.5* (1 + sin(2 * pi * (f_f1 * ones(1,length(t)) .* t) + psi_m3));  % Senal del planeta 3 sin
        modulacion [g]
127
128 % SE GRAFICA LA COMPARACION DE LA SENAL DEL PLANETA CON Y SIN RUIDO
129
130 figure
131 hold on
132 plot(t,P_eng1,'LineWidth',1.5);
133 plot(t,P_eng1_norm,'LineWidth',1.5);
134 xlim([0  0.001])
135 xlabel('Tiempo [Hz]', 'Interpreter', 'Latex', 'FontSize', 10);
136 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
137 title('Senales de los planetas','Interpreter','Latex','FontSize',10)
138 legend('Planeta 1 con ruido','Planeta 1 sin ruido')
139
140 %% SE GRAFICAN LAS SENALES DE LOS 3 PLANETAS
141
142 figure
143 hold on
144 plot(t,P_eng1,'LineWidth',1.5);
145 plot(t,P_eng2,'LineWidth',1.5);
146 plot(t,P_eng3,'LineWidth',1.5);
147 xlim([0  0.002])
148 xlabel('Tiempo [Hz]', 'Interpreter', 'Latex', 'FontSize', 10);
149 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
150 title('Senales de los planetas','Interpreter','Latex','FontSize',10)
151 legend('Planeta 1','Planeta 2','Planeta 3')
152
153 %% SE GRFICA LA SENAL QUE MODULA AL PLANETA 1
154
155 figure
156 hold on
157 plot(t,MOD_1,'LineWidth',1.5);
158 xlim([0  0.07])
159 xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
160 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
161 title('Senal que modula al planeta 1','Interpreter','Latex','FontSize',10)
162
163 %% SE GRAFICA LA SENAL DEL PLANETA CON Y SIN FALLA LOCAL
164
165 figure
166 hold on
167 plot(t,P_eng1_mod,'LineWidth',1.5);
168 plot(t,P_eng2_mod,'LineWidth',1.5);
169 plot(t,P_eng3_mod,'LineWidth',1.5);
170 plot(t, sq_wav1,'LineWidth',1.5);
171 plot(t, sq_wav2,'LineWidth',1.5);
172 plot(t, sq_wav3,'LineWidth',1.5);
173 xlim([0  0.01])
174 xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
175 ylabel('Amplitud  $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
176 title('Falla en el planeta 1','Interpreter','Latex','FontSize',10)
```

```
177 legend('Planeta 1 con fala en sol','Planeta 2 con fala en sol','Planeta 3 con fala en sol')
178
179 %% SE GRAFICA LA SENAL DEL PLANETA 1 MODULADA A LA FRECUENCIA DEL CARRIER
180
181 figure
182 hold on
183 plot(t,S_1,'LineWidth',1.5);
184 xlim([0 0.035])
185 xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
186 ylabel('Amplitud $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
187 title('Aceleracion del planeta 1 modulada a la frecuencia del carrier','Interpreter','Latex','FontSize',10)
188
189 %% SE GRAFICAN TODOS LOS PLANETAS MODULADOS A LA FRECUENCIA DEL CARRIER
190
191 figure
192 hold on
193 plot(t,S_1,'LineWidth',1.5);
194 plot(t,S_2,'LineWidth',1.5);
195 plot(t,S_3,'LineWidth',1.5);
196 xlim([0 0.035])
197 xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
198 ylabel('Amplitud $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
199 title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
        ,10)
200 legend('Planeta 1 modulado con falla en sol','Planeta 2 modulado con falla en sol','Planeta 3 modulado con
        falla en sol')
201
202 %% SE GRAFICA LA SUMA DE LAS SENALES DE LOS PLANETAS
203
204 figure
205 hold on
206 plot(t,S,'LineWidth',1.5);
207 xlim([0 0.035])
208 xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
209 ylabel('Amplitud $[m/s^2$]','Interpreter', 'Latex','FontSize', 10)
210 title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
        ,10)
211
212 %% SE CALCULA LA TDF DE UNO DE LOS PLANETAS MODULADOS
213
214 [F_S1,X_S1,Z_S1] = TDF(S_1,fs);
215 X_S1(1,1) = zeros(1,1);
216
217 figure
218 hold on
219 plot(F_S1, X_S1,'LineWidth',1.5);
220 xlim([0 2500])
221 xlabel('Frecuencia [s]', 'Interpreter', 'Latex', 'FontSize', 10);
222 ylabel('Amplitud $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
223 title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
        ,10)
224
225 %% SE CALCULA LA TDF DE TODA LA SENAL
226
227 [F_S,X_S,Z_S] = TDF(S,fs);
228 X_S(1,1) = zeros(1,1);
229
230 figure
231 hold on
232 plot(F_S, X_S,'LineWidth',1.5);
233 xlim([0 2500])
234 xlabel('Frecuencia [s]', 'Interpreter', 'Latex', 'FontSize', 10);
235 ylabel('Amplitud $[m/s^2$','Interpreter', 'Latex','FontSize', 10)
236 title('Aceleracion de los planetas modulados a la frecuencia del carrier','Interpreter','Latex','FontSize'
        ,10)
```

Función Transformada de Fourier [Aplicada a todas las funciones]

```
1  function [f,Y,Z] = TDF(x,Fs)
2
3  %Extraido de www.mathworks.com
4
5  Y = fft(x);
6  L = length(x);
7  Z = angle(round(fft(x),7));
8  Z = Z(1:L/2+1);
9  Y = abs(Y/L);
10 Y = Y(1:L/2+1);
11 Y(2:end-1) = 2*Y(2:end-1);
12 f = (Fs/L)*(0:L/2);
13
14 end
```

Apéndice B

EXTRACCIÓN Y ENVOLVENTE

Código para la extracción de datos [Formato UFF]

```
%SE EXTRAE EL PRIMER VECTOR DEL UFF [OK]

clc
clear all
close all

[data, kk, oo] = READUFF('PLANETA  3 [29 - 06].UFF');    %Se extraen los datos del UFF

t = data{1,3}.x;              %Se crea un vector de tiempo [s]
acc = data{1,3}.measData;     %Se extrae el vector de aceleracion [g]
ts = data{1,3}.dx;            %Se extrae el paso de tiempo [s]
fs = 1/ts;                    %Se calcula la frecuencia de adquisicion [hz]

[a,b] = size(data);           %Se calcula la cantidad de datos que tiene el UFF
c = b-2;                      %Se deterina la cantidad de datos relevantes en el UFF

ACC = zeros(c,length(acc)); %Creo un vector vacio para guardar las aceleraciones
T = zeros(c,length(t));       %Creo un vector vacio para guardar los tiempos

%Se extraen los datos en una iteracion

for i = 1:c
    ACC(i,:) = data{1,2+i}.measData;
    T(i,:) = data{1,2+i}.x;
end


figure

for i = 1:c
    hold on
    plot(T(1,:),ACC(i,:));
end

xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
xlim([0 max(T(1,:))])
ylabel('Aceleracion  [g]','Interpreter', 'Latex','FontSize', 10)
title('Vibraciones para falla en el planeta nivel 4','Interpreter','Latex','FontSize',16)
legend('Medicion 1','Medicion 2','Medicion 3')

d = 4;          %Numero de secciones que voy a tomar de cada dato [*]
in = 100000;     %Determino el inicio para la seleccion de la aceleracion [*]
en = 40000;      %Determino el largo del vector de aceleracion [*]
tot = in + d * en;  %Se calcula el final para la seleccion de la aceleracion [*]

ACC_MOD = zeros(c * d, en);
T_MOD = zeros(c * d, en);

for i = 1:c
    for j = 1:d
        ACC_MOD(d * (i-1) + j,:) = ACC(i,(in + (j-1) * en: in + j * en - 1));  %Se corta la aceleracion
            segun lo indicado
        T_MOD(d * (i-1) + j,:)= T(i,(in + (j-1) * en: in + j * en - 1));       %Se corta el tiempo segun lo
            indicado
    end
end

%Se grafican todas las aceleraciones

for i = 1:c * d
    figure
    hold on
```

```
61      plot(T_MOD(1,:),ACC_MOD(i,:));
62      pause(1)
63      xlim([min(T_MOD(1,:)) max(T_MOD(1,:))])
64      xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 10);
65      ylabel('Aceleracion  [g]','Interpreter', 'Latex','FontSize', 10)
66      title('Comparacion de aceleraciones en la caracasa del reductor','Interpreter','Latex','FontSize',16)
67  end
68
69  ACC_PLANETA_3 = ACC_MOD;
70
71  T_PLANETA_3 = T_MOD;
72
73  save('T_PLANETA_3','T_PLANETA_3')
74  save('ACC_PLANETA_3','ACC_PLANETA_3')
```

Función lectura de datos [Formato UFF]

```
1   function [UffDataSets, Info, errmsg] = READUFF(varargin)
2   %READUFF Reads UFF (Universal File Format) files of 10 types:
3   %   151, 15, 18, 55, 58, 82, 164, 2411, 2412 and also the hybrid one, 58b
4   %
5   %   Works in Matlab/Octave.
6   %
7   %   Usage:
8   %   [UffDataSets, Info, errmsg] = readuff(fileName, 'InfoOnly')  Extract only the
9   %                     basic information from the file. UffDataSets will be returned
10  %                     empty in this case.
11  %
12  %   [UffDataSets, Info, errmsg] = readuff(fileName) Extract the basic
13  %                     information from the file as well as the whole file
14  %                     contents - see the description for the UffDataSets
15  %                     below.
16  %
17  %   [UffDataSets, Info, errmsg] = readuff(fileName, recs) Extract only the
18  %                     records and their information requested by the recs
19  %                     array - the array of indices, e.g., recs=[1 3 10]. If
20  %                     empty, all the records are considered.
21  %
22  %   [UffDataSets, Info, errmsg] = readuff(..., recs, dsTypes) Extract only the
23  %                     records that meet the criterria of the dsTypes where
24  %                     dsTypes is an array of data-set types that are to be
25  %                     read - actually, this is another filter in addition to
26  %                     the recs one; e.g. dsTypes = [58 55]
27  %   [UffDataSets, Info, errmsg] = readuff(..., ..., ..., 0) Does not show
28  %                     any warning messages.
29  %
30  %   The ouput values are:
31  %   - UffDataSets:  an array of structures; each structure holds one data set
32  %                     (the data set between -1 and -1; Each structure,
33  %                     UffDataSets{i}, has the fields
34  %                         .dsType
35  %                         .binary
36  %                     and some additional field which are data-set dependant
37  %                     and are
38  %                     as follows:
39  %                     #58 - for measurement data - function at dof (58):
40  %                         .d1 (description 1)      .d2 (description 2)      .date
41  %                         .ID_4                    .ID_5                    .functionType (see notes)
42  %                         .loadCaseId              .measData                .refEntName
43  %                         .refDir                  .refNode                 .rspDir
44  %                         .rspEntName              .rspNode                 .x (time or frequency)
45  %                         .dx (abscissa spacing)    .abscUnitsLabel
46  %                         .ordinateNumUnitsLabel   .ordinateDenumUnitsLabel
47  %                         .zUnitsLabel             .zAxisValue
48  %                         .abscLengthUnitsExponent.abscForceUnitsExponent
49  %                         .abscTempUnitsExponent
50  %                         .abscAxisLabel           .ordinateLengthUnitsExponent
51  %                         .ordinateForceUnitsExponent                        .ordinateTempUnitsExponent
52  %                         .ordinateAxisLabel
53  %
54  %                     #58b - for measurement data - the same as 58 but the data
55  %                     is written in binary format
56  %
57  %                     #15 - coordinate data (15)  (Grid points):
58  %                         .nodeN                   .defCS                   .dispCS
59  %                         .color                   .x                       .y
60  %                         .z
61  %
62  %                     #18 - coordinate system data (18) :
63  %                         .csNum                   .csType                  .refCsNum
64  %                                                  .color                   .method    (=1)          .csName
65  %                                          .csX                    .csY                      .csZ
66  %                         .ref1X                   .ref1Y                   .ref1Z
67  %                                  .ref2X                   .ref2Y                    .ref2Z
68  %                         Method 1 defines the CS with three points: origin,
69  %                         point on +x axis, point on +xz plane
70  %
71  %                     #2411 - coordinate data (2411)  (Grid points):
72  %                         .nodeN                   .defCS                   .dispCS
73  %                         .color                   .x                       .y
74  %                         .z
75  %
76  %                     #2412 - element data (2412):
```

```
77  %                       .ElementLabel           .FEDescriptor           .PhysicalProp
78  %                       .MaterialProp           .ElementColour          .NumNodes
79  %                       .Elements
80  %
81  %                   #82 - display Sequence data (82):
82  %                       .traceNum               .nNodes                 .color
83  %                       .ID                     .lines
84  %
85  %                   #151 - header data (151):
86  %                       .modelName              .description            .dbApp
87  %                       .dateCreated            .timeCreated            .dbVersion
88  %                       .dbLastSaveDate         .dbLastSaveTime         .uffApp
89  %
90  %                   #164 - units (164):
91  %                       .unitsCode              .unitsDescription       .tempMode (1=absoulute, 2=relative)
92  %                       Unit factors for converting universal file units to SI. To convert from
93  %                       universal file units to SI divide by the appropriate factor listed below:
94  %                       .facLength              .facForce               .facTemp
95  %                       .facTempOffset
96  %
97  %                   #55 - data at nodes (55):
98  %                       /Common fields:/
99  %                       .analysisType           .dataCharacter = 1      .r1
100 %                       .dataType (2=real data, 5=complex data)
101 %                       .r2                     .r3                     .responseType
102 %                       .r4                     .r5                     .r6
103 %                       /Normal modes specific fields (analysisType = 2)/
104 %                       .modeNum                .modeFreq               .modeMass
105 %                       .mode_v_damping_ratio   .mode_h_damping_ratio
106 %                       /...or, for complex modes specific fields (analysisType = 3 or 7)/
107 %                       .modeNum                .eigVal                 .modalA
108 %                       .modalB
109 %                       /...or, for frequency response specific fields (analysisType = 5)/
110 %                       .freqNum                .freq
111 %
112 %   - Info:         (optional) structure with the following fields:
113 %                   .dsTypes    -   an array of data-set types read
114 %                   .binary     -   an array of 1s (binary format) and 0s (ascii format)
115 %                   .nDataSets  -   number of data sets found
116 %                   .errcode    -   an array of error codes for each data
117 %                                   set; 0 = no error otherwise an error occured in data
118 %                                   set read - see errmsg
119 %                   .errmsg     -   error messages (cell array of strings) for each
120 %                                   data set - empty if no error occured at specific data set
121 %                   .nErrors    -   number of errors found (unsupported
122 %                                   datasets, error reading data set,...)
123 %                   .errorMsgs  -   all the error messages (empty if no error is found)
124 %   - errmsg:       (optional) general (overall), file-based error
125 %                   messages - to enable reading of uncorrupted data for
126 %                   example,...
127 %
128 %
129 %   NOTES: r1..r6 are response vectors with node numbers in ROWS and
130 %   direction in COLUMN (r1=x, r2=y,...,r6=rz).
131 %
132 %   functionType can be one of the following:
133 %               0 -  General or Unknown
134 %               1 -  Time Response
135 %               2 -  (supported) Auto Spectrum
136 %               3 -  (supported) Cross Spectrum
137 %               4 -  (supported) Frequency Response Function
138 %               5 -  Transmissibility
139 %               6 -  (supported) Coherence
140 %               7 -  Auto Correlation
141 %               8 -  Cross Correlation
142 %               9 -  Power Spectral Density (PSD)
143 %               10 - Energy Spectral Density (ESD)
144 %               11 - Probability Density Function
145 %               12 - Spectrum
146 %               13 - Cumulative Frequency Distribution
147 %               14 - Peaks Valley
148 %               15 - Stress/Cycles
149 %               16 - Strain/Cycles
150 %               17 - Orbit
151 %               18 - Mode Indicator Function
152 %               19 - Force Pattern
153 %               20 - Partial Power
154 %               21 - Partial Coherence
155 %               22 - Eigenvalue
156 %               23 - Eigenvector
157 %               24 - Shock Response Spectrum
158 %               25 - Finite Impulse Response Filter
159 %               26 - Multiple Coherence
160 %               27 - Order Function
161 %
162 %   analysisType can be one of the following:
163 %               0: Unknown
164 %               1: Static
165 %               2: (supported) Normal Mode
166 %               3: (supported) Complex eigenvalue first order
167 %               4: Transient
168 %               5: (supported) Frequency Response
169 %               6: Buckling
170 %               7: (supported) Complex eigenvalue second order
171 %
172 %   dataCharacter can be one of the following:
```

```
%               0: Unknown
%               1: Scalar
%               2: 3 DOF Global Translation Vector
%               3: 6 DOF Global Translation & Rotation Vector
%               4: Symmetric Global Tensor
%
%   unitsCode can be one of the following:
%               1 - SI: Meter (newton)
%               2 - BG: Foot (pound f)
%               3 - MG: Meter (kilogram f)
%               4 - BA: Foot (poundal)
%               5 - MM: mm (milli newton)
%               6 - CM: cm (centi newton)
%               7 - IN: Inch (pound f)
%               8 - GM: mm (kilogram f)
%
%   functionType as well as other parameters are described in
%   Test_Universal_File_Formats.pdf
%
%   Examples:
%       [Data, Info, errmsg] = readuff('test.unv', 'InfoOnly');
%           Extracts only the information on the content of the test.unv.
%       [Data, Info, errmsg] = readuff('test.unv', [1 3 10]);
%           Reads the 1st, 3rd and 10th data-set from the test.unv, while
%       [Data, Info, errmsg] = readuff('test.unv', [1 3 10], [55, 58]);
%           Reads the 1st, 3rd and 10th data-set from the test.unv, but,
%           only those sets whose type is either 55 or 58.
%       [Data, Info, errmsg] = readuff('test.unv');
%           Reads the whole file content - all the data-sets.
%
%   See also: WRITEUFF
%
%   SOURCES:    [1] Bryce Gardner's read_uff obtained from the internet
%               [2] http://www.sdrl.uc.edu/uff/SDRChelp/LANG/English/unv_ug/book.htm
%
%
%   First release on 30.05.2004
%   (c) Primoz Cermelj, Slovenia
%   Contact: primoz.cermelj@gmail.com
%   Download location: http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=6395&
    objectType=file
%
%   Version: 1.3.0
%   Last revision: 11.2.2018
%
%   Contributors:
%   - Thomas Emmert
%   - Ben Cazzolato
%   - Ulrich Bittner
%   - Edward Hage
%
%   Bug reports, questions, contributions, etc. can be sent to the e-mail given above.
%--------------------------------------------------------------------------

%----------------
%READUFF history
%----------------
%
% [v.1.3.0] 11.2.2018
%- FIX: fixes (Thomas Emmert)
% [v.1.2.0] 14.11.2015
%- NEW: dataset 18 added (Ulrich Bittner)
% [1.1.2] 24.11.2014
%- FIX: fixed a bug: dataType field not explicitly described in the description
%       (thanks to Edward Hage)
% [1.1.1] 27.05.2013
%- FIX: fixed data reading in extract58: extra zero values not read anymore
% [1.1.0] 27.02.2013
%- FIX: data-set 151 fields unified (readuff and writeuff)
% [1.0.9] 18.02.2013
%- FIX: readuff(fileName, recs) reading mode works now
% [1.0.6-1.0.8] 30.08.2010
%- FIX: in thecase of inconsistency in 58b data sets (extra bytes of data
%       present), extra bytes at the end willbe skipped.
%- FIX: regular expressions are now used to detect data-set blocks - more
%       robust
%- FIX: more robust start/end block detection
%- FIX: minor changes
% [v.1.0.1-1.0.5] 12.03.2008-08.11.2008
%- NEW: some restrictions relaxed when reading the data-set 58
%- NEW: some additional checking for badly formatted data-set id line
%- NEW: some additional warning/error messages displayed for the 58b set
%- FIX: uint8=>char instead of char=>char in fread fixes a problem on
%       some Linux systems
%- FIX: minor bug removed (related to the extracted abscissa values for
%       the 58, complex case of data)
% [v.1.0.0] 10.03.2008
%- NEW: datasets 2411 and 2412 added (Ben Cazzolato)
% [v.0.9.9b1-5] 08.01.2008
%- NEW: additional checking when reading 58b data
%- NEW: additional filter added - dsTypes
%- FIX: minor bug removed concerning the finding of the "    -1" tags
%- FIX: previously, when reading data-set 58b, some data-sets were
%       skipped; this bug is now removed
%- NEW: new functionality to read only a portion of file and to extract
%       the information only
```

```
268 %[v.0.9.7-v.0.9.8b7] 28.02.2006
269 %- FIX: a bug reading even abscissa data from the 58 set removed
270 %- NEW: uneven abscissa data-reading is now supported
271 %- FIX: removing leading and trailing spaces from the strings read
272 %- NEW: hybrid binary-58 format (58b) is now supported
273 %- NEW: binary field was added to UffDataSets structures
274 %[v.0.9.7] 24.05.2005
275 %- NEW: dsType field was added to UffDataSets structures
276 %[v.0.9.6b4] 11.05.2005
277 %- FIX: Matlab version down to 5.3 is now supported
278 %- FIX: Some minor bugs removed
279 %- NEW: Speed improvement; reading is much faster now
280 %
281 %----------------
282 global show_warning
283 error(nargchk(1, 4, nargin));
284
285 %--------------
286 %Default outputs
287 %--------------
288 show_warning = 1;
289 UffDataSets = [];
290 Info.errcode = [];
291 Info.nDataSets = 0;
292 Info.dsTypes = [];
293 Info.binary = [];
294 Info.errmsg = [];
295 Info.nErrors = 0;
296 errmsg = [];
297
298
299 %--------------
300 %Handle input parameters
301 %--------------
302 recs = [];
303 dsTypes = [];
304 fileName = varargin{1};
305 readMode = 1;    %0=info only, 1=read all, 2=read filtered data-sets
306 if nargin > 1
307     if isnumeric(varargin{2}) || isempty(varargin{2})
308         recs = varargin{2};
309         readMode = 2;
310     elseif strcmpi(varargin{2}, 'infoonly')
311         readMode = 0;
312     else
313         error('Unknown request in the second parameter');
314     end
315 end
316 if nargin > 2
317     if isnumeric(varargin{3}) || isempty(varargin{3})
318         dsTypes = varargin{3};
319         readMode = 2;
320     else
321         error('Unknown request in the third parameter');
322     end
323 end
324 if nargin > 3
325     if isnumeric(varargin{4}) && varargin{4} == 0
326         show_warning = 0;
327     end
328 end
329
330
331 %--------------
332 %Some variables
333 %--------------
334 errN = 0;                %current global error number (data-set number independent)
335
336
337 %--------------
338 %Read the whole file data into an array of characters
339 %--------------
340 try
341     fid = fopen(fileName, 'r');
342     if fid == -1,
343         errN = errN + 1;
344         errmsg{errN,1} = ['could not open file: ' fileName];
345         disp(errmsg{errN});
346         return
347     end
348     FILE_DATA = (fread(fid, 'uint8=>char')).';
349 catch
350     errN = errN + 1;
351     errmsg{errN,1} = ['error reading file contents: ' lasterr];
352     disp(errmsg{errN});
353     %Close the file
354     fclose(fid);
355     return
356 end
357 %Close the file
358 err = fclose(fid);
359 if err == -1
360     errN = errN + 1;
361     errmsg{errN,1} = 'error while closing file';
362     disp(errmsg{errN});
363 end
```

```
364
365
366  %--------------
367  % Find all valid blocks, data between -1 and -1; pointers to blocks of
368  % data; include the first -1 but exclude the last -1;
369  % the first -1 will be skipped further later on in get_block_prop
370  %--------------
371  % ind = strfind(FILE_DATA, '    -1
                                                                                ');
372  ind = regexpi(FILE_DATA, '(((?<=^|[\r\n])(    -1 *))|((    -1)( ){74}))(?:$|[\r\n])');
373  data_len = length(FILE_DATA);
374  for ii=length(ind):-1:1
375      if ind(ii) == data_len
376          continue
377      end
378      if ~isspace(FILE_DATA(ind(ii)+6))
379          ind(ii) = [];
380      end
381  end
382  nBlocks = floor(length(ind)/2);
383  if nBlocks < 1
384      errN = errN + 1;
385      errmsg{errN,1} = 'No valid blocks found';
386      disp(errmsg{errN});
387      return
388  elseif rem(length(ind), 2)
389      errN = errN + 1;
390      errmsg{errN,1} = 'Uneven (odd) -1 tags found (one -1 tag too many). Check your file.';
391      disp(errmsg{errN});
392      return
393  end
394  blocks = zeros(nBlocks, 2);
395  blocks(:,1) = ind(1:2:2*nBlocks)';
396  blocks(:,2) = ind(2:2:2*nBlocks)'-1;
397
398
399  %============================
400  % MAIN FILE LOOP - go through all the blocks and extract data from each
401  % block according to the data type
402  %============================
403  dataSetN = 0;        % counts VALID data-sets (including non-supported ones)
404  if isempty(recs)
405      recs = 1:nBlocks;
406  end
407  try
408      if readMode==2
409          readScope = recs;
410          if max(recs) > nBlocks
411              error('Max block number to be read is too high (%d)', max(recs));
412          end
413      else
414          readScope = 1:nBlocks;
415      end
416
417      for ii=readScope
418
419          % Skips the first  -1, detects the data-set type and any possible
420          % properties (e.g., for 58b there are some additional fields in the data-set
421          % id record), and also returns blockLines - pointers to start and
422          % end offsets of lines of the data-set-block data
423          [data_set_type, DataSetProp, blockLines, errMessage] = ...
424                  get_block_prop(ii, blocks(ii,1), blocks(ii,2), FILE_DATA);
425          if ~isempty(errMessage)
426              errN = errN + 1;
427              errmsg{errN,1} = errMessage;
428              continue
429          end
430
431          if readMode~=0
432              % First check if dataSetN meets the filter
433              if ~isempty(dsTypes)
434                  if isempty(find(dsTypes==data_set_type))
435                      continue
436                  end
437              end
438
439              dataSetN = dataSetN + 1;
440              ds_errmsg = [];
441
442              % Now, read the record
443              if data_set_type == 58       % Function at nodal dof
444                  [ds_data,ds_errmsg] = extract58(fileName, FILE_DATA, blockLines, DataSetProp, ii);
445              elseif data_set_type == 15   % Coordinate data
446                  [ds_data,ds_errmsg] = extract15(FILE_DATA, blockLines);
447              elseif data_set_type == 18   % Coordinate System Definition
448                  [ds_data,ds_errmsg] = extract18(FILE_DATA, blockLines);
449              elseif data_set_type == 2411   % Node Coordinate data
450                  [ds_data,ds_errmsg] = extract2411(FILE_DATA, blockLines);
451              elseif data_set_type == 2412   % Element data
452                  [ds_data,ds_errmsg] = extract2412(FILE_DATA, blockLines);
453              elseif data_set_type == 151  % Header data
454                  [ds_data,ds_errmsg] = extract151(FILE_DATA, blockLines);
455              elseif data_set_type == 164  % Units data
456                  [ds_data,ds_errmsg] = extract164(FILE_DATA, blockLines);
457              elseif data_set_type == 82   % Display sequence data
458                  [ds_data,ds_errmsg] = extract82(FILE_DATA, blockLines);
```

```
459             elseif data_set_type == 55  % Modal data file
460                 [ds_data,ds_errmsg] = extract55(FILE_DATA, blockLines);
461             else
462                 ds_data = [];
463                 ds_errmsg = ['unknown data-set (' num2str(data_set_type)  ') found in ' num2str(ii) '-th
                        data-set '];
464             end
465             UffDataSets{dataSetN} = ds_data;
466             UffDataSets{dataSetN}.dsType = data_set_type;
467             UffDataSets{dataSetN}.binary = DataSetProp.binary;
468         end
469
470         Info.errmsg{dataSetN} = ds_errmsg;
471         Info.dsTypes(dataSetN) = data_set_type;
472         Info.binary(dataSetN) = DataSetProp.binary;
473         if isempty(ds_errmsg)
474             Info.errcode(dataSetN) = 0;
475         else
476             Info.errcode(dataSetN) = 1;
477         end
478     end
479
480 catch
481     errN = errN + 1;
482     errmsg{errN,1} = lasterr;
483 end
484 %=================================
485 %END OF MAIN FILE LOOP
486 %=================================
487
488
489 Info.nErrors = length(find(Info.errcode));
490 Info.nDataSets = dataSetN;
491 Info.errorMsgs = Info.errmsg(find(Info.errcode));
492
493 if ~isempty(errmsg)
494     for ii=1:length(errmsg)
495         disp(errmsg{ii});
496     end
497 end
498
499
500
501
502 %=========================================================================
503 %                          SUBFUNCTIONS SECTION
504 %=========================================================================
505
506
507
508 %-------------------------------------------------------------------------
509 function [dataSet, DataSetProp, blockLines, errMessage] = get_block_prop(ds_num, so, eo, FILE_DATA)
510 % Extract block-data lines' pointers (start and end for each line) and also
511 % returns the data-set number identified along with any additional
512 % parameters such as in the case of 58b data-set. so points to the first -1 tag
513 % (designated by o): o___-1 while eo points to the end -1 tag: o___-1.
514 % blockLInes are start and end offsets of each line in the current data-set
515 % starting from the line right after the data-set id line.
516 % Empty lines are skipped.
517
518 % Scans for block data and returns lines' pointers (start and end offsets
519 % in a 2-column matrix).
520
521 dataSet = [];
522 DataSetProp = [];
523 blockLines = [];
524 errMessage = [];
525 try
526     % For a two-column matrix of start and end indices designating the
527     % start and end for each line of the data set
528     blockData = FILE_DATA(so:eo);
529     dataLen = length(blockData);
530     lineBreaksIndn = strfind(blockData,sprintf('\n'));
531     lineBreaksIndrn = strfind(blockData,sprintf('\r\n'));
532     diffn = setdiff(lineBreaksIndn-1,lineBreaksIndrn);
533
534     %% Determine Linefeed character
535     if (length(lineBreaksIndrn)>length(diffn)) % windows linefeeds '\r\n'
536         lengthLF = 2;
537         lineBreaksInd = lineBreaksIndrn;
538     else % unix linefeeds '\n'
539         lengthLF = 1;
540         lineBreaksInd = lineBreaksIndn;
541     end
542     %% Determine start index of Blocklines
543     if lineBreaksInd(1)>1
544         % Dataset does not start with a newline '\r\n'
545         fromIdx = [1; lineBreaksInd'+lengthLF];
546     else
547         % Dataset does start with a newline '\r\n'
548         fromIdx = lineBreaksInd'+lengthLF;
549     end
550     %% Determine end index of Blocklines
551     if fromIdx(end) < dataLen
552         % Dataset does not end with a newline '\r\n'
553         toIdx = [fromIdx(2:end)-lengthLF-1; dataLen];
```

```
554       else
555           % Dataset does end with a newline '\r\n'
556           toIdx = fromIdx(2:end)-lengthLF-1;
557           fromIdx(end) = [];
558       end
559
560       blockLines = [fromIdx toIdx];
561
562       % The data-set line; get the data-set number
563       dataSetLine = blockData(blockLines(2,1):blockLines(2,2));
564       if isempty(dataSetLine) || length(dataSetLine) < 6
565           warning('Badly formatted data-set id for data-set # %d', ds_num);
566           dataSet = sscanf(dataSetLine, '%i', 1);
567       else
568           dataSet = sscanf(dataSetLine(1:6), '%i', 1);
569       end
570       if isempty(dataSet)
571           errMessage = 'no valid data-set type found';
572           return
573       end
574
575       % Get the format
576       if length(dataSetLine) < 7
577           format = '';
578       else
579           format = sscanf(dataSetLine(7), '%c', 1);
580       end
581       if strcmpi(format, 'b')
582           DataSetProp.binary = 1;
583           DataSetProp.byteOrdering = sscanf(dataSetLine(8:13),'%i',1);
584           DataSetProp.fpFormat = sscanf(dataSetLine(14:19),'%i',1);
585           DataSetProp.nAsciiLines = sscanf(dataSetLine(20:31),'%i',1);
586           DataSetProp.nBytes = sscanf(dataSetLine(32:43),'%i',1);
587           DataSetProp.d1 = sscanf(dataSetLine(44:49),'%i',1);
588           DataSetProp.d2 = sscanf(dataSetLine(50:55),'%i',1);
589           DataSetProp.d3 = sscanf(dataSetLine(56:67),'%i',1);
590           DataSetProp.d4 = sscanf(dataSetLine(68:end),'%i',1);
591       else
592           DataSetProp.binary = 0;
593       end
594
595       % Global blockLines (with respect to FILE_DATA)
596       blockLines = blockLines(3:end,:) + so - 1;
597       if size(blockLines,1) < 2
598           errMessage = 'empty data block found';
599           return
600       end
601
602   catch
603       errMessage = ['error while reading the header info at data set #: ' num2str(ds_num) ' (' lasterr ')'];
604       return
605   end
606
607
608
609
610   %--------------------------------------------------------------------------
611   function [UFF, errMessage] = extract58(fileName, DATA, blockLines, DataSetProp, setn)
612   % #58 - Extract data-set type 58 data
613   global show_warning
614
615   UFF = [];
616   UFF.measData = [];
617   errMessage = [];
618   lineN = 1;
619   nLines = size(blockLines, 1);
620
621   try
622       % Line 1
623       UFF.d1 = strim(sscanf(DATA(blockLines(1,1):blockLines(1,2)), '%c', 80));
624       lineN = lineN + 1;
625       % Line 2
626       UFF.d2 = strim(sscanf(DATA(blockLines(2,1):blockLines(2,2)), '%c', 80));
627       lineN = lineN + 1;
628       % Line 3
629       UFF.date = strim(sscanf(DATA(blockLines(3,1):blockLines(3,2)), '%c', 80));
630       lineN = lineN + 1;
631       % Line 4
632       UFF.ID_4 = strim(sscanf(DATA(blockLines(4,1):blockLines(4,2)), '%c', 80));
633       lineN = lineN + 1;
634       % Line 5
635       UFF.ID_5 = strim(sscanf(DATA(blockLines(5,1):blockLines(5,2)), '%c', 80));
636       lineN = lineN + 1;
637       % Line 6
638       tmpLine = DATA(blockLines(6,1):blockLines(6,2));
639       tmpLine = [tmpLine repmat(' ', 1, 80-length(tmpLine))];
640       UFF.functionType = sscanf(tmpLine(1:5), '%i', 1);
641       tmp = sscanf(tmpLine(6:15),'%i', 1);
642       tmp = sscanf(tmpLine(16:20),'%i', 1);
643       UFF.loadCaseId = sscanf(tmpLine(21:30), '%i', 1);
644       UFF.rspEntName = sscanf(tmpLine(32:41), '%c');
645       UFF.rspNode = sscanf(tmpLine(42:51), '%i', 1);
646       UFF.rspDir = sscanf(tmpLine(52:55), '%i', 1);
647       UFF.refEntName = sscanf(tmpLine(57:66), '%c');
648       UFF.refNode = sscanf(tmpLine(67:76),'%i', 1);
649       UFF.refDir = sscanf(tmpLine(77:80),'%i', 1);
```

```
650       lineN = lineN + 1;
651
652       % Line 7; data form
653       tmpLine = DATA(blockLines(7,1):blockLines(7,2));
654       tmpLine = [tmpLine repmat(' ', 1, 80-length(tmpLine))];
655       ordDataType = sscanf(tmpLine(1:10), '%i', 1);
656       numpt = sscanf(tmpLine(11:20),'%i', 1);  %# of points if even spacing or # of pairs if uneven spacing
657       spacingType = sscanf(tmpLine(21:30), '%i', 1);
658       UFF.xmin = sscanf(tmpLine(31:43), '%g', 1);
659       UFF.dx = sscanf(tmpLine(44:56), '%g', 1);
660       UFF.zAxisValue = sscanf(tmpLine(57:69), '%g', 1);
661       complexOrd = (ordDataType == 5 | ordDataType == 6);
662       if (ordDataType == 2)||(ordDataType == 5)
663           UFF.precision = 'single';
664       else
665           UFF.precision = 'double';
666       end
667       lineN = lineN + 1;
668
669       % Line 8; abscissa data characteristics
670       tmpLine = DATA(blockLines(8,1):blockLines(8,2));
671       tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
672       UFF.abscDataChar = sscanf(tmpLine(1:10),'%i',1);
673       UFF.abscLengthUnitsExponent = sscanf(tmpLine(11:15),'%i',1);
674       UFF.abscForceUnitsExponent = sscanf(tmpLine(16:20),'%i',1);
675       UFF.abscTempUnitsExponent = sscanf(tmpLine(21:25),'%i',1);
676       UFF.abscAxisLabel = sscanf(tmpLine(27:46),'%c');
677       UFF.abscUnitsLabel = sscanf(tmpLine(48:end),'%c');
678       lineN = lineN + 1;
679
680       % Line 9; Ordinate (or ordinate numerator) Data Characteristics
681       tmpLine = DATA(blockLines(9,1):blockLines(9,2));
682       tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
683       UFF.ordDataChar = sscanf(tmpLine(1:10),'%i',1);
684       UFF.ordinateLengthUnitsExponent = sscanf(tmpLine(11:15),'%i',1);
685       UFF.ordinateForceUnitsExponent = sscanf(tmpLine(16:20),'%i',1);
686       UFF.ordinateTempUnitsExponent = sscanf(tmpLine(21:25),'%i',1);
687       UFF.ordinateAxisLabel = sscanf(tmpLine(27:46),'%c');
688       UFF.ordinateNumUnitsLabel = sscanf(tmpLine(48:end),'%c');
689       lineN = lineN + 1;
690
691       % Line 10; Ordinate Denominator Data Characteristics
692       tmpLine = DATA(blockLines(10,1):blockLines(10,2));
693       tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
694       ordDenominatorDataType = sscanf(tmpLine(1:10),'%i',1);
695       tmp = sscanf(tmpLine(11:15),'%i',1);
696       tmp = sscanf(tmpLine(16:20),'%i',1);
697       tmp = sscanf(tmpLine(21:25),'%i',1);
698       temp = sscanf(tmpLine(27:46),'%c');
699       UFF.ordinateDenumUnitsLabel = sscanf(tmpLine(48:end),'%c');
700       lineN = lineN + 1;
701
702       % Line 11; Z-axis Data Characteristics
703       tmpLine = DATA(blockLines(11,1):blockLines(11,2));
704       tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
705       tmp = sscanf(tmpLine(1:10),'%i',1);
706       tmp = sscanf(tmpLine(11:15),'%i',1);
707       tmp = sscanf(tmpLine(16:20),'%i',1);
708       tmp = sscanf(tmpLine(21:25),'%i',1);
709       temp = sscanf(tmpLine(27:46),'%c');
710       UFF.zUnitsLabel = sscanf(tmpLine(48:end),'%c');
711       lineN = lineN + 1;
712
713       % Line 12 ...; Data Values
714       if DataSetProp.binary
715           %-------
716           % BINARY
717           %-------
718           if DataSetProp.byteOrdering == 1; format = 'l'; else format = 'b'; end;
719           if (ordDataType==2 || ordDataType==5)
720               prec = 'single';    % single precision
721               numLen = 4;
722           else
723               prec = 'double';     % double precision
724               numLen = 8;
725           end
726           fid = fopen(fileName,'r',format);
727           if fid == -1
728               errMessage = ['could not reopen file for binary data reading: ' fileName];
729               return
730           end
731
732           % It was observed that some programs write some inconsistent values
733           % to the header concerning the number of data and/or bytes the data
734           % is to occupy. In case such incosistency is found, the max number
735           % will be used and a warning displayed.
736
737           % According to the UFF documentation, in the case of uneven
738           % abscissa, the abscissa is always stored as real, single
739           % precision.
740           if spacingType == 0   % uneven
741               n_ord_vals_to_read = (DataSetProp.nBytes - numpt*4)/numLen;
742           else                  % even
743               n_ord_vals_to_read = DataSetProp.nBytes/numLen;
744           end
745           n_ord_vals_to_read = n_ord_vals_to_read/(1+complexOrd);
```

```
746
747         n_act_bytes = blockLines(end, 2) - blockLines(12,1) + 1;
748         skipbytes = n_act_bytes - DataSetProp.nBytes;
749
750         if skipbytes < 0
751             errMessage = ['Badly formated binary uff file (' fileName '): '...
752                           'not enough bytes of data according to the bytes '...
753                           'specified in the header of the set #' num2str(setn)];
754             return
755         end
756         if numpt ~= n_ord_vals_to_read && show_warning
757             warning(['Badly formated binary uff file (%s) at set #%d: the number of bytes '...
758                      'specified does not match the specified number of '...
759                      'values; only the data corresponding to the number '...
760                      'of bytes will be read.'], fileName, setn);
761         end
762         if skipbytes > 0 && show_warning
763             warning(['Badly formated binary uff file (%s) at set #%d: the size of the data '...
764                      'is greater than the number of bytes specified; extra '...
765                      'extra bytes at the end will be skipped.'], fileName, setn);
766         end
767
768 %         status = fseek(fid, blockLines(12,1)-1+skipbytes, 'bof');
769         status = fseek(fid, blockLines(12,1)-1, 'bof');
770         if status
771             errMessage = ['could not start reading binary data from ' fileName...
772                           ' at set #' num2str(setn)];
773             return
774         end
775
776         n_ord_vals_to_read = max(numpt, n_ord_vals_to_read)*(1+complexOrd);
777
778         try
779             dimData = (1+complexOrd);
780             if spacingType == 0 %uneven spacing
781                 absc_values = fread(fid, numpt, 'float32', numLen*(1+complexOrd*1));
782                 fseek(fid, blockLines(12,1)-1+4+skipbytes, 'bof');
783             end
784             measData = fread(fid, [dimData,n_ord_vals_to_read/dimData], [num2str(dimData),'*',prec], not(
                    spacingType)*4);
785             if complexOrd
786                 measData(1,:) = measData(1,:)+1j*measData(2,:);
787                 measData(2,:) = [];
788             end
789         catch
790             errMessage = ['error while reading binary data from ' fileName];
791             return
792         end
793         fclose(fid);
794     else
795         %------
796         % ASCII
797         %------
798         dimData = (1+complexOrd);
799         values = sscanf(DATA(blockLines(12,1):blockLines(end,2)),'%g');
800         % Split time/frequency vector from data values
801         if spacingType == 0 %uneven spacing
802             absc_values = values(1:(dimData+1):end); %TODO: Check
803             values(1:(dimData+1):end) = [];
804         end
805
806         measData = reshape(values,dimData,length(values)/dimData);
807         if complexOrd
808             measData(1,:) = measData(1,:)+1j*measData(2,:);
809             measData(2,:) = [];
810         end
811     end
812     if not(spacingType == 0) % if even spacing create abscissa values
813         nVal = length(measData);
814         absc_values = UFF.xmin : UFF.dx : UFF.xmin + (nVal-1)*UFF.dx;
815     end
816     UFF.x = absc_values;
817     UFF.measData = measData;
818
819 catch
820     errMessage = ['error reading measurement data: ' lasterr];
821     return
822 end
823
824
825 %-------------------------------------------------------------------------
826 function [UFF,errMessage] = extract151(DATA,blockLines)
827 % #151 - Extract data-set type 151 data
828
829 UFF = [];
830 errMessage = [];
831 lineN = 1;
832 nLines = size(blockLines,1);
833
834 try
835     % Line 1
836     UFF.modelName = strim(sscanf(DATA(blockLines(1,1):blockLines(1,2)),'%c',80));
837     lineN = lineN + 1;
838     % Line 2
839     UFF.description = strim(sscanf(DATA(blockLines(2,1):blockLines(2,2)),'%c',80));
840     lineN = lineN + 1;
```

```
841         % Line 3
842         UFF.dbApp = strim(sscanf(DATA(blockLines(3,1):blockLines(3,2)),'%c',80));
843         lineN = lineN + 1;
844         % Line 4
845         tmpLine = DATA(blockLines(4,1):blockLines(4,2));
846         tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
847         UFF.dateCreated = sscanf(tmpLine(1:10),'%c');
848         UFF.timeCreated = sscanf(tmpLine(11:20),'%c');
849         UFF.dbVersion = sscanf(tmpLine(21:30),'%i',10);
850         lineN = lineN + 1;
851         % Line 5
852         tmpLine = DATA(blockLines(5,1):blockLines(5,2));
853         tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
854         UFF.dateSaved = sscanf(tmpLine(1:10),'%c');
855         UFF.timeSaved = sscanf(tmpLine(11:20),'%c');
856         lineN = lineN + 1;
857         % Line 6
858         UFF.uffApp = strim(sscanf(DATA(blockLines(6,1):blockLines(6,2)),'%c',80));
859     catch
860         errMessage = ['error reading header data at line ' num2str(lineN) ' relatively to current data-set'];
861         return
862     end
863
864
865     %-------------------------------------------------------------------------
866     function [UFF,errMessage] = extract164(DATA,blockLines)
867     % #164 - Extract data-set type 164 data
868     UFF = [];
869     errMessage = [];
870     lineN = 1;
871     nLines = size(blockLines,1);
872     try
873         % Line 1
874         tmpLine = DATA(blockLines(1,1):blockLines(1,2));
875         tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
876         UFF.unitsCode = sscanf(tmpLine(1:10),'%i');
877         UFF.unitsDescription = strim(sscanf(tmpLine(11:31),'%c'));
878         UFF.tempMode = sscanf(tmpLine(32:41),'%i');
879         lineN = lineN + 1;
880         % Line 2
881         tmpLine = DATA(blockLines(2,1):blockLines(2,2));
882         tmpLine = lower([tmpLine repmat(' ',1,80-length(tmpLine))]);
883         tmpLine = strrep(tmpLine,'d-','e-');
884         tmpLine = strrep(tmpLine,'d+','e+');
885         UFF.facLength = sscanf(tmpLine(1:25),'%f');
886         UFF.facForce = sscanf(tmpLine(26:50),'%f');
887         UFF.facTemp = sscanf(tmpLine(51:75),'%f');
888         lineN = lineN + 1;
889         % Line 3
890         tmpLine = DATA(blockLines(3,1):blockLines(3,2));
891         tmpLine = lower([tmpLine repmat(' ',1,80-length(tmpLine))]);
892         tmpLine = strrep(tmpLine,'d-','e-');
893         tmpLine = strrep(tmpLine,'d+','e+');
894         UFF.facTempOffset = sscanf(tmpLine(1:25),'%f');
895     catch
896         errMessage = ['error reading units data at line' num2str(lineN) ' relatively to current data-set: '
                lasterr];
897         return
898     end
899
900
901
902     %-------------------------------------------------------------------------
903     function [UFF,errMessage] = extract15(DATA, blockLines)
904     % #15 - Extract data-set type 15 data
905
906     UFF = [];
907     errMessage = [];
908     nLines = size(blockLines,1);
909
910     try
911         values = sscanf(DATA(blockLines(1,1):blockLines(end,2)),'%g');
912         nVals = length(values);
913         nNodes = round(nVals/7);
914         values = reshape(values,7,nNodes).';
915         %
916         UFF.nodeN = round(values(:,1));
917         UFF.defCS = round(values(:,2));
918         UFF.dispCS = round(values(:,3));
919         UFF.color = round(values(:,4));
920         UFF.x = values(:,5);
921         UFF.y = values(:,6);
922         UFF.z = values(:,7);
923     catch
924         errMessage = ['error reading coordinate data: ' lasterr];
925         return
926     end
927
928
929     %-------------------------------------------------------------------------
930     function [UFF, errMessage] = extract18(DATA, blockLines)
931     % #18 - Extract Coordinate System data-set type 18 data
932
933     UFF = [];
934     errMessage = [];
935     lineN =1;
```

```
936  nNodes = round(size(blockLines,1)/4);
937  try
938      for nodeNum = 1:nNodes
939          lineN = 1+4*(nodeNum-1);
940          % Line 1
941  %         tmpLine = DATA(blockLines(lineN,1):blockLines(lineN,2));
942  %         tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
943  %         UFF.csNum(nodeNum) = sscanf(tmpLine(1:10),'%i',1);
944  %         UFF.csType(nodeNum) = sscanf(tmpLine(11:20),'%i',1);
945  %         UFF.refCsNum(nodeNum) = sscanf(tmpLine(21:30),'%i',1);
946  %         UFF.color(nodeNum) = sscanf(tmpLine(31:40),'%i',1);
947  %         UFF.method(nodeNum) = sscanf(tmpLine(41:50),'%i',1);
948          %test for better method
949          values = sscanf(DATA(blockLines(lineN,1):blockLines(lineN,2)),'%g');
950          UFF.csNum(nodeNum) =  round(values(1));
951          UFF.csType(nodeNum) =  round(values(2));
952          UFF.refCsNum(nodeNum) =  round(values(3));
953          UFF.color(nodeNum) =  round(values(4));
954          UFF.method(nodeNum) =  round(values(5));
955          lineN = lineN + 1;
956          % Line 2
957          UFF.csName(nodeNum) = cellstr(strim(sscanf(DATA(blockLines(lineN,1):blockLines(lineN,2)),'%c')));
958          lineN = lineN + 1;
959          % Line 3
960  %         tmpLine = DATA(blockLines(lineN,1):blockLines(lineN,2));
961  %         tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
962  %         UFF.csX(nodeNum) = sscanf(tmpLine(1:14),'%g',1);
963  %         UFF.csY(nodeNum) = sscanf(tmpLine(15:27),'%g',1);
964  %         UFF.csZ(nodeNum) = sscanf(tmpLine(28:40),'%g',1);
965  %         UFF.ref1X(nodeNum) = sscanf(tmpLine(41:53),'%g',1);
966  %         UFF.ref1Y(nodeNum) = sscanf(tmpLine(54:66),'%g',1);
967  %         UFF.ref1Z(nodeNum) = sscanf(tmpLine(67:79),'%g',1);
968          values = sscanf(DATA(blockLines(lineN,1):blockLines(lineN,2)),'%g');
969          UFF.csX(nodeNum) = values(1);
970          UFF.csY(nodeNum) = values(2);
971          UFF.csZ(nodeNum) = values(3);
972          UFF.ref1X(nodeNum) = values(4);
973          UFF.ref1Y(nodeNum) = values(5);
974          UFF.ref1Z(nodeNum) = values(6);
975          lineN = lineN + 1;
976          % Line 4
977  %         tmpLine = DATA(blockLines(lineN,1):blockLines(lineN,2));
978  %         tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
979  %         UFF.ref2X(nodeNum) = sscanf(tmpLine(1:14),'%g',1);
980  %         UFF.ref2Y(nodeNum) = sscanf(tmpLine(15:27),'%g',1);
981  %         UFF.ref2Z(nodeNum) = sscanf(tmpLine(28:40),'%g',1);
982          values = sscanf(DATA(blockLines(lineN,1):blockLines(lineN,2)),'%g');
983          UFF.ref2X(nodeNum) = values(1);
984          UFF.ref2Y(nodeNum) = values(2);
985          UFF.ref2Z(nodeNum) = values(3);
986      end
987  catch
988      errMessage = ['error reading trace-line data at line' num2str(lineN) ' relatively to current data-set:
         ' lasterr];
989      return
990  end
991
992  %--------------------------------------------------------------------------
993  function [UFF,errMessage] = extract82(DATA,blockLines)
994  %#82 - Extract display sequence data-set type 82 data
995
996  UFF = [];
997  errMessage = [];
998  lineN = 1;
999  nLines = size(blockLines,1);
1000 try
1001     % Line 1
1002     tmpLine = DATA(blockLines(1,1):blockLines(1,2));
1003     tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
1004     UFF.traceNum = sscanf(tmpLine(1:10),'%i',1);
1005     UFF.nNodes = sscanf(tmpLine(11:20),'%i',1);
1006     UFF.color = sscanf(tmpLine(21:30),'%i',1);
1007     lineN = lineN + 1;
1008     % Line 2
1009     UFF.ID = strim(sscanf(DATA(blockLines(2,1):blockLines(2,2)),'%c'));
1010     lineN = lineN + 1;
1011     % Line 3
1012     UFF.lines = sscanf(DATA(blockLines(3,1):blockLines(end,2)),'%g');
1013 catch
1014     errMessage = ['error reading trace-line data at line' num2str(lineN) ' relatively to current data-set:
         ' lasterr];
1015     return
1016
1017 end
1018
1019
1020
1021
1022 %--------------------------------------------------------------------------
1023 function [UFF,errMessage] = extract55(DATA,blockLines)
1024 %#55 - Extract modal data-set type 55 data
1025
1026 UFF = [];
1027 errMessage = [];
1028 lineN = 1;
1029 nLines = size(blockLines,1);
```

```
1030  errN = 0;
1031
1032  try
1033      %Line 1
1034      UFF.d1 = strim(sscanf(DATA(blockLines(1,1):blockLines(1,2)),'%c',80));
1035      lineN = lineN + 1;
1036      %Line 2
1037      UFF.d2 = strim(sscanf(DATA(blockLines(2,1):blockLines(2,2)),'%c',80));
1038      lineN = lineN + 1;
1039      %Line 3
1040      UFF.date = strim(sscanf(DATA(blockLines(3,1):blockLines(3,2)),'%c',80));
1041      lineN = lineN + 1;
1042      %Line 4
1043      UFF.IDs = strim(sscanf(DATA(blockLines(4,1):blockLines(4,2)),'%c',80));
1044      %Line 5
1045      temp = sscanf(DATA(blockLines(5,1):blockLines(5,2)),'%c');
1046      lineN = lineN + 1;
1047      %Line 6
1048      tmpLine = DATA(blockLines(6,1):blockLines(6,2));
1049      tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
1050      UFF.modelType = sscanf(tmpLine(1:10),'%i',1);
1051      lineN = lineN + 1;
1052      if UFF.modelType ~=1,
1053          errMessage = ['not structural model type (line: ' num2str(lineN) ' relatively to current data-set)'
                ];
1054          return
1055      end
1056      UFF.analysisType = sscanf(tmpLine(11:20),'%i',1);
1057      UFF.dataCharacter = sscanf(tmpLine(21:30),'%i',1);
1058      UFF.responseType = sscanf(tmpLine(31:40),'%i',1);
1059      UFF.dataType = sscanf(tmpLine(41:50),'%i',1);
1060      num_data_per_pt = sscanf(tmpLine(51:60),'%c');
1061
1062      %Read records 7 and 8 which are analysis-type dependent
1063      if UFF.analysisType == 2   %Normal Mode
1064          %Line 7
1065          tmpLine = DATA(blockLines(7,1):blockLines(7,2));
1066          tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
1067          two = sscanf(tmpLine(1:10),'%i',1);
1068          lineN = lineN + 1;
1069          if two ~= 2,
1070              errMessage = ['unexpected value at line ' num2str(lineN) ' relatively to current data-set'];
1071              return
1072          end
1073          four = sscanf(tmpLine(11:20),'%i',1);
1074          if four ~= 4,
1075              errMessage = ['unexpected value at line: ' num2str(lineN) ' relatively to current data-set'];
1076              return
1077          end
1078          tmp = sscanf(tmpLine(21:30),'%i',1);
1079          UFF.modeNum = sscanf(tmpLine(31:40),'%i',1);
1080          %Line 8
1081          tmpLine = DATA(blockLines(8,1):blockLines(8,2));
1082          tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
1083          lineN = lineN + 1;
1084          UFF.modeFreq = sscanf(tmpLine(1:13),'%g',1);
1085          UFF.modeMass = sscanf(tmpLine(14:26),'%g',1);
1086          UFF.mode_v_damping_ratio = sscanf(tmpLine(27:39),'%g',1);
1087          UFF.mode_h_damping_ratio = sscanf(tmpLine(40:52),'%g',1);
1088
1089      elseif UFF.analysisType == 3, %Complex Eigenvalue, First Order (Displacement)
1090          %Line 7
1091          tmpLine = DATA(blockLines(7,1):blockLines(7,2));
1092          tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
1093          lineN = lineN + 1;
1094          two = sscanf(tmpLine(1:10),'%i',1);
1095          if two ~= 2,
1096              errMessage = ['unexpected value at line ' num2str(lineN) ' relatively to current data-set'];
1097              return
1098          end
1099          six = sscanf(tmpLine(11:20),'%i',1);
1100          if six ~= 6,
1101              errMessage = ['unexpected value at line: ' num2str(lineN) ' relatively to current data-set'];
1102              return
1103          end
1104          tmp = sscanf(tmpLine(21:30),'%i',1);
1105          UFF.modeNum = sscanf(tmpLine(31:40),'%i',1);
1106
1107          %Line 8
1108          tmpLine = DATA(blockLines(8,1):blockLines(8,2));
1109          tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
1110          lineN = lineN + 1;
1111          real_part = sscanf(tmpLine(1:13),'%g',1);
1112          imaginary_part = sscanf(tmpLine(14:26),'%g',1);
1113          UFF.eigVal = real_part + j * imaginary_part;
1114          real_part = sscanf(tmpLine(27:39),'%g',1);
1115          imaginary_part = sscanf(tmpLine(40:52),'%g',1);
1116          UFF.modalA = real_part + j * imaginary_part;
1117          real_part = sscanf(tmpLine(53:65),'%g',1);
1118          imaginary_part = sscanf(tmpLine(66:78),'%g',1);
1119          UFF.modalB = real_part + j * imaginary_part;
1120
1121      elseif UFF.analysisType == 5, %Frequency Response
1122          %Line 7
1123          tmpLine = DATA(blockLines(7,1):blockLines(7,2));
1124          tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
```

```
1125            lineN = lineN + 1;
1126            two = sscanf(tmpLine(1:10),'%i',1);
1127            if two ~= 2,
1128                errMessage = ['unexpected value at line ' num2str(lineN) ' relatively to current data-set'];
1129                return
1130            end
1131            one = sscanf(tmpLine(11:20),'%i',1);
1132            if one ~= 1,
1133                errMessage = ['unexpected value at line ' num2str(lineN) ' relatively to current data-set'];
1134                return
1135            end
1136            tmp = sscanf(tmpLine(21:30),'%i',1);
1137            UFF.freqNum = sscanf(tmpLine(31:40),'%i',1);
1138
1139            % Line 8
1140            tmpLine = DATA(blockLines(8,1):blockLines(8,2));
1141            tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
1142            lineN = lineN + 1;
1143            UFF.freq = sscanf(tmpLine(1:13),'%g',1);     % in Hz
1144
1145        elseif UFF.analysisType == 7   % Complex Eigenvalue, Second Order (Velocity)
1146            % Line 7
1147            tmpLine = DATA(blockLines(7,1):blockLines(7,2));
1148            tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
1149            lineN = lineN + 1;
1150            two = sscanf(tmpLine(1:10),'%i',1);
1151            if two ~= 2,
1152                errMessage = ['unexpected value at line ' num2str(lineN) ' relatively to current data-set'];
1153                return
1154            end
1155            six = sscanf(tmpLine(11:20),'%i',1);
1156            if six ~= 6,
1157                errMessage = ['unexpected value at line ' num2str(lineN) ' relatively to current data-set'];
1158                return
1159            end
1160            tmp = sscanf(tmpLine(21:30),'%i',1);
1161            UFF.modeNum = sscanf(tmpLine(31:40),'%i',1);
1162
1163            % Line 8
1164            tmpLine = DATA(blockLines(8,1):blockLines(8,2));
1165            tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
1166            lineN = lineN + 1;
1167            real_part = sscanf(tmpLine(1:13),'%g',1);
1168            imaginary_part = sscanf(tmpLine(14:26),'%g',1);
1169            UFF.eigVal = real_part + j * imaginary_part;
1170            real_part = sscanf(tmpLine(27:39),'%g',1);
1171            imaginary_part = sscanf(tmpLine(40:52),'%g',1);
1172            UFF.modalA = real_part + j * imaginary_part;
1173            real_part = sscanf(tmpLine(53:65),'%g',1);
1174            imaginary_part = sscanf(tmpLine(66:78),'%g',1);
1175            UFF.modalB = real_part + j * imaginary_part;
1176
1177        else
1178            errMessage = ['analysis type is not supported at line ' num2str(lineN) ' relatively to current data
                    -set'];
1179            return
1180        end
1181
1182        % Read response data by x,y,... components into r1..r6
1183        ii = 0;
1184        nnodes = floor((nLines-9)/2)+1;
1185        r1 = zeros(nnodes,1);
1186        r2 = r1;
1187        r3 = r1;
1188        r4 = r1;
1189        r5 = r1;
1190        r6 = r1;
1191        nodeNum = r1;
1192        for lne = 9:2:nLines-1,
1193            ii = ii + 1;
1194            %========
1195            %    line = 9; % line 9 type of line
1196            %========
1197            lineRead = lne;
1198            tmpLine = DATA(blockLines(lineRead,1):blockLines(lineRead,2));
1199            nodeNum(ii) = sscanf(tmpLine(1:10),'%i',1);
1200            lineN = lineN + 1;
1201
1202            %========
1203            %    line = 10; % line 9 type of line
1204            %========
1205            lineRead = lne + 1;
1206            lineN = lineN + 1;
1207            tmpLine = DATA(blockLines(lineRead,1):blockLines(lineRead,2));
1208  %           tmpLine = [tmpLine repmat(' ',1,80-length(tmpLine))];
1209            if UFF.dataType == 2,        % real data
1210                r1(ii) = sscanf(tmpLine(1:13),'%g',1);
1211                r2(ii) = sscanf(tmpLine(14:26),'%g',1);
1212                r3(ii) = sscanf(tmpLine(27:39),'%g',1);
1213                if num_data_per_pt == 6,
1214                    r4(ii) = sscanf(tmpLine(40:52),'%g',1);
1215                    r5(ii) = sscanf(tmpLine(53:65),'%g',1);
1216                    r6(ii) = sscanf(tmpLine(66:78),'%g',1);
1217                end
1218            elseif UFF.dataType == 5,    % complex data
1219                p1 = sscanf(tmpLine(1:13),'%g',1);
```

```
            p2 = sscanf(tmpLine(14:26),'%g',1);
            r1(ii) = p1 + j * p2;
            p3 = sscanf(tmpLine(27:39),'%g',1);
            p4 = sscanf(tmpLine(40:52),'%g',1);
            r2(ii) = p3 + j * p4;
            p5 = sscanf(tmpLine(53:65),'%g',1);
            p6 = sscanf(tmpLine(66:78),'%g',1);
            r3(ii) = p5 + j * p6;
            if num_data_per_pt == 6,
                errMessage = ['not setup to handle six coordinate of complex data at line ' num2str(lineN)
                    ' relatively to current data-set'];
                return
            end
        else
            errMessage = sprintf('Unknown dataType (%d). Only dataType=2 (real data) and dataType=5 (
                complex data) are supported', UFF.dataType);
            return
        end
    end
    UFF.r1 = r1;
    UFF.r2 = r2;
    UFF.r3 = r3;
    UFF.r4 = r4;
    UFF.r5 = r5;
    UFF.r6 = r6;
    UFF.nodeNum = nodeNum;
catch
    errMessage = ['error reading modal data: ' lasterr];
    return
end



%--------------------------------------------------------------------------
function outstr = strim(str)
% Removes leading and trailing spaces (spaces, tabs, endlines,...)
% from the str string.
if isnumeric(str);
    outstr = str;
    return
end
ind = find( ~isspace(str) );        % indices of the non-space characters in the str
if isempty(ind)
    outstr = [];
else
    outstr = str( ind(1):ind(end) );
end







%--------------------------------------------------------------------------
function [UFF,errMessage] = extract2411(DATA,blockLines)
% #2411 - Extract data-set type 2411 data
% Added by Ben Cazzolato, 10/3/2008
%
% Universal Dataset Number 2411
% by zopeown last modified 2007-05-02 06:56
%
% Name:    Nodes - Double Precision
% Status: Current
% Owner:   Simulation
% Revision Date: 23-OCT-1992
%--------------------------------------------------------------------------
%
% Record 1:         FORMAT(4I10)
%                   Field 1       -- node label
%                   Field 2       -- export coordinate system number
%                   Field 3       -- displacement coordinate system number
%                   Field 4       -- color
% Record 2:         FORMAT(1P3D25.16)
%                   Fields 1-3    -- node coordinates in the part coordinate
%                                    system
%
% Records 1 and 2 are repeated for each node in the model.
%
% Example:
%
%     -1
%    2411
%         121          1          1         11
%     5.0000000000000000D+00    1.0000000000000000D+00    0.0000000000000000D+00
%         122          1          1         11
%     6.0000000000000000D+00    1.0000000000000000D+00    0.0000000000000000D+00
%     -1
%
%--------------------------------------------------------------------------
UFF = [];
errMessage = [];
nLines = size(blockLines,1);

try
    values = sscanf(DATA(blockLines(1,1):blockLines(end,2)),'%g');
```

```
1314        nVals = length(values);
1315        nNodes = round(nVals/7);
1316        values = reshape(values,7,nNodes).';
1317        %
1318        UFF.nodeN = round(values(:,1));
1319        UFF.defCS = round(values(:,2));
1320        UFF.dispCS = round(values(:,3));
1321        UFF.color = round(values(:,4));
1322        UFF.x = values(:,5);
1323        UFF.y = values(:,6);
1324        UFF.z = values(:,7);
1325    catch
1326        errMessage = ['error reading coordinate data: ' lasterr];
1327        return
1328    end
1329
1330
1331
1332
1333
1334
1335    %-------------------------------------------------------------------------------
1336    function [UFF,errMessage] = extract2412(DATA,blockLines)
1337
1338    % Define all "beam like" elements since these have a different structure
1339    beam_like = [11,21:24,31:32,121:122];
1340    Largest_Num_Nodes = 20;      % This is used to zero pad the data if different element types present
1341    UFF = [];
1342    errMessage = [];
1343    % Initialise matrices containing field types
1344    ElementLabel = [];
1345    FEDescriptor = [];
1346    PhysicalProp = [];
1347    MaterialProp = [];
1348    ElementColour = [];
1349    NumNodes = [];
1350    Element = [];
1351    try
1352        values = sscanf(DATA(blockLines(1,1):blockLines(end,2)),'%g');
1353        nVals = length(values);
1354        data_remaining = 1;
1355        while data_remaining
1356            ElementLabel = [ElementLabel;round(values(1))];
1357            FEDescriptor = [FEDescriptor;round(values(2))];
1358            PhysicalProp = [PhysicalProp;round(values(3))];
1359            MaterialProp = [MaterialProp;round(values(4))];
1360            ElementColour = [ElementColour;round(values(5))];
1361            NumNodes = [NumNodes;round(values(6))];
1362            % Check for beam elements
1363            if sum(round(values(2))==beam_like)
1364                % Beam Element
1365                %disp('Beam Elements')
1366                Element = [Element;[[round(values(7:6+3+round(values(6))))]',NaN*zeros(1,Largest_Num_Nodes-3-
                        round(values(6)))]];
1367                values = values(7+3+round(values(6)):end);    % Remove the element from the table
1368            else
1369                % Not Beam Element
1370                Element = [Element;[[round(values(7:6+round(values(6))))]',NaN*zeros(1,Largest_Num_Nodes-round(
                        values(6)))]];
1371                values = values(7+round(values(6)):end);    % Remove the element from the table
1372            end
1373            if isempty(values)       % Check if any data remaining
1374                data_remaining=0;
1375            end
1376        end
1377        UFF.ElementLabel = ElementLabel;
1378        UFF.FEDescriptor = FEDescriptor;
1379        UFF.PhysicalProp = PhysicalProp;
1380        UFF.MaterialProp = MaterialProp;
1381        UFF.ElementColour = ElementColour;
1382        UFF.NumNodes = NumNodes;
1383        % Strip unnecessary columns from element matrix
1384        temp = find(sum(~isnan(Element))>0);
1385        UFF.Element = Element(:,temp);
1386    catch
1387        errMessage = ['error reading trace-line data at line' num2str(lineN) ' relatively to current data-set:
                ' lasterr];
1388        return
1389    end
```

Función TDF [TDF Normalizada]

```
%% SE COMIENZA LIMPIANDO TODO

clc
clear all
close all

%% SE EXTRAEN DATOS EXPERIMENTALES 3

load('ACC_PLANETA_3')

load('T_PLANETA_3')

FS_PLANETA_3 = 1/mean(diff(T_PLANETA_3(1,:)));

%% TDF DE DATOS EXPERIMENTALES 3

[A,B] = size(ACC_PLANETA_3);

F_PLANETA_3 = zeros(A,B/2+1);

X_PLANETA_3 = zeros(A,B/2+1);

Z_PLANETA_3 = zeros(A,B/2+1);

for i = 1:A

    [F_PLANETA_3(i,:),X_PLANETA_3(i,:),Z_PLANETA_3(i,:)] = TDF(ACC_PLANETA_3(i,:),FS_PLANETA_3);

end

%% ANALISIS DE ENVOLVENTE DEL PLANETA 3

F_R = FS_PLANETA_3 / length(ACC_PLANETA_3);
m = round(1/F_R);                       % Limite inferior de los datos seleccionados
n = round(200/F_R);                       % Limite superior de los datos seleccionados

X_ENV_PLANETA_3 = zeros(A,B/2+1);       % Se crea un vector para guardar la amplitud de la TDF de la
    envolvente de la senal
F_ENV_PLANETA_3 = zeros(A,B/2+1);       % Se crea un vector para guardar la frecuencia de la TDF de la
    envolvente de la senal
Z_ENV_PLANETA_3 = zeros(A,B/2+1);       % Se crea un vector para guardar la fase de la TDF de la envolvente
    de la senal

X_ENV_MOD_PLANETA_3 = zeros(A,n-m);     % Se crea un vector para guardar la amplitud de la TDF de la
    envolvente cortada
F_ENV_MOD_PLANETA_3 = zeros(A,n-m);     % Se crea un vector para guardar la frecuencia de la TDF de la
    envolvente cortada

ACC_env = zeros(size(ACC_PLANETA_3));
low_ACC_env = zeros(size(ACC_PLANETA_3));

for i = 1:A
    [ACC_env(i,:),low_ACC_env(i,:)] = envelope(ACC_PLANETA_3(i,:),100,'peak');    % Se calcula la envolvente
        de la aceleracion
end

%Se grafican las envolventes de las senales

figure
for i =1:1
    plot(T_PLANETA_3(1,:), ACC_PLANETA_3(i,:),'LineWidth',1.5);
    hold on
    plot(T_PLANETA_3(1,:), ACC_env(i,:),'LineWidth',1.5);
end
xlim([min(T_PLANETA_3(1,:)) max(T_PLANETA_3(1,:))])
xlabel('Tiempo [s]', 'Interpreter', 'Latex', 'FontSize', 12);
ylabel('Amplitud  [g]','Interpreter', 'Latex','FontSize', 12)
% title('Comparacion de la senal y su envolvente','Interpreter','Latex','FontSize',12)
legend('Senal original','Envolvente de la senal')


% Se calcula la TDF de la envolvente de las aceleraciones experimentales

for i = 1:A
    [a,b,c] = TDF(ACC_env(i,:),FS_PLANETA_3);                % Se calcula la transformada de la envolvente
    F_ENV_PLANETA_3(i,:) = a;
    X_ENV_PLANETA_3(i,:) = b;
    Z_ENV_PLANETA_3(i,:) = c;
    X_ENV_MOD_PLANETA_3(i,:) = X_ENV_PLANETA_3(i,m:n-1)/max(X_ENV_PLANETA_3(i,m:n-1));    % Se normalizan y
        se guardan los datos en el vector
    F_ENV_MOD_PLANETA_3(i,:) = F_ENV_PLANETA_3(i,m:n-1);                        % Se guardan las
        frecuencias en el vector
end

%Se grafican la transformada de las envolventes de las senales

figure
for i =1:A
    hold on
    plot(F_ENV_MOD_PLANETA_3(i,:), X_ENV_MOD_PLANETA_3(i,:),'LineWidth',1.5);
end
xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 12);
ylabel('Amplitud  [g]','Interpreter', 'Latex','FontSize', 12)
box on
```

```
87  %title('Espectro de Fourier de la envolvente','Interpreter','Latex','FontSize',14)
88
89  M = zeros(A,1);       %En este vector se guardan los maximos de la TDF de la envolvente
90  I = zeros(A,1);       %En este vector se guardan los indices de los maximos de la TDF de la envolvente
91
92  for i = 1:A
93      [M(i),I(i)] = max(X_ENV_MOD_PLANETA_3(i,:));     %Se guardan tanto los valores como los indices
94  end
95
96  F_MAX_PLANETA_3 = zeros(1,A);             %Se genera un vector donde se van a guardar las frecuencias de
        giro
97  F_PLANETA_3_NORM = zeros(A,B/2+1);        %Se genera un vector donde se van a guardar las frecuecias
        normalizadas
98
99  %Se guardan las frecuencias de giro para cada medicion
100
101 N = 4;        %Numero de elementos que se toman ademas del central
102
103 for i = 1:A
104     F_MAX_PLANETA_3(i) = sum(F_ENV_MOD_PLANETA_3(i,I(i)-N:I(i)+N).* X_ENV_MOD_PLANETA_3(i,I(i)-N:I(i)+N)/
            sum(X_ENV_MOD_PLANETA_3(i,I(i)-N:I(i)+N)));
105 end
106
107 %Se normaliza dividiendo por la frecuencia de giro
108
109 for i = 1:A
110     F_PLANETA_3_NORM(i,:) = F_PLANETA_3(i,:)/F_MAX_PLANETA_3(i);
111 end
112
113 %INTERPOLACION DE LOS DATOS
114
115 L = length(0:0.01:350);                  %Se calcula el largo para el nuevo vector de frecuencias que se
        quiere utilizar
116 F_PLANETA_3_RES = zeros(A,L);             %Se crea un vector para guardar las frecuencias resampleadas
        despues de la interpolacion
117 X_PLANETA_3_RES = zeros(A,L);             %Se crea un vector para guardar las amplitudes resampleadas
        despues de la interpolacion
118
119 %Se realiza la interpolacion para cada conjunto de datos
120
121 for i = 1:A
122     F_PLANETA_3_RES(i,:) = 0:0.01:350;
123     X_PLANETA_3_RES(i,:) = interp1(F_PLANETA_3_NORM(i,:),X_PLANETA_3(i,:),F_PLANETA_3_RES(i,:));
124     X_PLANETA_3_RES(i,:) = X_PLANETA_3_RES(i,:)/max(X_PLANETA_3_RES(i,:));
125 end
126
127 %% SE GRAFICA LA TDF CON EL EJE DE FRECUENCIAS NORMALIZADA
128
129 figure
130
131 for i = 1:1
132     hold on
133     plot(F_PLANETA_3_RES(i,:),X_PLANETA_3_RES(i,:),'Linewidth',1.5);
134 end
135 xlabel('Ordenes [S/U]', 'Interpreter', 'Latex', 'FontSize', 12);
136 ylabel('Amplitud  [g]','Interpreter', 'Latex','FontSize', 12)
137 box on
```

Apéndice C

FILTRO AR

Aplicación del filtro a datos experimentales [Filtro AR]

```
%% SE COMIENZA LIMPIANDO TODO

clc
clear all
close all

% SE EXTRAEN DATOS EXPERIMENTALES 3

load('ACC_PLANETA_3')

load('T_PLANETA_3')

FS_PLANETA_3 = 1/mean(diff(T_PLANETA_3(1,:))); % Se calcula la frecuencia de adquisicion.

%% SE APLICA EL FILTRO AR A LOS DATOS EXPERIMENTALES 3

[A,B] = size(ACC_PLANETA_3); % Se extrae el tamano de la matriz.

ACC_AR_PLANETA_3 = zeros(A,B);

for i = 1:A
    X =['Porcentaje de avance filtro AR: ',num2str(i*100/A),' %'];     % Se determina el porcentaje de
        avance del proceso [%]
    disp(X)
    [ACC_AR_PLANETA_3(i,:)] = AR_FILTER(ACC_PLANETA_3(i,:));
end

%% SE COMPARAN LOS DATOS ANTES Y DESPUES DEL FILTRO

k = 5;  % Se selecciona solo un conjunto de datos.

figure
for i =k:k
    hold on
    plot(T_PLANETA_3(1,:), ACC_PLANETA_3(i,:),'LineWidth',1.5);
    plot(T_PLANETA_3(1,:), ACC_AR_PLANETA_3(i,:),'LineWidth',1.5);
end
xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 14);
xlim([4  4.1])
ylabel('Amplitud  [$m/s^2$]','Interpreter', 'Latex','FontSize', 14)
title('Espectro de Fourier de la envolvente','Interpreter','Latex','FontSize',14)
legend('Datos sin procesar','Datos con filtro AR')

%% SE GUARDAN LOS DATOS

T_AR_PLANETA_3 = T_PLANETA_3;

save('T_AR_PLANETA_3','T_AR_PLANETA_3')

save('ACC_AR_PLANETA_3','ACC_AR_PLANETA_3')
```

Función del filtro Autoregresivo [Filtro AR]

```
function [AR_FILT_Y] = AR_FILTER(signal)

clear y y_e A    %Se eliminan variables que pueden estar ocupadas

%Se itera a traves de los ordenes del filtro AR para obtener uno que
%tenga la menor kurtosis, para dejar solamente la senal estacionaria.

for order = 1:100

[A,E]=aryule(signal,order);                 %Se genera un filtro AR de orden n
y(:,order)=signal - filter(2,A,signal);     %Se usa la diferencia entre la senal y su filtro AR
kurt_y(:, order)=kurtosis(y(:,order));      %Se calcula la kurtosis para cada caso

end


[~,index_y]=sort(kurt_y,'descend');          %Se guardan en orden descendente los indices de la kurtosis de
      las senales

AR_FILT_Y =y(:,index_y(1));                  %El filtro AR toma  la senal con mayor kurtosis para pasar al
     proceso de filtrado AR

end
```

Gráfico Kurtosis vs Órden del filtro [Filtro AR]

```
function [AR_FILT_Y] = AR_FILTER(signal)

clear y y_e A    %Se eliminan variables que pueden estar ocupadas

%Se itera a traves de los ordenes del filtro AR para obtener uno que
%tenga la menor kurtosis, para dejar solamente la senal estacionaria.

for order = 1:100

[A,E]=aryule(signal,order);                 %Se genera un filtro AR de orden n
y(:,order)=signal - filter(2,A,signal);     %Se usa la diferencia entre la senal y su filtro AR
kurt_y(:, order)=kurtosis(y(:,order));      %Se calcula la kurtosis para cada caso

end


[~,index_y]=sort(kurt_y,'descend');          %Se guardan en orden descendente los indices de la kurtosis de
      las senales

AR_FILT_Y =y(:,index_y(1));                  %El filtro AR toma  la senal con mayor kurtosis para pasar al
     proceso de filtrado AR

end
```

Gráficos TDF [Filtro AR]

```
%% SE COMIENZA LIMPIANDO TODO

clc
clear all
close all

%% SE EXTRAEN DATOS EXPERIMENTALES 3

load('ACC_AR_PLANETA_3')

load('T_AR_PLANETA_3')

load('ACC_PLANETA_3')

load('T_PLANETA_3')

FS_PLANETA_3 = 1/mean(diff(T_PLANETA_3(1,:)));
FS_AR_PLANETA_3 = 1/mean(diff(T_AR_PLANETA_3(1,:)));

%% TDF DE DATOS EXPERIMENTALES 3

[A,B] = size(ACC_PLANETA_3);

F_PLANETA_3 = zeros(A,B/2+1);
F_AR_PLANETA_3 = zeros(A,B/2+1);

X_PLANETA_3 = zeros(A,B/2+1);
X_AR_PLANETA_3 = zeros(A,B/2+1);

Z_PLANETA_3 = zeros(A,B/2+1);
Z_AR_PLANETA_3 = zeros(A,B/2+1);

for i = 1:A
    [F_PLANETA_3(i,:),X_PLANETA_3(i,:),Z_PLANETA_3(i,:)] = TDF(ACC_PLANETA_3(i,:),FS_PLANETA_3);
```

```
35      [F_AR_PLANETA_3(i,:),X_AR_PLANETA_3(i,:),Z_AR_PLANETA_3(i,:)] = TDF(ACC_AR_PLANETA_3(i,:),
            FS_AR_PLANETA_3);
36
37  end
38
39  %% ANALISIS DE ENVOLVENTE DEL PLANETA 3
40
41  F_R = FS_PLANETA_3 / length(ACC_PLANETA_3);
42  m = round(20/F_R);                          % Limite inferior de los datos seleccionados
43  n = round(50/F_R);                          % Limite superior de los datos seleccionados
44
45  X_ENV_PLANETA_3 = zeros(A,B/2+1);       % Se crea un vector para guardar la amplitud de la TDF de la
        envolvente de la senal
46  F_ENV_PLANETA_3 = zeros(A,B/2+1);       % Se crea un vector para guardar la frecuencia de la TDF de la
        envolvente de la senal
47  Z_ENV_PLANETA_3 = zeros(A,B/2+1);       % Se crea un vector para guardar la fase de la TDF de la envolvente
        de la senal
48
49  X_ENV_MOD_PLANETA_3 = zeros(A,n-m);     % Se crea un vector para guardar la amplitud de la TDF de la
        envolvente cortada
50  F_ENV_MOD_PLANETA_3 = zeros(A,n-m);     % Se crea un vector para guardar la frecuencia de la TDF de la
        envolvente cortada
51
52  ACC_env = envelope(ACC_PLANETA_3,100,'peak');    % Se calcula la envolvente de la aceleracion
53
54  % Se calcula la TDF de la envolvente de las aceleraciones experimentales
55
56  for i = 1:A
57      [a,b,c] = TDF(ACC_env(i,:),FS_PLANETA_3);                % Se calcula la transformada de la envolvente
58      F_ENV_PLANETA_3(i,:) = a;
59      X_ENV_PLANETA_3(i,:) = b;
60      Z_ENV_PLANETA_3(i,:) = c;
61      X_ENV_MOD_PLANETA_3(i,:) = X_ENV_PLANETA_3(i,m:n-1)/max(X_ENV_PLANETA_3(i,m:n-1));     % Se normalizan y
            se guardan los datos en el vector
62      F_ENV_MOD_PLANETA_3(i,:) = F_ENV_PLANETA_3(i,m:n-1);                          % Se guardan las
            frecuencias en el vector
63  end
64
65  %Se grafican las envolventes de las senales
66
67  % figure
68  % for i =1:A
69  %     hold on
70  %     plot(F_ENV_MOD_PLANETA_1(i,:), X_ENV_MOD_PLANETA_1(i,:))
71  %     xlim([min(F_ENV_MOD_PLANETA_3(i,:)),max(F_ENV_MOD_PLANETA_3(i,:))])
72  % end
73  % xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 14);
74  % ylabel('Amplitud  [m]','Interpreter', 'Latex','FontSize', 14)
75  % title('Espectro de Fourier de la envolvente','Interpreter','Latex','FontSize',14)
76
77  M = zeros(A,1);      % En este vector se guardan los maximos de la TDF de la envolvente
78  I = zeros(A,1);      % En este vector se guardan los indices de los maximos de la TDF de la envolvente
79
80  for i = 1:A
81      [M(i),I(i)] = max(X_ENV_MOD_PLANETA_3(i,:));     % Se guardan tanto los valores como los indices
82  end
83
84  F_MAX_PLANETA_3 = zeros(1,A);            % Se genera un vector donde se van a guardar las frecuencias de
        giro
85  F_PLANETA_3_NORM = zeros(A,B/2+1);       % Se genera un vector donde se van a guardar las frecuecias
        normalizadas
86
87  % Se guardan las frecuencias de giro para cada medicion
88
89  N = 2;       % Numero de elementos que se toman ademas del central
90
91  for i = 1:A
92      F_MAX_PLANETA_3(i) = sum(F_ENV_MOD_PLANETA_3(i,I(i)-N:I(i)+N).* X_ENV_MOD_PLANETA_3(i,I(i)-N:I(i)+N))/
            sum(X_ENV_MOD_PLANETA_3(i,I(i)-N:I(i)+N));
93  end
94
95  % Se normaliza dividiendo por la frecuencia de giro
96
97  for i = 1:A
98      F_PLANETA_3_NORM(i,:) = F_PLANETA_3(i,:)/F_MAX_PLANETA_3(i);
99  end
100
101 % INTERPOLACION DE LOS DATOS
102
103 L = length(0:0.01:350);                  % Se calcula el largo para el nuevo vector de frecuencias que se
        quiere utilizar
104 F_AR_PLANETA_3_RES = zeros(A,L);            % Se crea un vector para guardar las frecuencias resampleadas
        despues de la interpolacion
105 X_AR_PLANETA_3_RES = zeros(A,L);            % Se crea un vector para guardar las amplitudes resampleadas
        despues de la interpolacion
106
107 % Se realiza la interpolacion para cada conjunto de datos
108
109 for i = 1:A
110     F_AR_PLANETA_3_RES(i,:) = 0:0.01:350;
111     X_AR_PLANETA_3_RES(i,:) = interp1(F_PLANETA_3_NORM(i,:),X_AR_PLANETA_3(i,:),F_AR_PLANETA_3_RES(i,:));
112     X_AR_PLANETA_3_RES(i,:) = X_AR_PLANETA_3_RES(i,:)/max(X_AR_PLANETA_3_RES(i,:));
113 end
114
115 % Se grafica una comparacion de los datos inpoerlados y los sin interpolar
116
```

```
% figure
%
% for i = 1:A
%     hold on
%     plot(F_AR_PLANETA_3_RES(i,:),X_AR_PLANETA_3_RES(i,:));
%     title('(Default) Linear Interpolation');
% end
% xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 14);
% ylabel('Amplitud  [m]','Interpreter', 'Latex','FontSize', 14)
% title('Espectro de Fourier normalizado','Interpreter','Latex','FontSize',14)

%% SE GRAFICAN LOS DATOS EXPERIMENTALES ANTES DE LA MODIFICACION

% figure
%
% for i = 1:1
%     hold on
%     plot(F_AR_PLANETA_3_RES(i,:), X_AR_PLANETA_3_RES(i,:),'LineWidth',1)
% end
%
% xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 14);
% ylabel('Amplitud  [m]','Interpreter', 'Latex','FontSize', 14)
% title('Espectro de Fourier normalizado','Interpreter','Latex','FontSize',14)
% legend()

%% SE ELIMINAN LAS FRECUENCIAS INNECESARIAS

S1 = 1;
S2 = 20000;

X_AR_PLANETA_3_RES = X_AR_PLANETA_3_RES (:,S1:S2);

F_AR_PLANETA_3_RES = F_AR_PLANETA_3_RES (:,S1:S2);


%% SE GRAFICAN LOS DATOS EXPERIMENTALES DESPUES DEL FILTRO

figure

k = 1;

for i = k:k
    hold on
    plot(F_AR_PLANETA_3_RES(i,:), X_AR_PLANETA_3_RES(i,:))
end
xlim([0 60])
xlabel('Ordenes [S/U]', 'Interpreter', 'Latex', 'FontSize', 12);
ylabel('Amplitud  [g]','Interpreter', 'Latex','FontSize', 12)
legend('Senal con filtro AR')
box on

%% SE GRAFICAN LOS DATOS EXPERIMENTALES ANTES DEL FILTRO

figure

k = 1;

for i = k:k
    hold on
    plot(F_PLANETA_3(i,:)/F_MAX_PLANETA_3(i), X_PLANETA_3(i,:)/max(X_PLANETA_3(i,:)),'LineWidth',0.1)
end
xlim([0 60])
xlabel('Ordenes [S/U]', 'Interpreter', 'Latex', 'FontSize', 12);
ylabel('Amplitud  [g]','Interpreter', 'Latex','FontSize', 12)
legend('Senal sin filtro AR')
box on

%% SE GRAFICAN LOS DATOS EXPERIMENTALES ANTES Y DESPUES DEL FILTRO

figure

k = 1;

for i = k:k
    hold on
    plot(F_AR_PLANETA_3_RES(i,:), X_AR_PLANETA_3_RES(i,:),'LineWidth',0.1)
    plot(F_PLANETA_3(i,:)/F_MAX_PLANETA_3(i), X_PLANETA_3(i,:)/max(X_PLANETA_3(i,:)),'LineWidth',0.1)
end
xlim([0 60])
xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 12);
ylabel('Amplitud  [m]','Interpreter', 'Latex','FontSize', 12)
legend('Senal con filtro AR','Senal sin filtrar')
```

Apéndice D

FILTRO MED

Aplicación del filtro a datos experimentales [Filtro MED]

```
%% SE COMIENZA LIMPIANDO TODO

clc
clear all
close all

% SE EXTRAEN DATOS EXPERIMENTALES 3

load('ACC_PLANETA_3')

load('T_PLANETA_3')

FS_PLANETA_3 = 1/mean(diff(T_PLANETA_3(1,:))); % Se calcula la frecuencia de adquisicion.

%% SE APLICA EL FILTRO MED A LOS DATOS EXPERIMENTALES 3

[A,B] = size(ACC_PLANETA_3); % Se extrae el tamano de la matriz.

ACC_MED_PLANETA_3 = zeros(A,B);

for i = 1:A
    X =['Porcentaje de avance filtro MED: ',num2str(i*100/A),' %];     % Se determina el porcentaje de
        avance del proceso [%]
    disp(X)
    [ACC_MED_PLANETA_3(i,:) f_armed kurt_armed] = MED_FILTER(ACC_PLANETA_3(i,:),30,[],0.01,0);
end

%% SE COMPARAN LOS DATOS ANTES Y DESPUES DEL FILTRO

k = 5;  % Se selecciona solo un conjunto de datos.

figure
for i =k:k
    hold on
    plot(T_PLANETA_3(1,:), ACC_PLANETA_3(i,:),'LineWidth',1.5);
    plot(T_PLANETA_3(1,:), ACC_MED_PLANETA_3(i,:),'LineWidth',1.5);
end
xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 14);
xlim([4 4.1])
ylabel('Amplitud [$m/s^2$]','Interpreter', 'Latex','FontSize', 14)
title('Espectro de Fourier de la envolvente','Interpreter','Latex','FontSize',14)
legend('Datos sin procesar','Datos con filtro MED')

%% SE GUARDAN LOS DATOS

T_MED_PLANETA_3 = T_PLANETA_3;

save('T_MED_PLANETA_3','T_MED_PLANETA_3')

save('ACC_MED_PLANETA_3','ACC_MED_PLANETA_3')
```

Función del filtro MED [Filtro MED]

```
function [y_final f_final kurtIter] = MED_FILTER(x,filterSize,termIter,termDelta,plotMode)
    %2D MINIMUM ENTROPY DECONVOLUTION
    % code by Geoff McDonald (glmcdona@gmail.com), February 2011
    % Used in my MSc research at the University of Alberta, Advanced
    % Control Systems Laboratory.
    %
    %med2d(x,filterSize,termIter,termDelta,plotMode)
    %
    %Algorithm Reference:
    %   R.A. Wiggins, Minimum Entropy Deconvolution, Geoexploration, vol.
    %   16, Elsevier Scientific Publishing, Amsterdam, 1978. pp. 2135.
    %
    %Inputs:
    %   x:
    %       Signal to perform Minimum Entropy Deconvolution on. If a single
    %       column/row of data is specified, a 1d filter is designed to
    %       minimize the entropy of the resulting signale. If a 2d data
    %       matrix is specified, a single 1d filter will be designed to
    %       minimize the averaged entropy of each column of the filtered
    %       data.
    %
    %   filterSize:
    %       This is the length of the finite inpulse filter filter to
    %       design. Using a value of around 30 is appropriate depending on
    %       the data. Investigate the performance difference using
    %       different values.
    %
    %   termIter: (OPTIONAL)
    %       This is the termination number of iterations. If the
    %       the number of iterations exceeds this number, the MED process
    %       will complete. Specify [] to use default value of 30.
    %
    %   termDelta: (OPTIONAL)
    %       This is the termination condition. If the change in kurtosis
    %       between iterations is below this threshold, the iterative
    %       process will terminate. Specify [] to use the default value
    %       of 0.01. You can specify a value of 0 to only terminate on
    %       the termIter condition, ie. execute an exact number of
    %       iterations.
    %
    %   plotMode:
    %       If this value is > 0, plots will be generated of the iterative
    %       performance and of the resulting signal.
    %
    %Outputs:
    %   y_final:
    %       The input signal(s) x, filtered by the resulting MED filter.
    %       This is obtained simply as: y_final = filter(f_final,1,x);
    %
    %   f_final:
    %       The final 1d MED filter in finite impulse response format.
    %
    %   kurtIter:
    %       Kurtosis according to MED iteration. kurtIter(end) is the
    %       final kurtosis, ie. the summed kurtosis of each y_final
    %       column of y_final. sum(kurtosis(each column of y_final))
    %
    %Example:
    %       %This will mostly extract the impulse-like
    %       %disturbances caused by 0.2*(mod(n,21)==0)
    %       %and plot the result.
    %       n = 0:999;
    %       x = [sin(n/30) + 0.2*(mod(n,21)==0);
    %            sin(n/13) + 0.2*(mod(n,21)==0)];
    %       [y_final f_final kurt] = med2d(x',30,[],0.01,1);
    %
    %
    %Note:
    %  The solution is not guaranteed to be the optimal solution to the
    %  entropy minimizataion problem, the solution is just a local
    %  minimum of the entropy and therefore a good pick.


    %Assign default values for inputs
    if( isempty(filterSize) )
        filterSize = 30;
    end
    if( isempty(termIter) )
        termIter = 30;
    end
    if( isempty(termDelta) )
        termDelta = 0.01;
    end
    if( isempty(plotMode) )
        plotMode = 0;
    end

    %Validate the inputs
    if( sum( size(x) > 1 ) > 2 )
        error('MED:InvalidInput', 'Input signal x must be of either 2d or 1d.')
    elseif( sum(size(termDelta) > 1) ~= 0 || termDelta < 0 )
        error('MED:InvalidInput', 'Input argument termDelta must be a positive scalar, or zero.')
    elseif( sum(size(termIter) > 1) ~= 0 || mod(termIter, 1) ~= 0 || termIter <= 0 )
        error('MED:InvalidInput', 'Input argument termIter must be a positive integer scalar.')
```

```
    elseif(  sum(size(plotMode) > 1) ~= 0 )
        error('MED:InvalidInput', 'Input argument plotMode must be a scalar.')
    elseif( sum(size(filterSize) > 1) ~= 0 || filterSize <= 0 || mod(filterSize, 1) ~= 0 )
        error('MED:InvalidInput', 'Input argument filterSize must be a positive integer scalar.')
    end

    % If the data is 1d, lets make it a column vector
    if( sum(size(x)>1) == 1 )
        x = x(:); %A column vector
    end
    L = filterSize;

    % Calculate the weighted toeplitz autocorrelation matrix
    % as the average autocorrelation matrix of the rows.
    autoCorr = zeros(1,L);
    for column = 1:size(x,2);
        for k = 0:L-1
            % Create the shifted x
            x2 = zeros(size(x,1),1);
            x2(k+1:end) = x(1:end-k,column);

            % Calculate the autocorrelation at this shift
            autoCorr(k+1) = autoCorr(k+1) + sum(x(:,column).*x2);
        end
    end
    autoCorr = autoCorr / size(x,2); % Average normalization
    A = toeplitz(autoCorr);
    A_inv = inv(A);


    % Initialize matrix sizes
    f = zeros(L,1);
    y = zeros(size(x,1),size(x,2));
    b = zeros(L,1);
    kurtIter = [];

    % Assume initial filter as a delayed impulse. This decision
    % was made by paper:
    % H. Endo and R. Randall, Enhancement of autoregressive model based
    % gear tooth fault detection technique by the use of minimum entropy
    % deconvolution filter, Mechanical Systems and Signal Processing vol.21,
    % no.2, February 2007
    f(2) = 1;

    % Iteratively adjust the filter to minimize entropy
    n = 1;
    while n == 1 || ( n < termIter && ( (kurt(filter(f,1,x)) - kurtIter(n-1)) > termDelta ) )

        % Compute output signal
        y = filter(f,1,x);

        % Calculate the kurtosis
        kurtIter(n) = kurt(y); %#ok<AGROW>

        % Calculate the matrix g = weighted av{ crosscorr( y.^3, x) }
        yc = y.^3;
        weightedCrossCorr = zeros(L,1);
        for column = 1:size(x,2);
            for k = 0:L-1
                % Create the shifted x
                x2 = zeros(size(x,1),1);
                x2(k+1:end) = x(1:end-k,column);

                % Calculate the crosscorrelation at this shift
                weightedCrossCorr(k+1) = weightedCrossCorr(k+1) + sum((y(:,column).^3).*x2);
            end
        end
        weightedCrossCorr = weightedCrossCorr / size(x,2);

        % Now we have new filter coefficients calculted as:
        % f = A^-1 * g
        f = A_inv*weightedCrossCorr;
        f = f/sqrt(sum(f.^2)); % Normalize the filter result

        % Next iteration
        n = n + 1;
    end

    % Update the final result
    f_final = f;
    y_final = filter(f_final,1,x);
    kurtIter(n) = kurt(y_final);

    % Plot the results
    if( plotMode > 0 )

        figure;
        subplot(2,1,1)
        plot(x)
        title('Input Signal(s)')
        ylabel('Value')
        xlabel('Sample Number')
        axis tight

        subplot(2,1,2)
        plot(y_final)
```

```
191           title('Signal(s) Filtered by MED')
192           ylabel('Value')
193           xlabel('Sample Number')
194           axis tight
195
196           figure;
197           stem(f_final)
198           xlabel('Sample Number')
199           ylabel('Value')
200           title('Final Filter, Finite Impulse Response')
201
202           figure;
203           plot(kurtIter);
204           xlabel('MED Algorithm Iteration')
205           ylabel('Sum of Kurtosis for Filtered Signal(s)')
206
207           if( n == termIter )
208               display('Terminated for iteration condition.')
209           else
210               display('Terminated for minimum change in kurtosis condition.')
211           end
212       end
213   end
214
215   function [result] = kurt(x)
216       % This function simply calculates the summed kurtosis of the input
217       % signal, x.
218       result = mean( (sum((x-ones(size(x,1),1)*mean(x)).^4)./(size(x,1)-2))./(std(x).^4) );
219   end
```

Gráficos TDF [Filtro MED]

```
1   %% SE COMIENZA LIMPIANDO TODO
2
3   clc
4   clear all
5   close all
6
7   %% SE EXTRAEN DATOS EXPERIMENTALES 3
8
9   load('ACC_MED_PLANETA_3')
10
11  load('T_MED_PLANETA_3')
12
13  load('ACC_PLANETA_3')
14
15  load('T_PLANETA_3')
16
17  FS_PLANETA_3 = 1/mean(diff(T_PLANETA_3(1,:)));
18  FS_MED_PLANETA_3 = 1/mean(diff(T_MED_PLANETA_3(1,:)));
19
20  %% TDF DE DATOS EXPERIMENTALES 3
21
22  [A,B] = size(ACC_PLANETA_3);
23
24  F_PLANETA_3 = zeros(A,B/2+1);
25  F_MED_PLANETA_3 = zeros(A,B/2+1);
26
27  X_PLANETA_3 = zeros(A,B/2+1);
28  X_MED_PLANETA_3 = zeros(A,B/2+1);
29
30  Z_PLANETA_3 = zeros(A,B/2+1);
31  Z_MED_PLANETA_3 = zeros(A,B/2+1);
32
33  for i = 1:A
34      [F_PLANETA_3(i,:),X_PLANETA_3(i,:),Z_PLANETA_3(i,:)] = TDF(ACC_PLANETA_3(i,:),FS_PLANETA_3);
35      [F_MED_PLANETA_3(i,:),X_MED_PLANETA_3(i,:),Z_MED_PLANETA_3(i,:)] = TDF(ACC_MED_PLANETA_3(i,:),
            FS_MED_PLANETA_3);
36
37  end
38
39  %% ANALISIS DE ENVOLVENTE DEL PLANETA 3
40
41  F_R = FS_PLANETA_3 / length(ACC_PLANETA_3);
42  m = round(20/F_R);                          % Limite inferior de los datos seleccionados
43  n = round(50/F_R);                          % Limite superior de los datos seleccionados
44
45  X_ENV_PLANETA_3 = zeros(A,B/2+1);       % Se crea un vector para guardar la amplitud de la TDF de la
        envolvente de la senal
46  F_ENV_PLANETA_3 = zeros(A,B/2+1);       % Se crea un vector para guardar la frecuencia de la TDF de la
        envolvente de la senal
47  Z_ENV_PLANETA_3 = zeros(A,B/2+1);       % Se crea un vector para guardar la fase de la TDF de la envolvente
        de la senal
48
49  X_ENV_MOD_PLANETA_3 = zeros(A,n-m);     % Se crea un vector para guardar la amplitud de la TDF de la
        envolvente cortada
50  F_ENV_MOD_PLANETA_3 = zeros(A,n-m);     % Se crea un vector para guardar la frecuencia de la TDF de la
        envolvente cortada
51
52  ACC_env = envelope(ACC_PLANETA_3,100,'peak');    % Se calcula la envolvente de la aceleracion
53
54  % Se calcula la TDF de la envolvente de las aceleraciones experimentales
55
```

```
for i = 1:A
    [a,b,c] = TDF(ACC_env(i,:),FS_PLANETA_3);                    %Se calcula la transformada de la envolvente
    F_ENV_PLANETA_3(i,:) = a;
    X_ENV_PLANETA_3(i,:) = b;
    Z_ENV_PLANETA_3(i,:) = c;
    X_ENV_MOD_PLANETA_3(i,:) = X_ENV_PLANETA_3(i,m:n-1)/max(X_ENV_PLANETA_3(i,m:n-1));     %Se normalizan y
        se guardan los datos en el vector
    F_ENV_MOD_PLANETA_3(i,:) = F_ENV_PLANETA_3(i,m:n-1);                              %Se guardan las
        frecuencias en el vector
end

%Se grafican las envolventes de las senales

% figure
% for i =1:A
%     hold on
%     plot(F_ENV_MOD_PLANETA_1(i,:), X_ENV_MOD_PLANETA_1(i,:))
%     xlim([min(F_ENV_MOD_PLANETA_3(i,:)),max(F_ENV_MOD_PLANETA_3(i,:))])
% end
% xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 14);
% ylabel('Amplitud  [m]','Interpreter', 'Latex','FontSize', 14)
% title('Espectro de Fourier de la envolvente','Interpreter','Latex','FontSize',14)

M = zeros(A,1);      %En este vector se guardan los maximos de la TDF de la envolvente
I = zeros(A,1);      %En este vector se guardan los indices de los maximos de la TDF de la envolvente

for i = 1:A
    [M(i),I(i)] = max(X_ENV_MOD_PLANETA_3(i,:));     %Se guardan tanto los valores como los indices
end

F_MAX_PLANETA_3 = zeros(1,A);            %Se genera un vector donde se van a guardar las frecuencias de
    giro
F_PLANETA_3_NORM = zeros(A,B/2+1);       %Se genera un vector donde se van a guardar las frecuecias
    normalizadas

%Se guardan las frecuencias de giro para cada medicion

N = 2;       %Numero de elementos que se toman ademas del central

for i = 1:A
    F_MAX_PLANETA_3(i) = sum(F_ENV_MOD_PLANETA_3(i,I(i)-N:I(i)+N).* X_ENV_MOD_PLANETA_3(i,I(i)-N:I(i)+N))/
        sum(X_ENV_MOD_PLANETA_3(i,I(i)-N:I(i)+N));
end

%Se normaliza dividiendo por la frecuencia de giro

for i = 1:A
    F_PLANETA_3_NORM(i,:) = F_PLANETA_3(i,:)/F_MAX_PLANETA_3(i);
end

%INTERPOLACION DE LOS DATOS

L = length(0:0.01:350);                 %Se calcula el largo para el nuevo vector de frecuencias que se
    quiere utilizar
F_MED_PLANETA_3_RES = zeros(A,L);            %Se crea un vector para guardar las frecuencias resampleadas
    despues de la interpolacion
X_MED_PLANETA_3_RES = zeros(A,L);            %Se crea un vector para guardar las amplitudes resampleadas
    despues de la interpolacion

%Se realiza la interpolacion para cada conjunto de datos

for i = 1:A
    F_MED_PLANETA_3_RES(i,:) = 0:0.01:350;
    X_MED_PLANETA_3_RES(i,:) = interp1(F_PLANETA_3_NORM(i,:),X_MED_PLANETA_3(i,:),F_MED_PLANETA_3_RES(i,:))
        ;
    X_MED_PLANETA_3_RES(i,:) = X_MED_PLANETA_3_RES(i,:)/max(X_MED_PLANETA_3_RES(i,:));
end

%Se grafica una comparacion de los datos inpoerlados y los sin interpolar

% figure
%
% for i = 1:A
%     hold on
%     plot(F_MED_PLANETA_3_RES(i,:),X_MED_PLANETA_3_RES(i,:));
%     title('(Default) Linear Interpolation');
% end
% xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 14);
% ylabel('Amplitud  [m]','Interpreter', 'Latex','FontSize', 14)
% title('Espectro de Fourier normalizado','Interpreter','Latex','FontSize',14)

%% SE GRAFICAN LOS DATOS EXPERIMENTALES ANTES DE LA MODIFICACION

% figure
%
% for i = 1:1
%     hold on
%     plot(F_MED_PLANETA_3_RES(i,:), X_MED_PLANETA_3_RES(i,:),'LineWidth',1)
% end
%
% xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 14);
% ylabel('Amplitud  [m]','Interpreter', 'Latex','FontSize', 14)
% title('Espectro de Fourier normalizado','Interpreter','Latex','FontSize',14)
% legend()

%% SE ELIMINAN LAS FRECUENCIAS INNECESMEDIAS
```

```
143
144 S1 = 1;
145 S2 = 20000;
146
147 X_MED_PLANETA_3_RES = X_MED_PLANETA_3_RES (:,S1:S2);
148
149 F_MED_PLANETA_3_RES = F_MED_PLANETA_3_RES (:,S1:S2);
150
151 %% SE GRAFICAN LOS DATOS EXPERIMENTALES DESPUES DEL FILTRO
152
153 figure
154
155 k = 1;
156
157 for i = k:k
158     hold on
159     plot(F_MED_PLANETA_3_RES(i,:), X_MED_PLANETA_3_RES(i,:)/max(X_MED_PLANETA_3_RES(i,:)))
160 end
161 xlim([0 60])
162 xlabel('Ordenes [S/U]', 'Interpreter', 'Latex', 'FontSize', 12);
163 ylabel('Amplitud [g]','Interpreter', 'Latex','FontSize', 12)
164 legend('Senal con filtro MED')
165 box on
166
167 %% SE GRAFICAN LOS DATOS EXPERIMENTALES ANTES DEL FILTRO
168
169 figure
170
171 k = 1;
172
173 for i = k:k
174     hold on
175     plot(F_PLANETA_3(i,:)/F_MAX_PLANETA_3(i), X_PLANETA_3(i,:)/max(X_PLANETA_3(i,:)),'LineWidth',0.1)
176 end
177 xlim([0 60])
178 xlabel('Ordenes [S/U]', 'Interpreter', 'Latex', 'FontSize', 12);
179 ylabel('Amplitud [g]','Interpreter', 'Latex','FontSize', 12)
180 legend('Senal sin filtro MED')
181 box on
182
183 %% SE GRAFICAN LOS DATOS EXPERIMENTALES ANTES Y DESPUES DEL FILTRO
184
185 figure
186
187 k = 1;
188
189 for i = k:k
190     hold on
191     plot(F_MED_PLANETA_3_RES(i,:), X_MED_PLANETA_3_RES(i,:),'LineWidth',0.1)
192     plot(F_PLANETA_3(i,:)/F_MAX_PLANETA_3(i), X_PLANETA_3(i,:)/max(X_PLANETA_3(i,:)),'LineWidth',0.1)
193 end
194 xlim([0 60])
195 xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 12);
196 ylabel('Amplitud [g]','Interpreter', 'Latex','FontSize', 12)
197 legend('Senal con filtro MED','Senal sin filtrar')
```

Apéndice E

FILTRO ARMED

Aplicación del filtro a datos experimentales [Filtro ARMED]

```
%% SE COMIENZA LIMPIANDO TODO

clc
clear all
close all

% SE EXTRAEN DATOS EXPERIMENTALES 3

load('ACC_PLANETA_3')

load('T_PLANETA_3')

FS_PLANETA_3 = 1/mean(diff(T_PLANETA_3(1,:))); %Se calcula la frecuencia de adquisicion.

%% SE APLICA EL FILTRO ARMED A LOS DATOS EXPERIMENTALES 3

[A,B] = size(ACC_PLANETA_3); %Se extrae el tamano de la matriz.

ACC_ARMED_PLANETA_3 = zeros(A,B);

for i = 1:A
    X =['Porcentaje de avance PLANETA (1 de 3): ',num2str(i*100/A),' %];     %Se determina el porcentaje de avance del proceso [%]
    disp(X)
    [ACC_ARMED_PLANETA_3(i,:)] = ARMED_FILTER(ACC_PLANETA_3(i,:));
end

%% SE COMPARAN LOS DATOS ANTES Y DESPUES DEL FILTRO

k = 5;  %Se selecciona solo un conjunto de datos.

figure
for i =k:k
    hold on
    plot(T_PLANETA_3(1,:), ACC_PLANETA_3(i,:),'LineWidth',1.5);
    plot(T_PLANETA_3(1,:), ACC_ARMED_PLANETA_3(i,:),'LineWidth',1.5);
end
xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 14);
xlim([4 4.1])
ylabel('Amplitud  [$m/s^2$]','Interpreter', 'Latex','FontSize', 14)
title('Espectro de Fourier de la envolvente','Interpreter','Latex','FontSize',14)
legend('Datos sin procesar','Datos con filtro ARMED')

%% SE GUARDAN LOS DATOS

T_ARMED_PLANETA_3 = T_PLANETA_3;

save('T_ARMED_PLANETA_3','T_ARMED_PLANETA_3')

save('ACC_ARMED_PLANETA_3','ACC_ARMED_PLANETA_3')
```

Función del filtro ARMED [Filtro ARMED]

```
function [ARMED_FILT] = ARMED_FILTER(signal)

sig=signal;      %Se toma la senal

clear y y_e A    %Se eliminan variables que pueden estar ocupadas

%Se itera a traves de los ordenes del filtro AR para obtener uno que
%tenga la menor kurtosis, para dejar solamente la senal estacionaria.

for order = 1:100


[A,E]=aryule(sig,order);                    %Se genera un filtro AR de orden n
y(:,order)=sig - filter(1,A,sig);           %Se usa la diferencia entre la senal y su filtro AR
kurt_y(:, order)=kurtosis(y(:,order));       %Se calcula la kurtosis para cada
%kurt_w(:, order)=kurtosis(w(:,order));      %Se calcula la kurtosis para cada
end


[~,index_y]=sort(kurt_y,'descend');          %Se guardan en orden descendente los indices de la kurtosis de
    las senales

AR_FILT_Y =y(:,index_y(1));                  %El filtro AR toma  la senal con mayor kurtosis para pasar al
    proceso de filtrado MED

%Se aplica un filtro MED (Minimum Enthropy Deconvolution), el cual permite
%filtrar la senal para tener la mayor Kurtosis posible.

[ar_med f_armed kurt_armed] = MED_FILTER(AR_FILT_Y,30,[],0.01,0);

ARMED_FILT =ar_med; %Se guarda la senal filtrada por ARMED

end
```

Gráficos TDF [Filtro ARMED]

```
%% SE COMIENZA LIMPIANDO TODO

clc
clear all
close all

%% SE EXTRAEN DATOS EXPERIMENTALES 3

load('ACC_ARMED_PLANETA_3')

load('T_ARMED_PLANETA_3')

load('ACC_PLANETA_3')

load('T_PLANETA_3')

FS_PLANETA_3 = 1/mean(diff(T_PLANETA_3(1,:)));
FS_ARMED_PLANETA_3 = 1/mean(diff(T_ARMED_PLANETA_3(1,:)));

%% TDF DE DATOS EXPERIMENTALES 3

[A,B] = size(ACC_PLANETA_3);

F_PLANETA_3 = zeros(A,B/2+1);
F_ARMED_PLANETA_3 = zeros(A,B/2+1);

X_PLANETA_3 = zeros(A,B/2+1);
X_ARMED_PLANETA_3 = zeros(A,B/2+1);

Z_PLANETA_3 = zeros(A,B/2+1);
Z_ARMED_PLANETA_3 = zeros(A,B/2+1);

for i = 1:A
    [F_PLANETA_3(i,:),X_PLANETA_3(i,:),Z_PLANETA_3(i,:)] = TDF(ACC_PLANETA_3(i,:),FS_PLANETA_3);
    [F_ARMED_PLANETA_3(i,:),X_ARMED_PLANETA_3(i,:),Z_ARMED_PLANETA_3(i,:)] = TDF(ACC_ARMED_PLANETA_3(i,:),
        FS_ARMED_PLANETA_3);

end

%% ANALISIS DE ENVOLVENTE DEL PLANETA 3

F_R = FS_PLANETA_3 / length(ACC_PLANETA_3);
m = round(20/F_R);                          %Limite inferior de los datos seleccionados
n = round(50/F_R);                          %Limite superior de los datos seleccionados

X_ENV_PLANETA_3 = zeros(A,B/2+1);       %Se crea un vector para guardar la amplitud de la TDF de la
    envolvente de la senal
F_ENV_PLANETA_3 = zeros(A,B/2+1);       %Se crea un vector para guardar la frecuencia de la TDF de la
    envolvente de la senal
Z_ENV_PLANETA_3 = zeros(A,B/2+1);       %Se crea un vector para guardar la fase de la TDF de la envolvente
    de la senal

X_ENV_MOD_PLANETA_3 = zeros(A,n-m);     %Se crea un vector para guardar la amplitud de la TDF de la
    envolvente cortada
```

```
F_ENV_MOD_PLANETA_3 = zeros(A,n-m);    % Se crea un vector para guardar la frecuencia de la TDF de la envolvente cortada

ACC_env = envelope(ACC_PLANETA_3,100,'peak');    % Se calcula la envolvente de la aceleracion

% Se calcula la TDF de la envolvente de las aceleraciones experimentales

for i = 1:A
    [a,b,c] = TDF(ACC_env(i,:),FS_PLANETA_3);                % Se calcula la transformada de la envolvente
    F_ENV_PLANETA_3(i,:) = a;
    X_ENV_PLANETA_3(i,:) = b;
    Z_ENV_PLANETA_3(i,:) = c;
    X_ENV_MOD_PLANETA_3(i,:) = X_ENV_PLANETA_3(i,m:n-1)/max(X_ENV_PLANETA_3(i,m:n-1));     % Se normalizan y se guardan los datos en el vector
    F_ENV_MOD_PLANETA_3(i,:) = F_ENV_PLANETA_3(i,m:n-1);                                    % Se guardan las frecuencias en el vector
end

%Se grafican las envolventes de las senales

% figure
% for i =1:A
%     hold on
%     plot(F_ENV_MOD_PLANETA_1(i,:), X_ENV_MOD_PLANETA_1(i,:))
%     xlim([min(F_ENV_MOD_PLANETA_3(i,:)),max(F_ENV_MOD_PLANETA_3(i,:))])
% end
% xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 14);
% ylabel('Amplitud  [m]','Interpreter', 'Latex','FontSize', 14)
% title('Espectro de Fourier de la envolvente','Interpreter','Latex','FontSize',14)

M = zeros(A,1);      % En este vector se guardan los maximos de la TDF de la envolvente
I = zeros(A,1);      % En este vector se guardan los indices de los maximos de la TDF de la envolvente

for i = 1:A
    [M(i),I(i)] = max(X_ENV_MOD_PLANETA_3(i,:));     % Se guardan tanto los valores como los indices
end

F_MAX_PLANETA_3 = zeros(1,A);            % Se genera un vector donde se van a guardar las frecuencias de giro
F_PLANETA_3_NORM = zeros(A,B/2+1);       % Se genera un vector donde se van a guardar las frecuecias normalizadas

% Se guardan las frecuencias de giro para cada medicion

N = 2;       % Numero de elementos que se toman ademas del central

for i = 1:A
    F_MAX_PLANETA_3(i) = sum(F_ENV_MOD_PLANETA_3(i,I(i)-N:I(i)+N).* X_ENV_MOD_PLANETA_3(i,I(i)-N:I(i)+N))/sum(X_ENV_MOD_PLANETA_3(i,I(i)-N:I(i)+N));
end

% Se normaliza dividiendo por la frecuencia de giro

for i = 1:A
    F_PLANETA_3_NORM(i,:) = F_PLANETA_3(i,:)/F_MAX_PLANETA_3(i);
end

% INTERPOLACION DE LOS DATOS

L = length(0:0.01:350);                  % Se calcula el largo para el nuevo vector de frecuencias que se quiere utilizar
F_ARMED_PLANETA_3_RES = zeros(A,L);            % Se crea un vector para guardar las frecuencias resampleadas despues de la interpolacion
X_ARMED_PLANETA_3_RES = zeros(A,L);            % Se crea un vector para guardar las amplitudes resampleadas despues de la interpolacion

% Se realiza la interpolacion para cada conjunto de datos

for i = 1:A
    F_ARMED_PLANETA_3_RES(i,:) = 0:0.01:350;
    X_ARMED_PLANETA_3_RES(i,:) = interp1(F_PLANETA_3_NORM(i,:),X_ARMED_PLANETA_3(i,:),F_ARMED_PLANETA_3_RES(i,:));
    X_ARMED_PLANETA_3_RES(i,:) = X_ARMED_PLANETA_3_RES(i,:)/max(X_ARMED_PLANETA_3_RES(i,:));
end

% Se grafica una comparacion de los datos inpoerlados y los sin interpolar

% figure
%
% for i = 1:A
%     hold on
%     plot(F_ARMED_PLANETA_3_RES(i,:),X_ARMED_PLANETA_3_RES(i,:));
%     title('(Default) Linear Interpolation');
% end
% xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 14);
% ylabel('Amplitud  [m]','Interpreter', 'Latex','FontSize', 14)
% title('Espectro de Fourier normalizado','Interpreter','Latex','FontSize',14)

%% SE GRAFICAN LOS DATOS EXPERIMENTALES ANTES DE LA MODIFICACION

% figure
%
% for i = 1:1
%     hold on
%     plot(F_ARMED_PLANETA_3_RES(i,:), X_ARMED_PLANETA_3_RES(i,:),'LineWidth',1)
% end
```

```
%
% xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 14);
% ylabel('Amplitud  [m]','Interpreter', 'Latex','FontSize', 14)
% title('Espectro de Fourier normalizado','Interpreter','Latex','FontSize',14)
% legend()

%% SE ELIMINAN LAS FRECUENCIAS INNECESARMEDIAS

S1 = 1;
S2 = 20000;

X_ARMED_PLANETA_3_RES = X_ARMED_PLANETA_3_RES (:,S1:S2);

F_ARMED_PLANETA_3_RES = F_ARMED_PLANETA_3_RES (:,S1:S2);


%% SE GRAFICAN LOS DATOS EXPERIMENTALES DESPUES DEL FILTRO

figure

k = 1;

for i = k:k
    hold on
    plot(F_ARMED_PLANETA_3_RES(i,:), X_ARMED_PLANETA_3_RES(i,:),'LineWidth',1.5)
end
xlim([0 60])
xlabel('Ordenes [S/U]', 'Interpreter', 'Latex', 'FontSize', 12);
ylabel('Amplitud  [g]','Interpreter', 'Latex','FontSize', 12)
legend('Senal con filtro ARMED')
box on

%% SE GRAFICAN LOS DATOS EXPERIMENTALES ANTES DEL FILTRO

figure

k = 1;

for i = k:k
    hold on
    plot(F_PLANETA_3(i,:)/F_MAX_PLANETA_3(i), X_PLANETA_3(i,:)/max(X_PLANETA_3(i,:)),'LineWidth',0.1)
end
xlim([0 60])
xlabel('Ordenes [S/U]', 'Interpreter', 'Latex', 'FontSize', 12);
ylabel('Amplitud  [g]','Interpreter', 'Latex','FontSize', 12)
legend('Senal sin filtro ARMED')
box on

%% SE GRAFICAN LOS DATOS EXPERIMENTALES ANTES Y DESPUES DEL FILTRO

figure

k = 1;

for i = k:k
    hold on
    plot(F_ARMED_PLANETA_3_RES(i,:), X_ARMED_PLANETA_3_RES(i,:),'LineWidth',0.1)
    plot(F_PLANETA_3(i,:)/F_MAX_PLANETA_3(i), X_PLANETA_3(i,:)/max(X_PLANETA_3(i,:)),'LineWidth',0.1)
end
xlim([0 60])
xlabel('Frecuencia [Hz]', 'Interpreter', 'Latex', 'FontSize', 12);
ylabel('Amplitud  [m]','Interpreter', 'Latex','FontSize', 12)
legend('Senal con filtro ARMED')
```

Apéndice F

TDF NORMALIZADAS

Caso Normal [TDF]

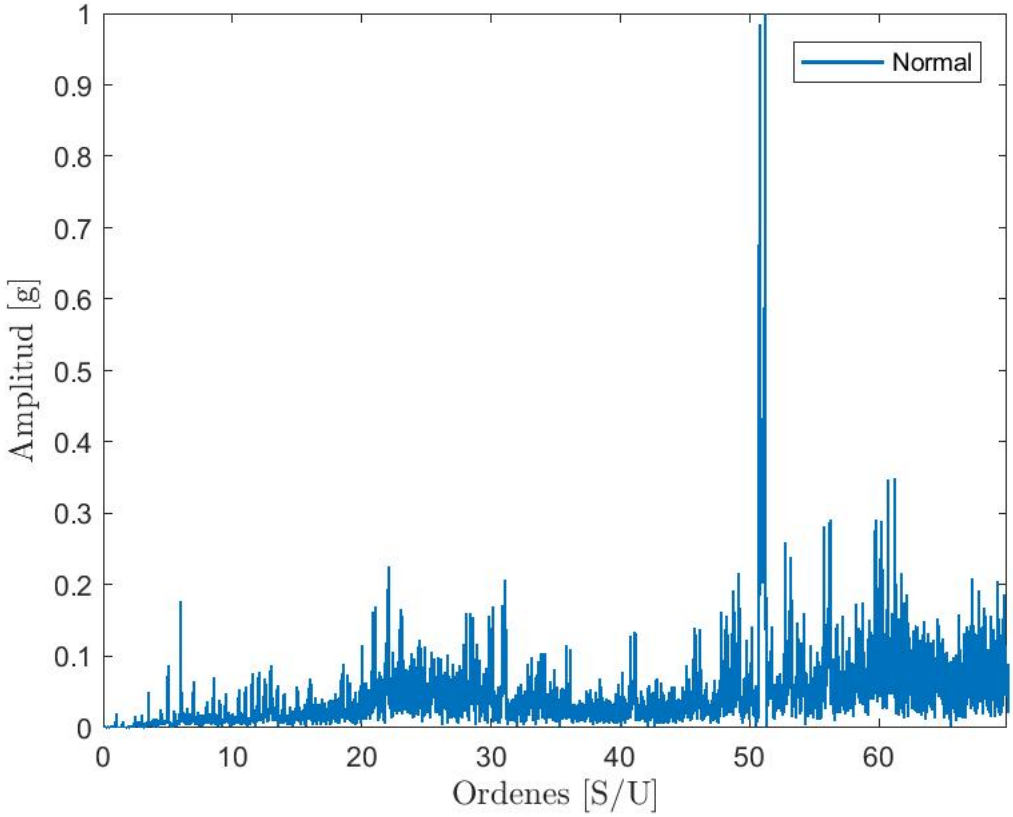

Figura F.1: Caso normal [TDF]

CASO PLANETAS [TDF]

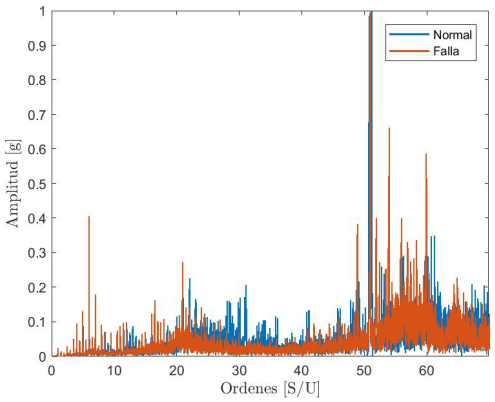

Figura F.2: Comparación PLANETA - Nivel 1 [TDF]

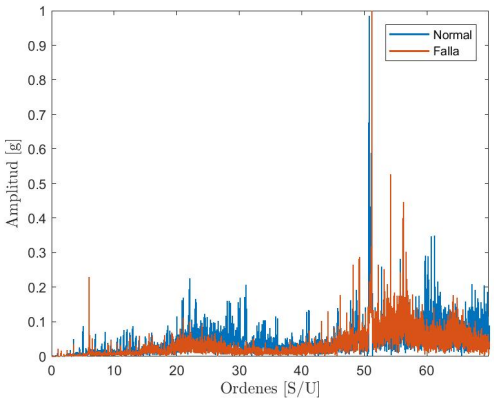

Figura F.3: Comparación PLANETA - Nivel 2 [TDF]

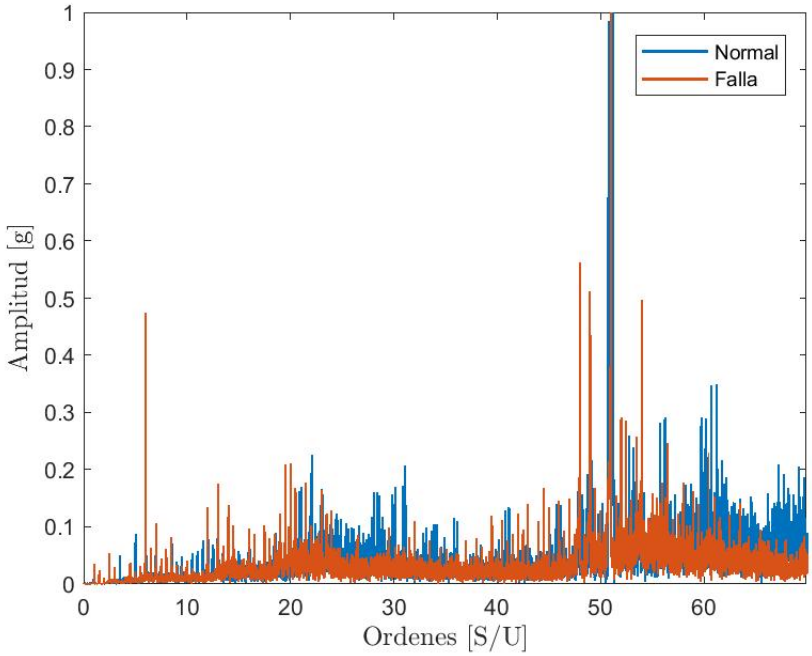

Figura F.4: Comparación PLANETA - Nivel 3 [TDF]

CASO ARO [TDF]

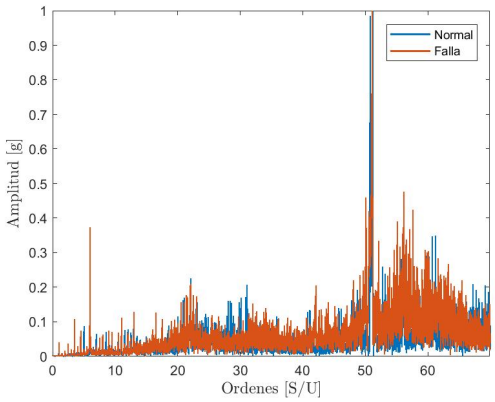

Figura F.5: Comparación ARO - Nivel 1 [TDF]

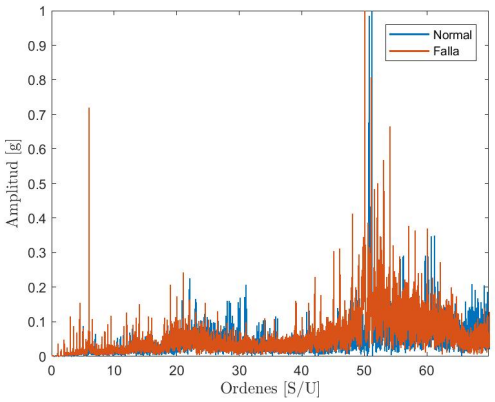

Figura F.6: Comparación ARO - Nivel 2 [TDF]

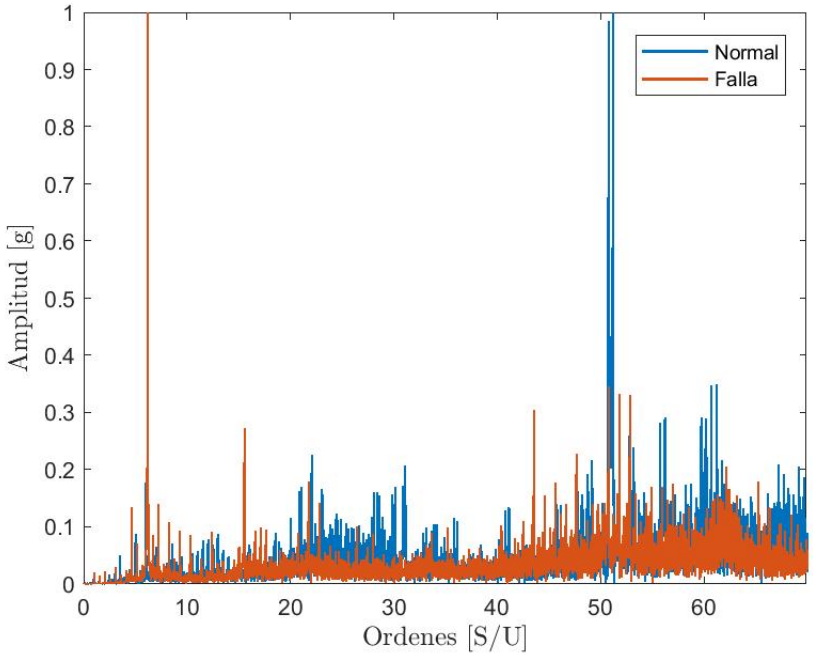

Figura F.7: Comparación ARO - Nivel 3 [TDF]

CASO SOL [TDF]

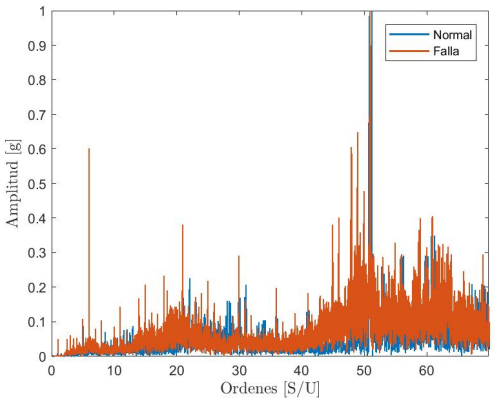

Figura F.8: Comparación SOL - Nivel 1 [TDF]

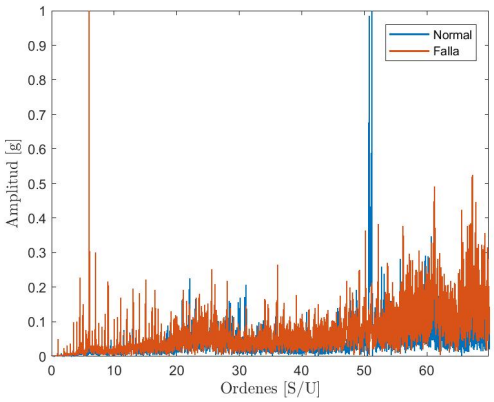

Figura F.9: Comparación SOL - Nivel 2 [TDF]

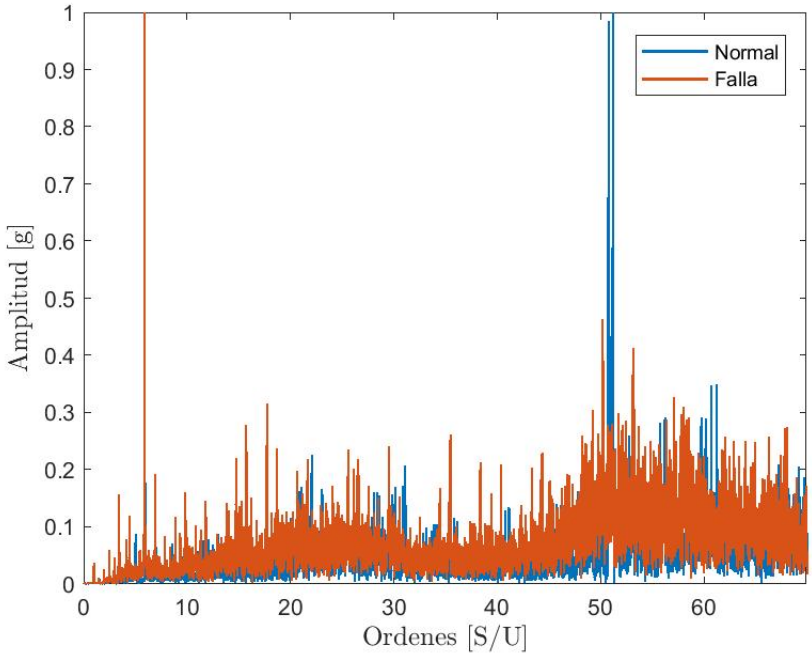

Figura F.10: Comparación SOL - Nivel 3 [TDF]

COMPARACIONES [TDF]

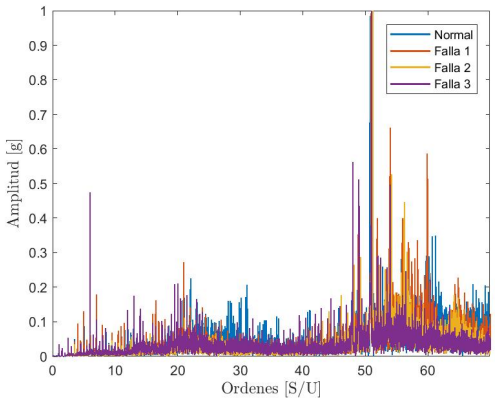

Figura F.11: Comparación PLANETA - Todos los niveles [TDF]

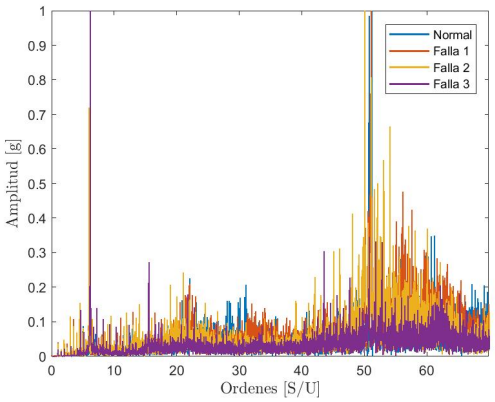

Figura F.12: Comparación SOL - Todos los niveles [TDF]

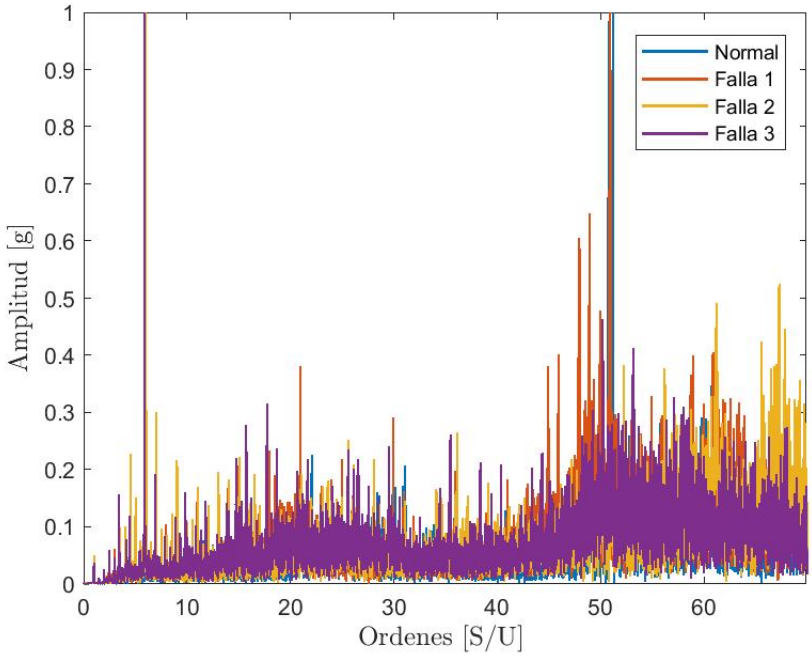

Figura F.13: Comparación SOL - Todos los niveles [TDF]

Apéndice G

FILTRO AUTOREGRESIVO

Caso Normal [AR]

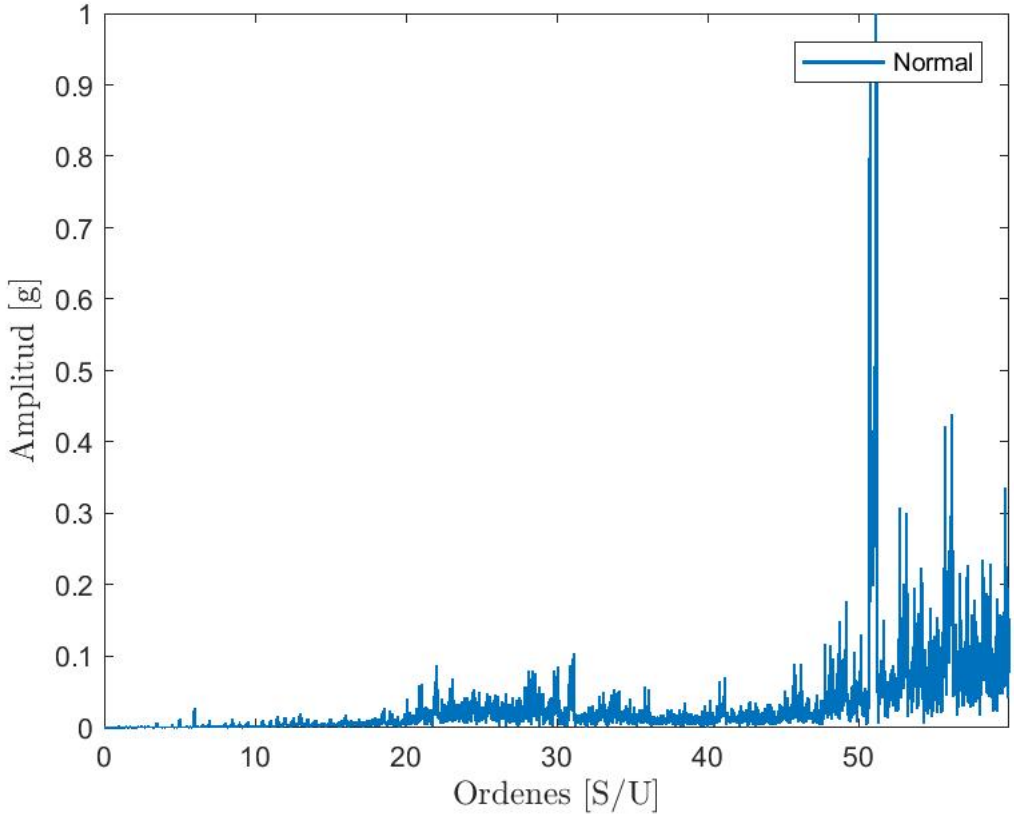

Figura G.1: Caso normal [AR]

CASO PLANETAS [AR]

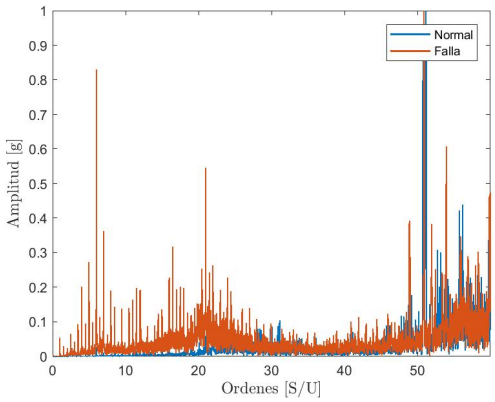

Figura G.2: Comparación PLANETA - Nivel 1 [AR]

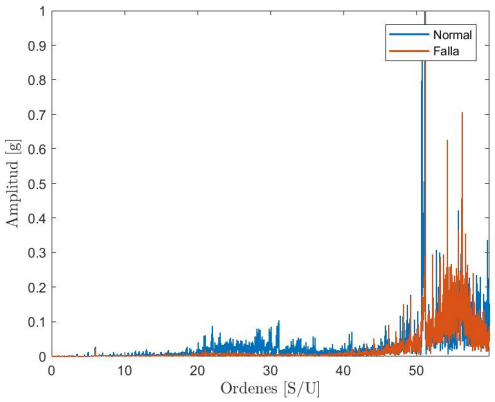

Figura G.3: Comparación PLANETA - Nivel 2 [AR]

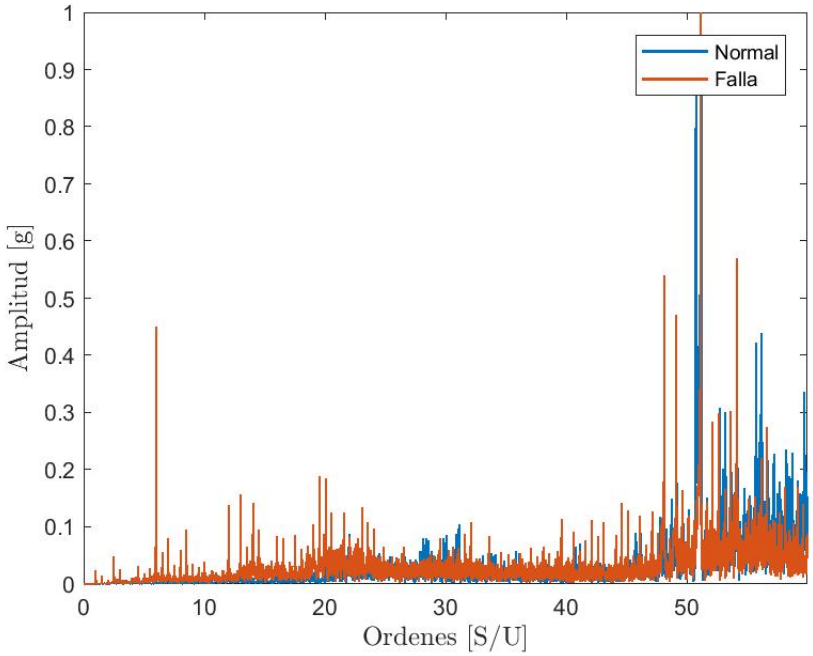

Figura G.4: Comparación PLANETA - Nivel 3 [AR]

CASO ARO [AR]

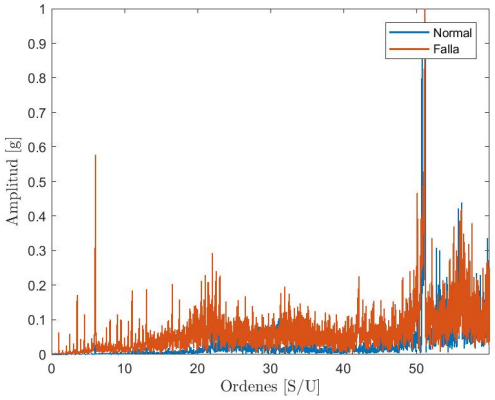

Figura G.5: Comparación ARO - Nivel 1 [AR]

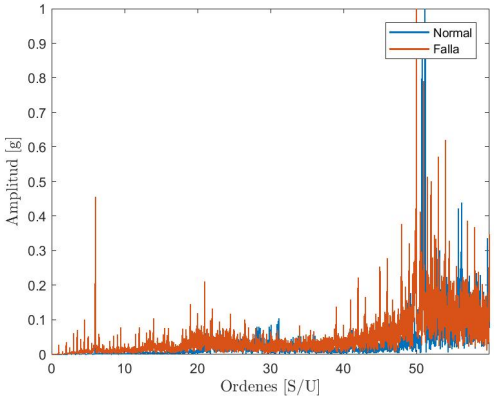

Figura G.6: Comparación ARO - Nivel 2 [AR]

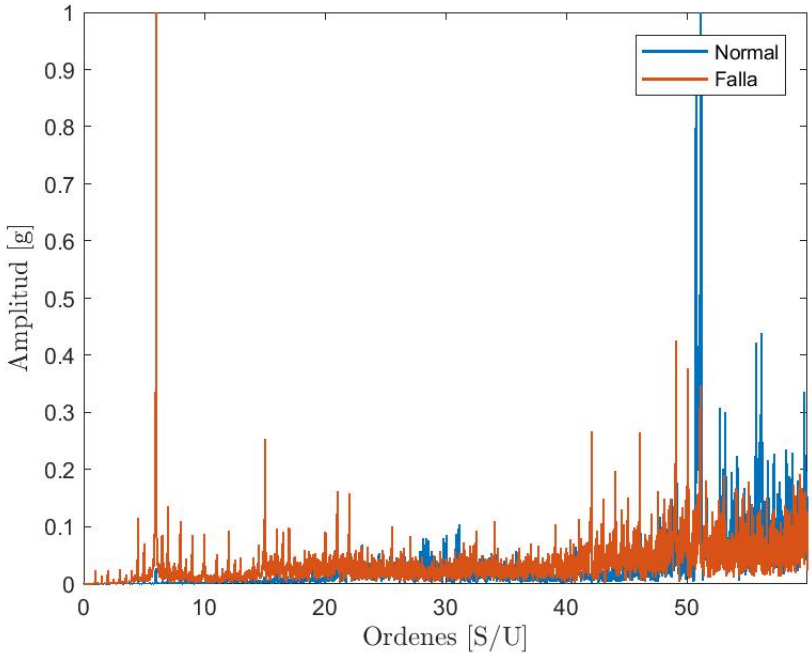

Figura G.7: Comparación ARO - Nivel 3 [AR]

CASO SOL [AR]

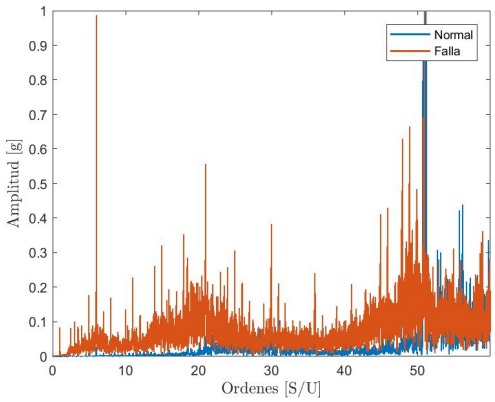

Figura G.8: Comparación SOL - Nivel 1 [AR]

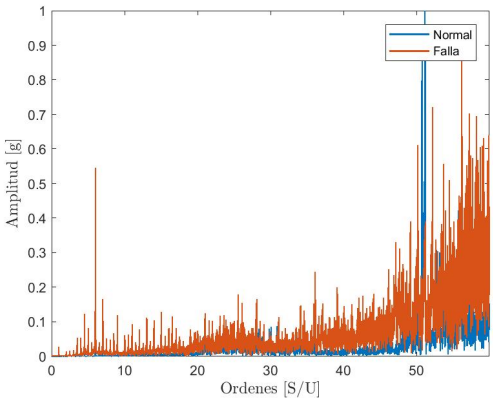

Figura G.9: Comparación SOL - Nivel 2 [AR]

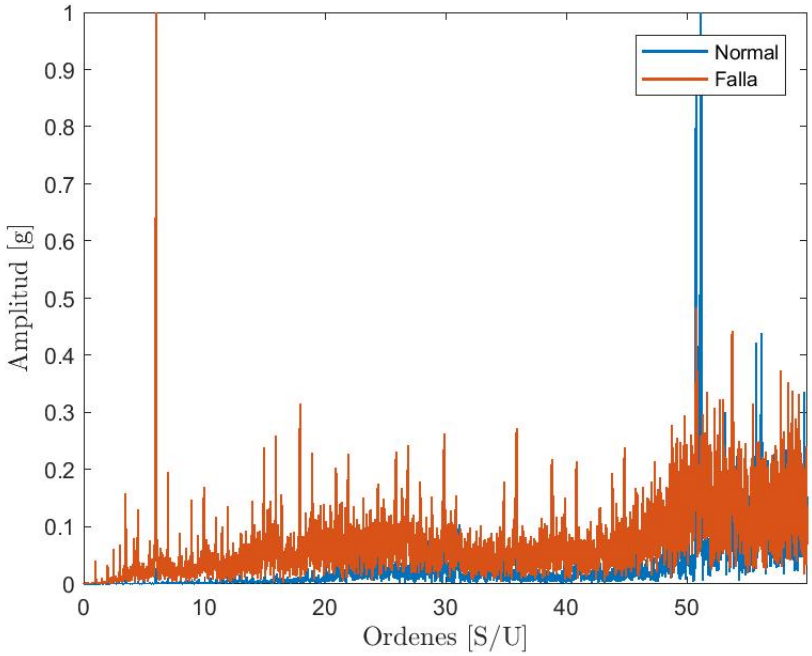

Figura G.10: Comparación SOL - Nivel 3 [AR]

COMPARACIONES [AR]

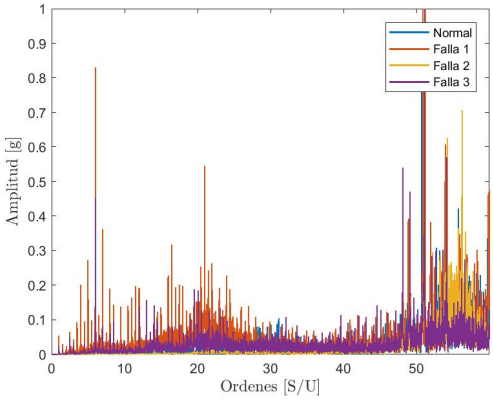

Figura G.11: Comparación PLANETA - Todos los niveles [AR]

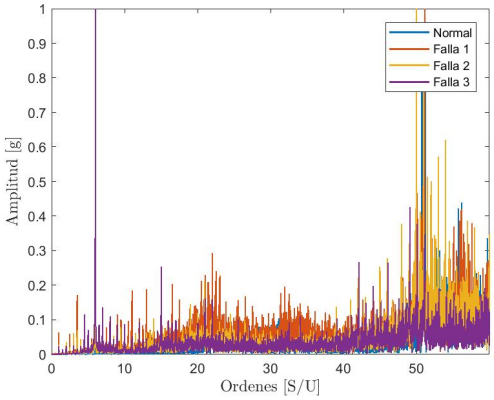

Figura G.12: Comparación SOL - Todos los niveles [AR]

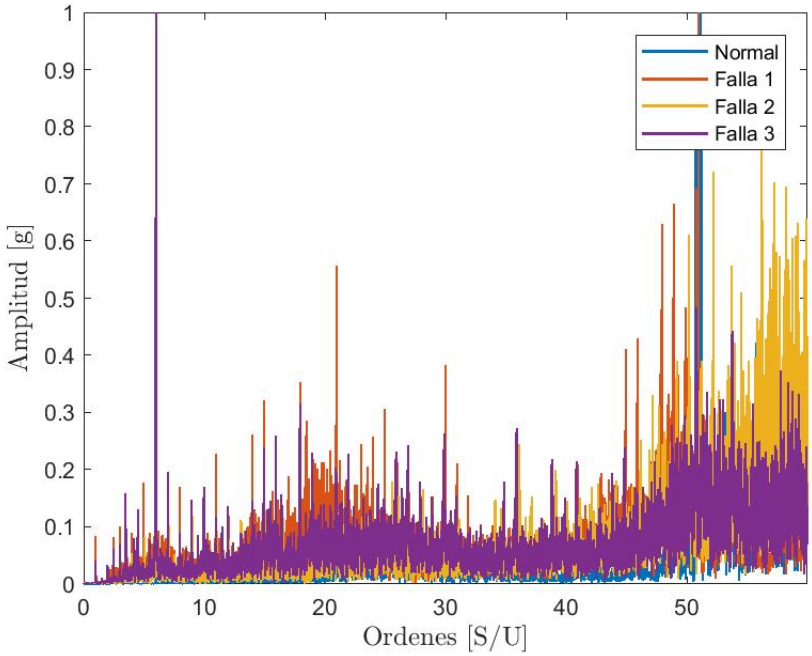

Figura G.13: Comparación SOL - Todos los niveles [AR]

Apéndice H

FILTRO DECONVOLUCIÓN DE MÍNIMA ENTROPÍA

Caso Normal [MED]

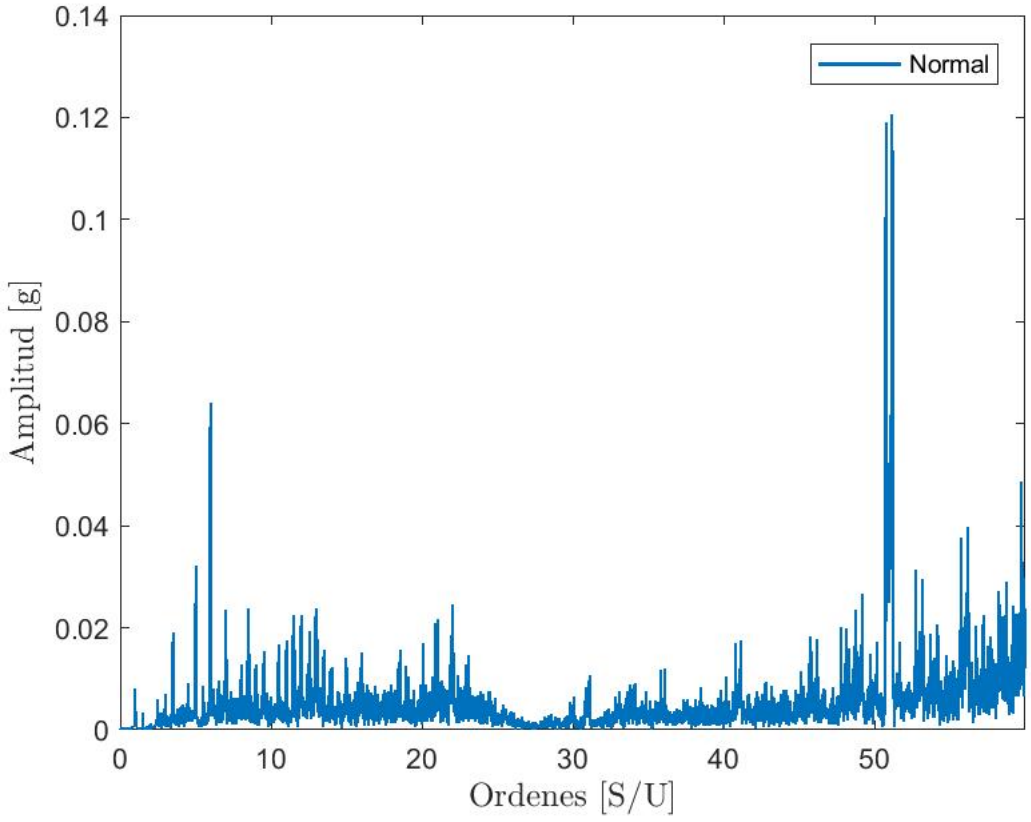

Figura H.1: Caso normal [MED]

CASO PLANETAS [MED]

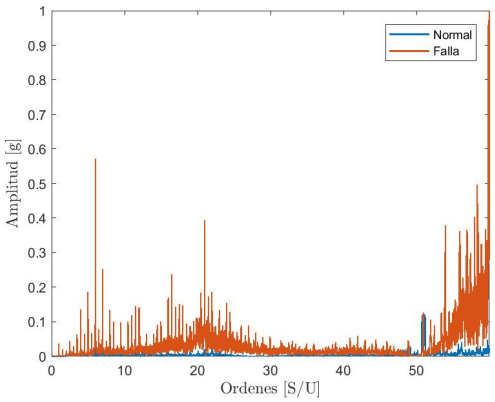

Figura H.2: Comparación PLANETA - Nivel 1 [MED]

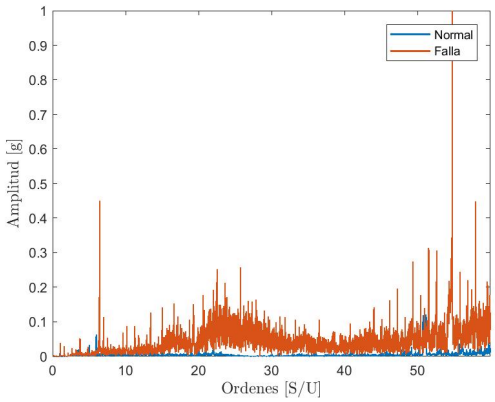

Figura H.3: Comparación PLANETA - Nivel 2 [MED]

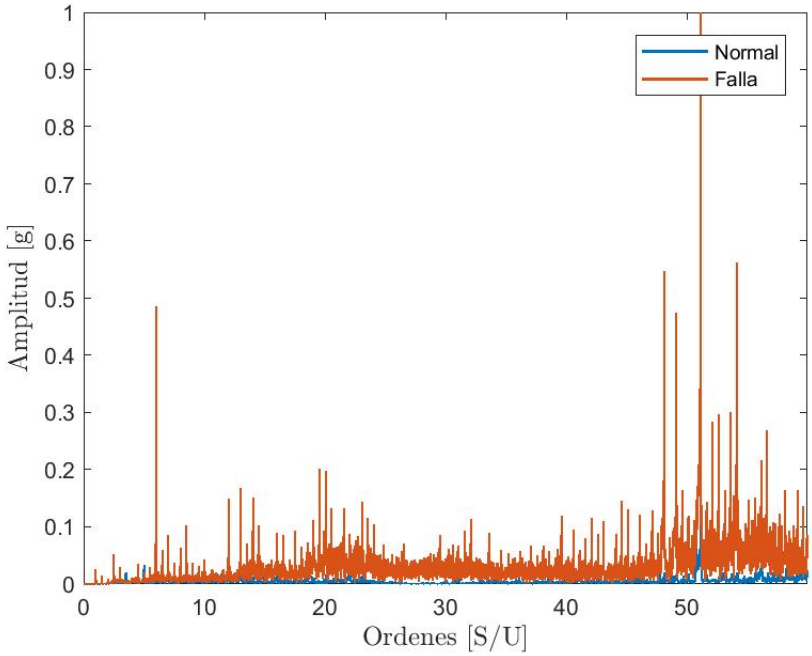

Figura H.4: Comparación PLANETA - Nivel 3 [MED]

CASO ARO [MED]

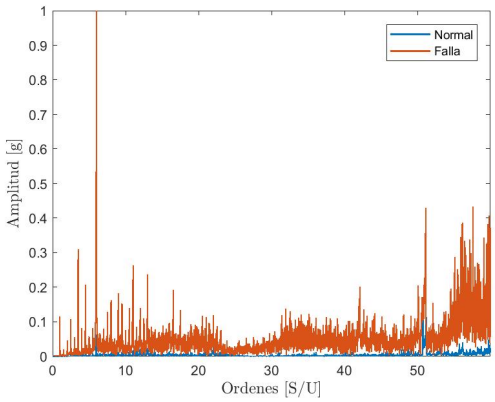

Figura H.5: Comparación ARO - Nivel 1 [MED]

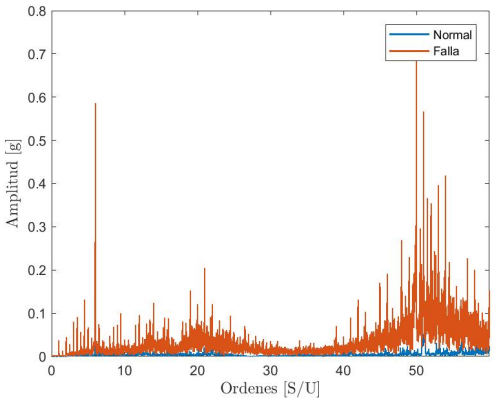

Figura H.6: Comparación ARO - Nivel 2 [MED]

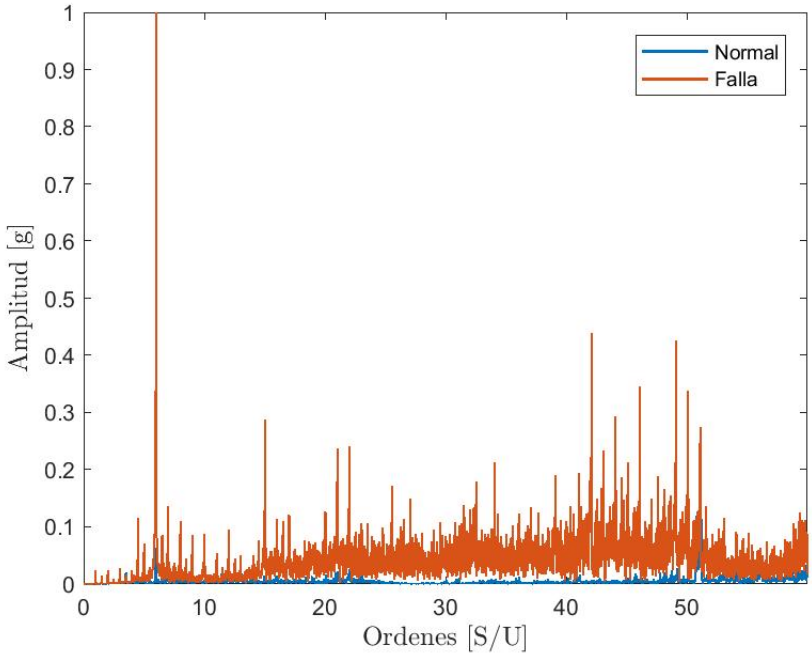

Figura H.7: Comparación ARO - Nivel 3 [MED]

CASO SOL [MED]

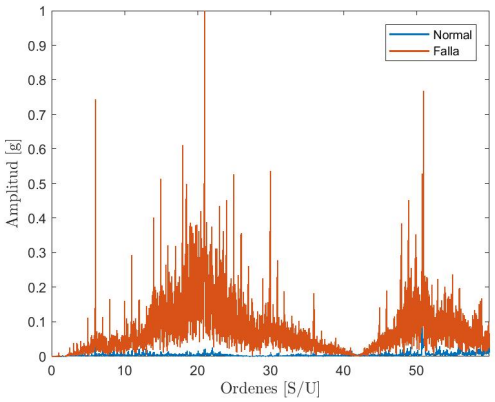

Figura H.8: Comparación SOL - Nivel 1 [MED]

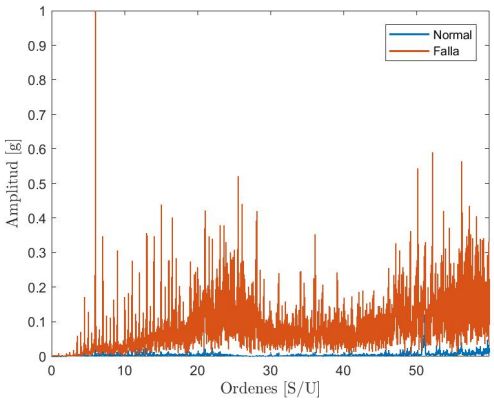

Figura H.9: Comparación SOL - Nivel 2 [MED]

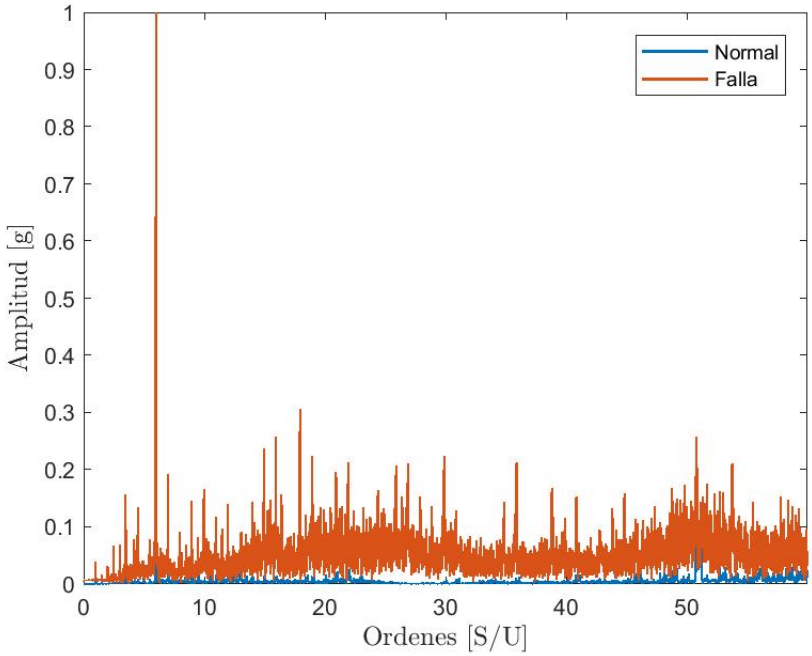

Figura H.10: Comparación SOL - Nivel 3 [MED]

COMPARACIONES [MED]

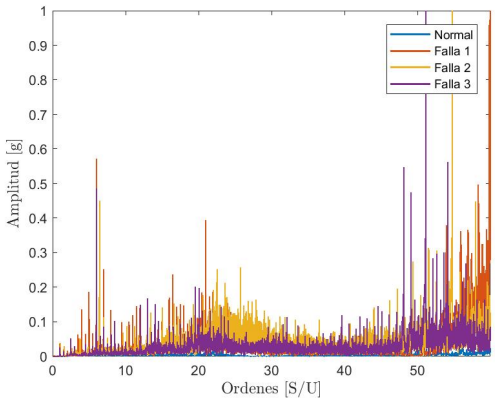

Figura H.11: Comparación PLANETA - Todos los niveles [MED]

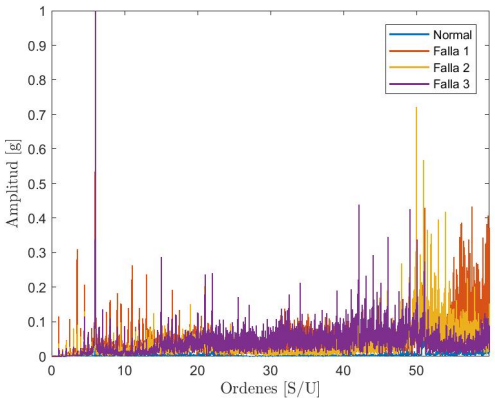

Figura H.12: Comparación SOL - Todos los niveles [MED]

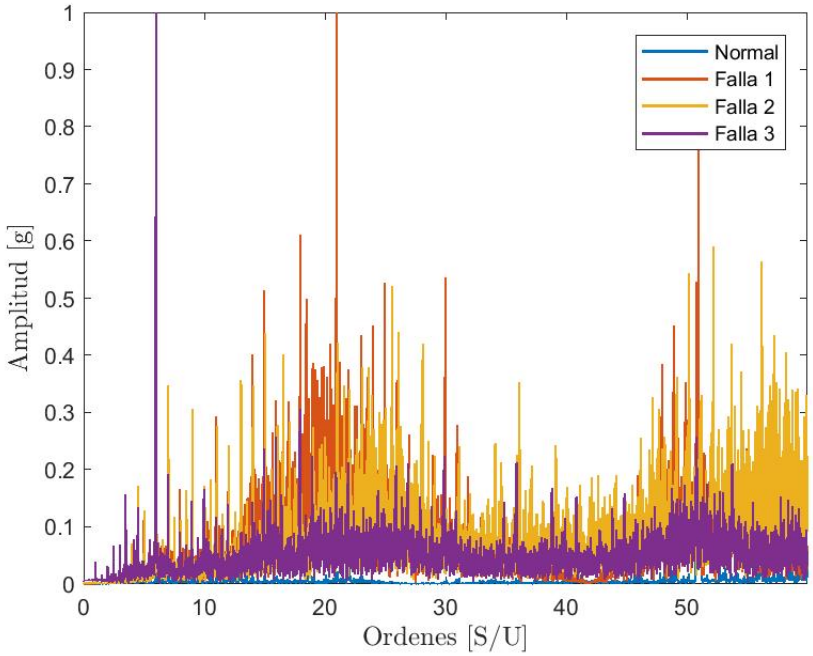

Figura H.13: Comparación SOL - Todos los niveles [MED]

Apéndice I

FILTRO ARMED

Caso Normal [ARMED]

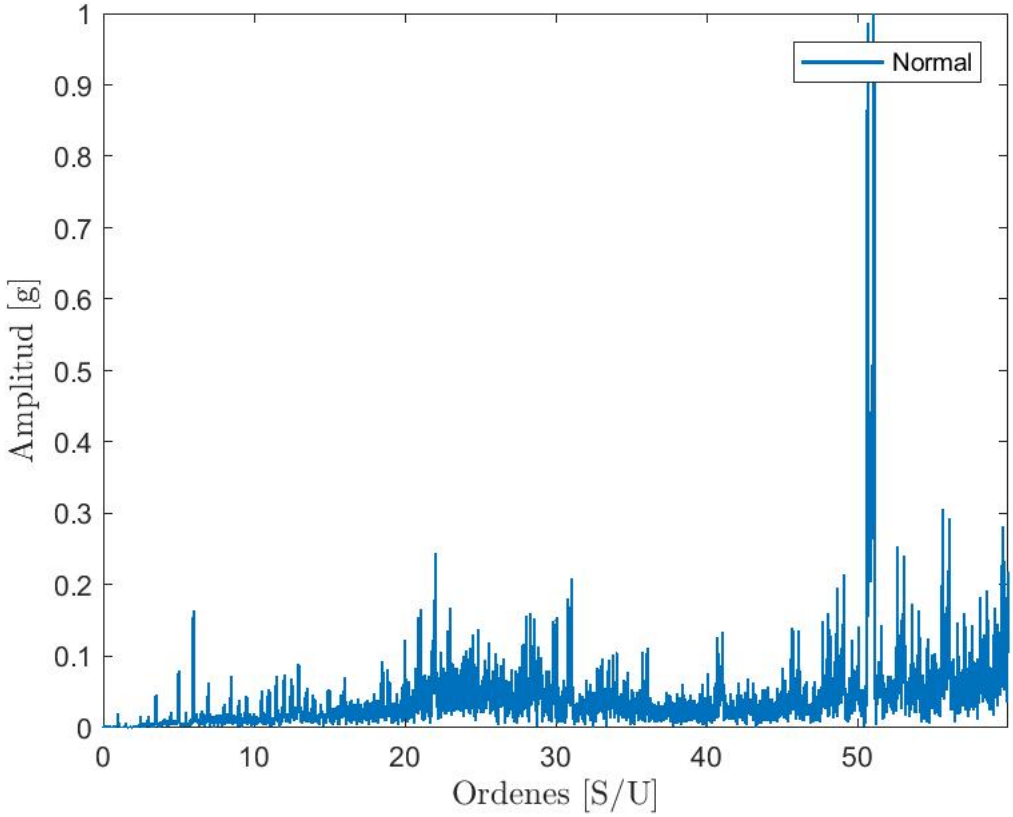

Figura I.1: Caso normal [ARMED]

CASO PLANETAS [ARMED]

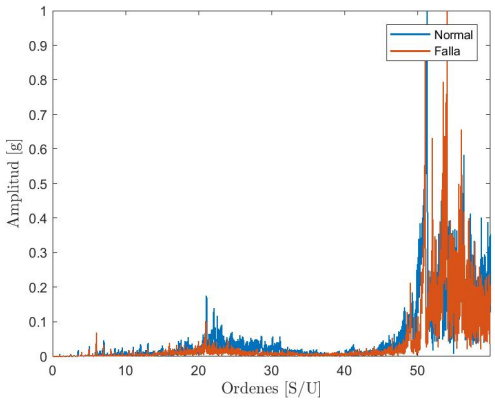

Figura I.2: Comparación PLANETA - Nivel 1 [ARMED]

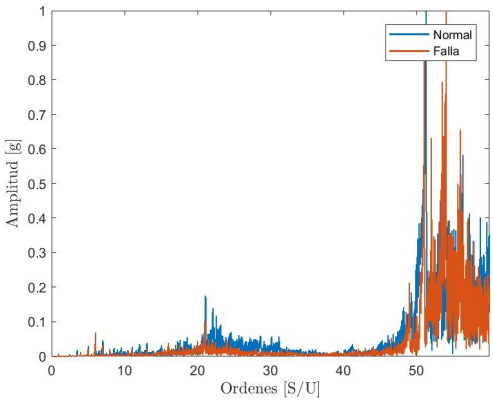

Figura I.3: Comparación PLANETA - Nivel 2 [ARMED]

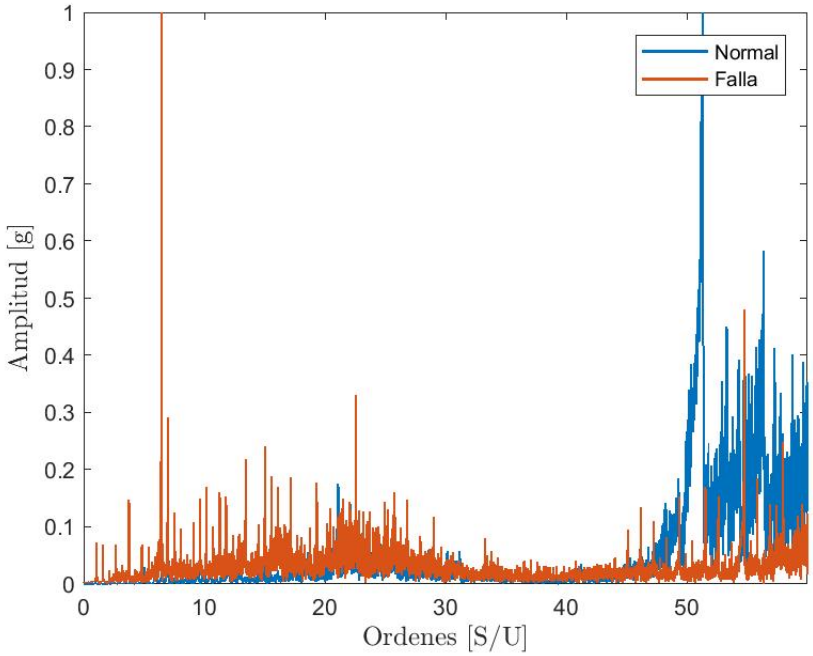

Figura I.4: Comparación PLANETA - Nivel 3 [ARMED]

CASO ARO [ARMED]

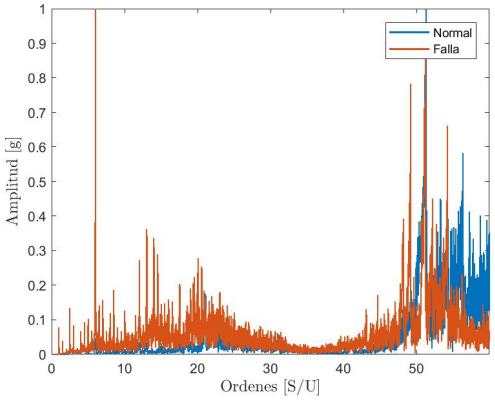

Figura I.5: Comparación ARO - Nivel 1 [ARMED]

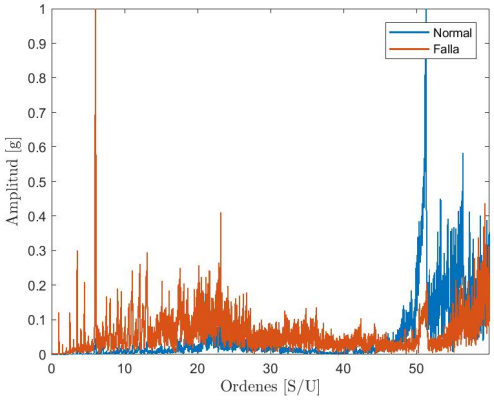

Figura I.6: Comparación ARO - Nivel 2 [ARMED]

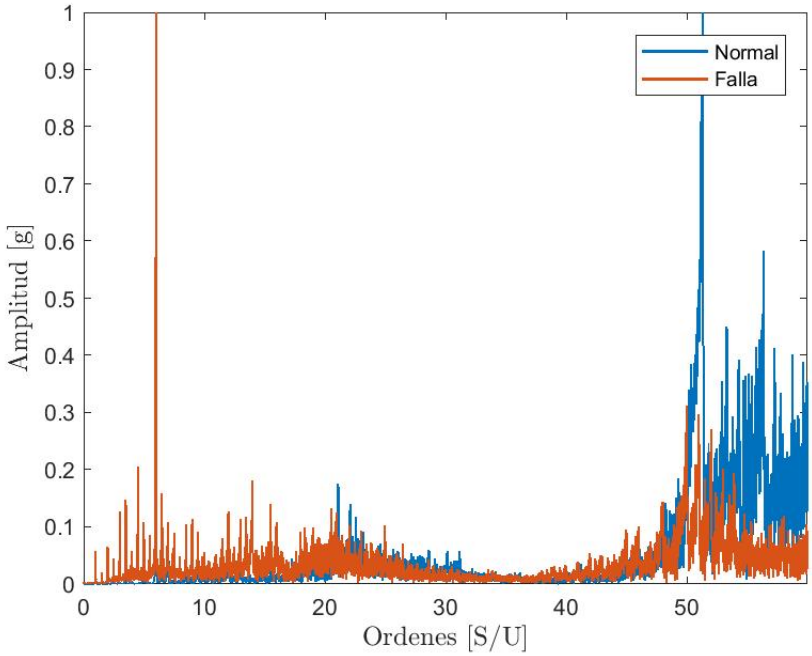

Figura I.7: Comparación ARO - Nivel 3 [ARMED]

CASO SOL [ARMED]

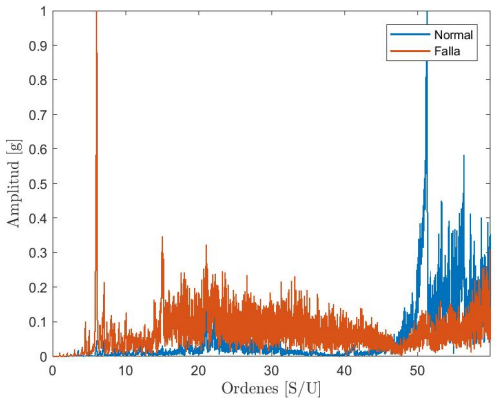

Figura I.8: Comparación SOL - Nivel 1 [ARMED]

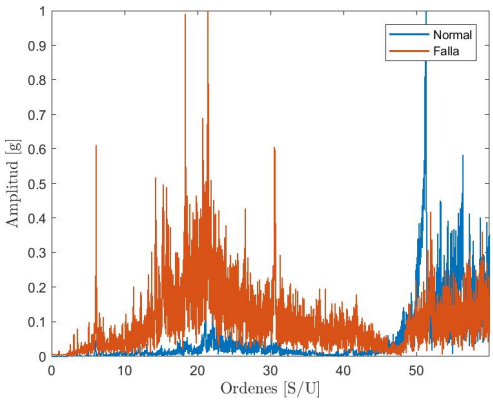

Figura I.9: Comparación SOL - Nivel 2 [ARMED]

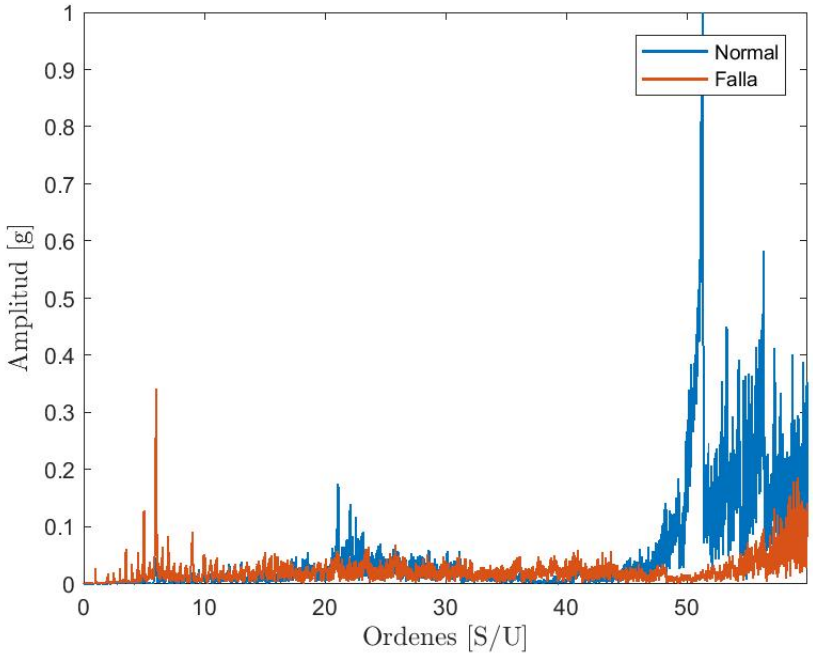

Figura I.10: Comparación SOL - Nivel 3 [ARMED]

COMPARACIONES [ARMED]

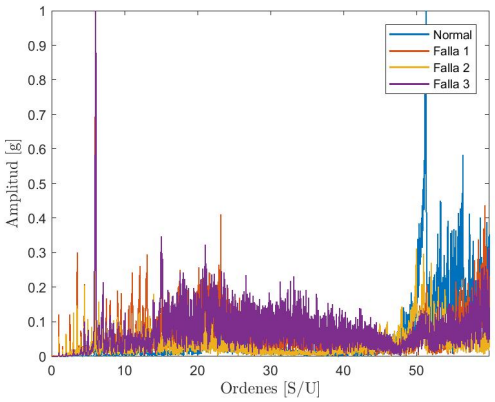

Figura I.11: Comparación PLANETA - Todos los niveles [ARMED]

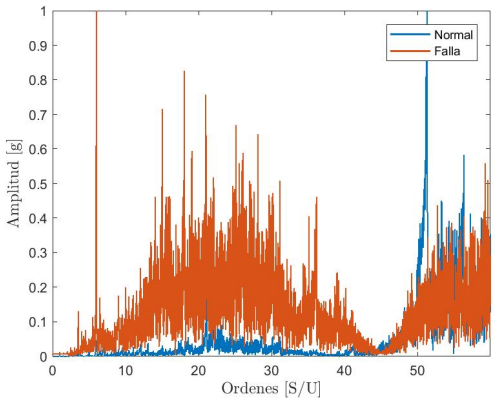

Figura I.12: Comparación SOL - Todos los niveles [ARMED]

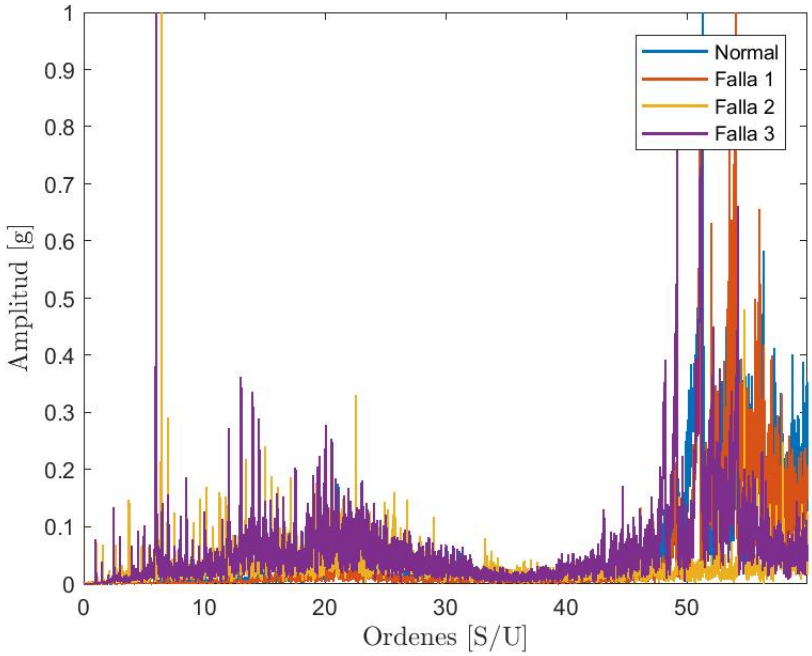

Figura I.13: Comparación SOL - Todos los niveles [ARMED]

Apéndice J

FILTRO ENV

Caso Normal [ENV]

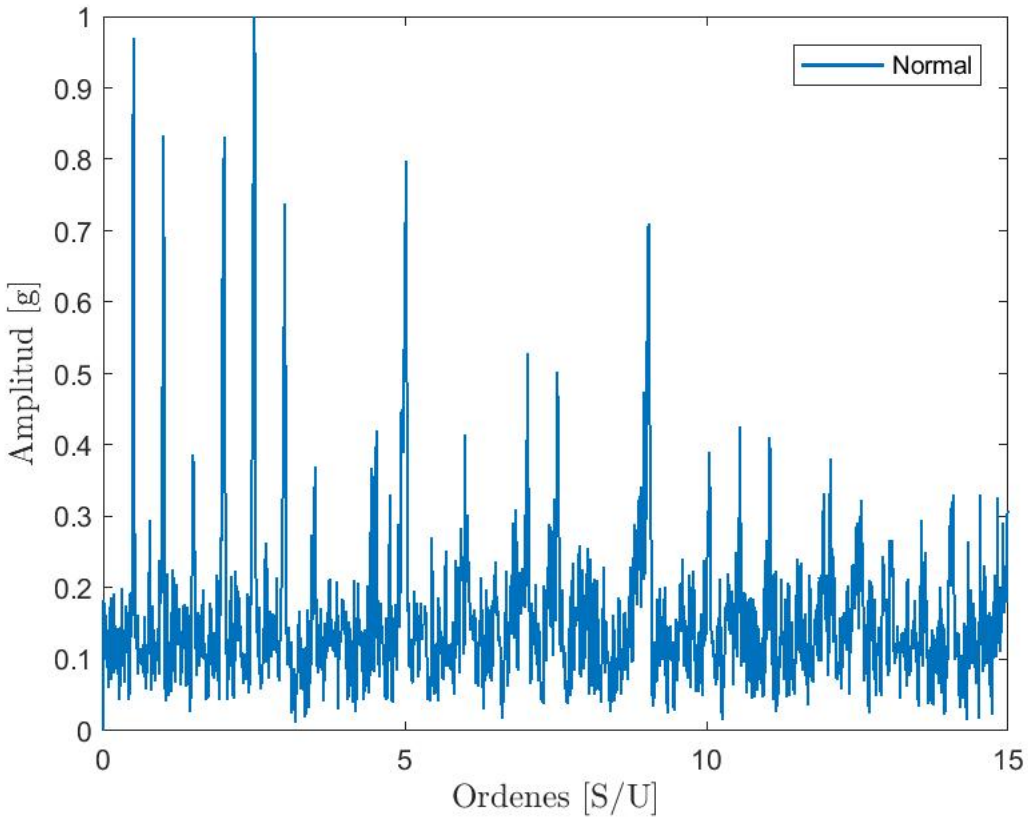

Figura J.1: Caso normal [ENV]

CASO PLANETAS [ENV]

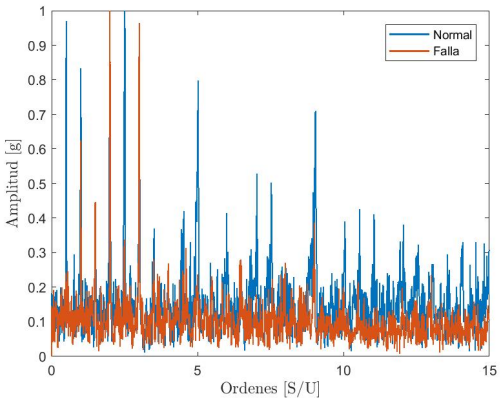

Figura J.2: Comparación PLANETA - Nivel 1 [ENV]

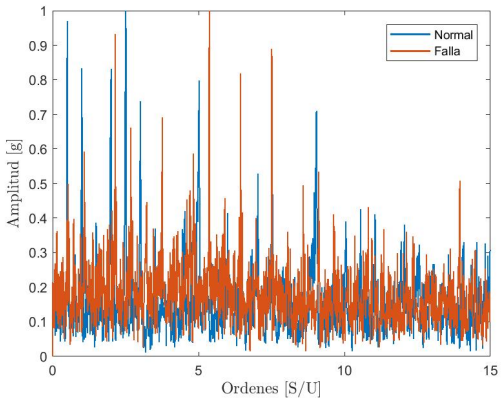

Figura J.3: Comparación PLANETA - Nivel 2 [ENV]

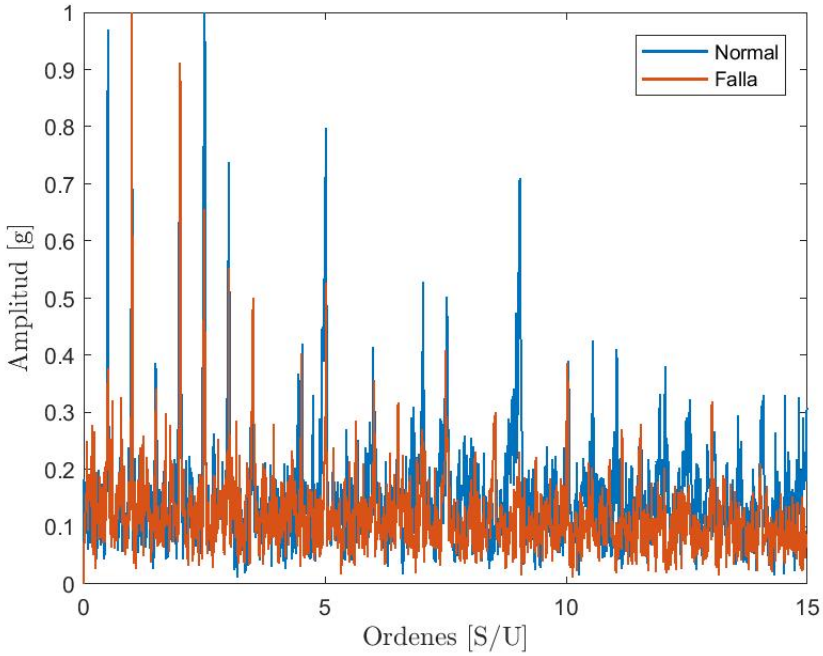

Figura J.4: Comparación PLANETA - Nivel 3 [ENV]

CASO ARO [ENV]

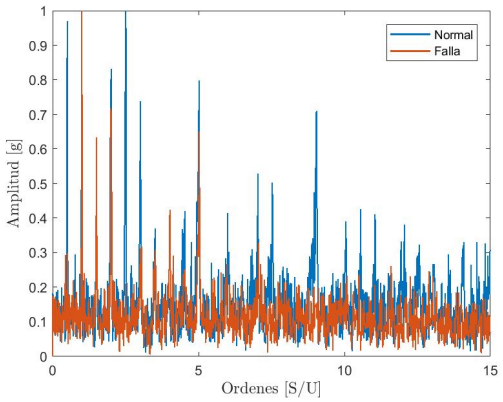

Figura J.5: Comparación ARO - Nivel 1 [ENV]

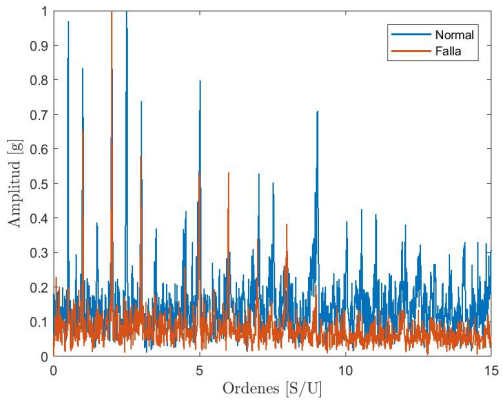

Figura J.6: Comparación ARO - Nivel 2 [ENV]

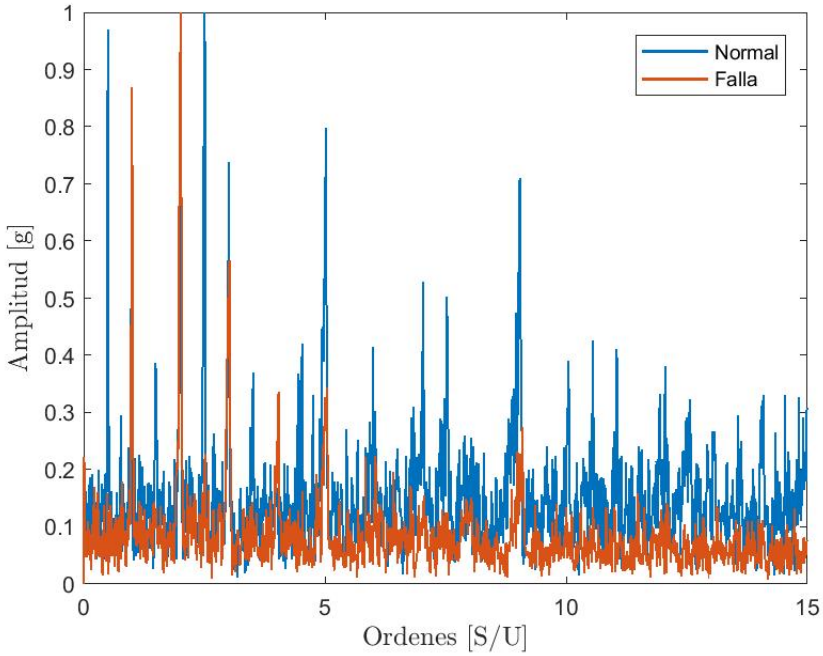

Figura J.7: Comparación ARO - Nivel 3 [ENV]

CASO SOL [ENV]

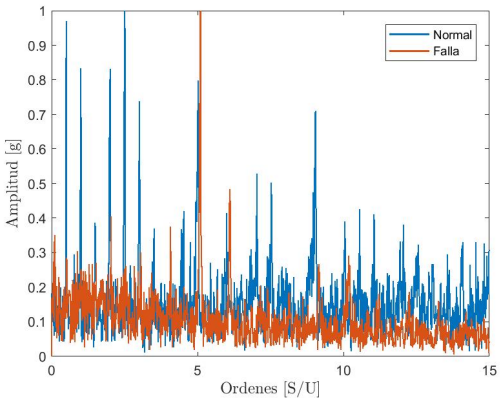

Figura J.8: Comparación SOL - Nivel 1 [ENV]

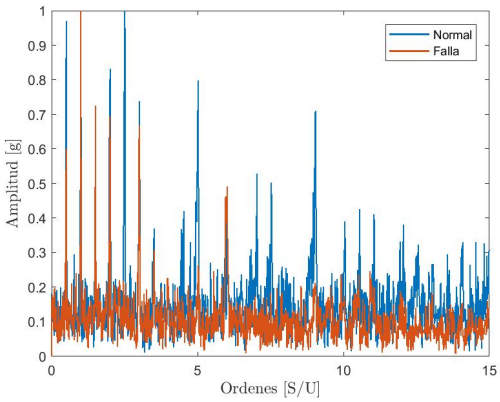

Figura J.9: Comparación SOL - Nivel 2 [ENV]

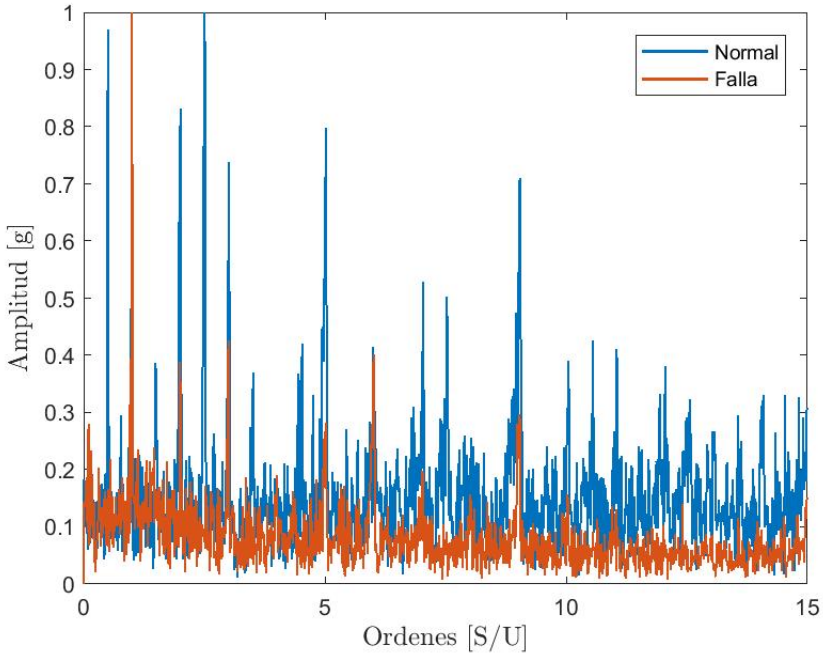

Figura J.10: Comparación SOL - Nivel 3 [ENV]

COMPARACIONES [ENV]

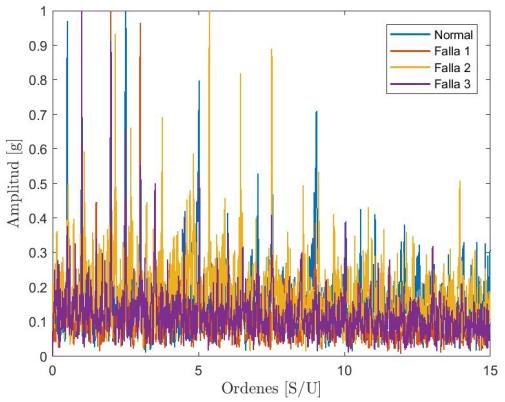

Figura J.11: Comparación PLANETA - Todos los niveles [ENV]

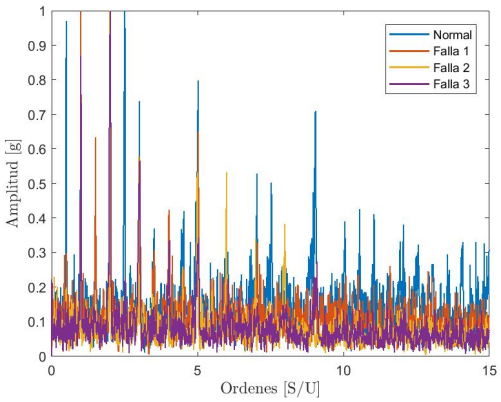

Figura J.12: Comparación SOL - Todos los niveles [ENV]

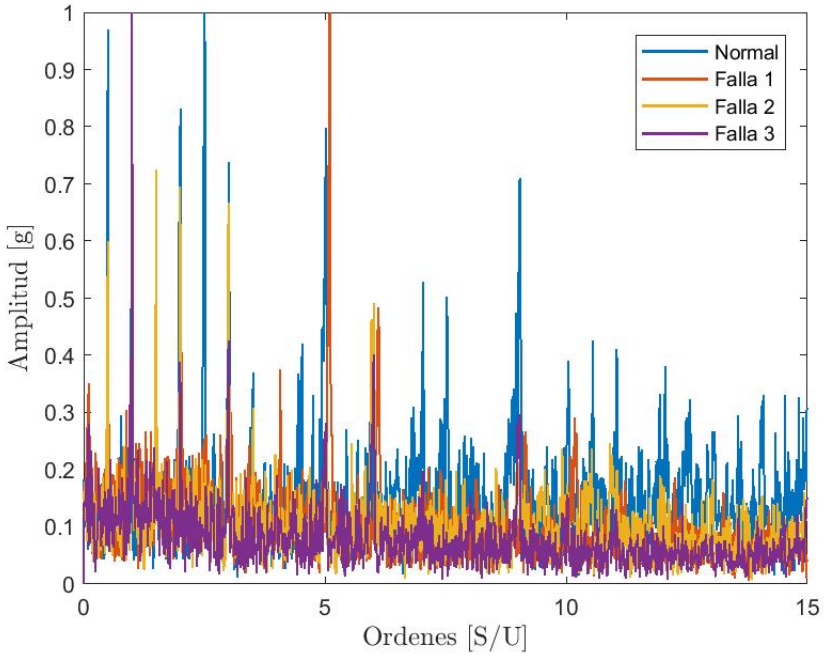

Figura J.13: Comparación SOL - Todos los niveles [ENV]

Apéndice K

PARÁMETROS SIMULACIÓN

Tabla K.1: Parámetros de la simulación para el caso sin falla

Parámetro	Símbolo	Valores probados	Valor seleccionado
Número de puntos de los datos [S/U]	L	[51200, 102400]	51200
Número de señales generadas [S/U]	N	[200, 600, 1800]	1800
Variación de la frecuencia de engrane [Hz]	f_{var_m}	[0.5, 1, 1.5]	1.5
Variación de la frecuencia de entrada [Hz]	f_{var_s}	[0.5, 1, 1.5]	1.5
Variación de la frecuencia de salida [Hz]	f_{var_c}	[0.5, 1, 1.5]	1.5
Variación de la frecuencia de falla local [Hz]	f_{var_p}	[0.2, 0.3, 0.4]	0.4
Magnitud del ruido en la señal de giro de entrada [S/U]	E_s	[0.5, 1, 2]	2
Magnitud del ruido en la señal de giro de salida [S/U]	E_c	[0.5, 1, 2]	2
Magnitud del ruido en la señal de engrane [S/U]	E_m	[0.5, 1, 2]	2
Magnitud de la falla [g]	A_p	[0.5, 1, 2]	2
Amplitud de la componente en la señal de giro de entrada [S/U]	A_s	[0.1 , 0.2 , 0.3]	0.2
Amplitud de la componente en la señal de giro de salida [S/U]	A_c	[0, 0.1, 0.2]	0
Amplitud de la componente en la señal de engrane [S/U]	A_m	[1, 2, 3]	3
Amplitud de la variación de la componente en la señal de giro de entrada [S/U]	V_s	[0.1, 0.2, 0.3]	0.3
Amplitud de la variación de la componente en la señal de giro de salida [S/U]	V_c	[0.05, 0.1, 0.3]	0.05
Amplitud de la variación de la componente en la señal de engrane [S/U]	V_m	[0.3, 0.5, 0.7]	0.5
Amplitud generalizada de la TDF [S/U]	TDF_{A_m}	[0.1, 0.15, 0.2]	0.1
Amplitud de la variación generalizada de la TDF [S/U]	TDF_{V_m}	[0.3, 0.25, 0.2]	0.1

Tabla K.2: Parámetros de la simulación para el caso falla en planetas

Parámetro	Símbolo	Valores probados	Valor seleccionado
Número de puntos de los datos [S/U]	L	[51200, 102400]	51200
Número de señales generadas [S/U]	N	[200, 600, 1800]	600
Variación de la frecuencia de engrane [Hz]	f_{var_m}	[0.5, 1, 1.5]	1.5
Variación de la frecuencia de entrada [Hz]	f_{var_s}	[0.5, 1, 1.5]	1.5
Variación de la frecuencia de salida [Hz]	f_{var_c}	[0.5, 1, 1.5]	1.5
Variación de la frecuencia de falla local [Hz]	f_{var_p}	[0.2, 0.3, 0.4]	0.4
Magnitud del ruido en la señal de giro de entrada [S/U]	E_s	[0.5, 1, 2]	2
Magnitud del ruido en la señal de giro de salida [S/U]	E_c	[0.5, 1, 2]	2
Magnitud del ruido en la señal de engrane [S/U]	E_m	[0.5, 1, 2]	2
Magnitud de la falla [g]	A_p	[0.5, 1, 2]	2
Amplitud de la componente en la señal de giro de entrada [S/U]	A_s	[0.1 , 0.2 , 0.3]	0.2
Amplitud de la componente en la señal de giro de salida [S/U]	A_c	[0, 0.1, 0.2]	0
Amplitud de la componente en la señal de engrane [S/U]	A_m	[1, 2, 3]	3
Amplitud de la variación de la componente en la señal de giro de entrada [S/U]	V_s	[0.1, 0.2, 0.3]	0.3
Amplitud de la variación de la componente en la señal de giro de salida [S/U]	V_c	[0.05, 0.1, 0.3]	0.05
Amplitud de la variación de la componente en la señal de engrane [S/U]	V_m	[0.3, 0.5, 0.7]	0.5
Amplitud generalizada de la TDF [S/U]	TDF_{A_m}	[0.6, 0.85, 1]	0.85
Amplitud de la variación generalizada de la TDF [S/U]	TDF_{V_m}	[0.4, 0.15, 0]	0.15

Tabla K.3: Parámetros de la simulación para el caso falla en soles

Parámetro	Símbolo	Valores probados	Valor seleccionado
Número de puntos de los datos [S/U]	L	[51200, 102400]	51200
Número de señales generadas [S/U]	N	[200, 600, 1800]	600
Variación de la frecuencia de engrane [Hz]	f_{var_m}	[0.5, 1, 1.5]	1.5
Variación de la frecuencia de entrada [Hz]	f_{var_s}	[0.5, 1, 1.5]	1.5
Variación de la frecuencia de salida [Hz]	f_{var_c}	[0.5, 1, 1.5]	1.5
Variación de la frecuencia de falla local [Hz]	f_{var_p}	[0.2, 0.3, 0.4]	0.6
Magnitud del ruido en la señal de giro de entrada [S/U]	E_s	[0.5, 1, 2]	2
Magnitud del ruido en la señal de giro de salida [S/U]	E_c	[0.5, 1, 2]	2
Magnitud del ruido en la señal de engrane [S/U]	E_m	[0.5, 1, 2]	2
Magnitud de la falla [g]	A_p	[0.5, 1, 2]	0.8
Amplitud de la componente en la señal de giro de entrada [S/U]	A_s	[0.1 , 0.2 , 0.3]	0.1
Amplitud de la componente en la señal de giro de salida [S/U]	A_c	[0, 0.1, 0.2]	0
Amplitud de la componente en la señal de engrane [S/U]	A_m	[1, 2, 3]	2
Amplitud de la variación de la componente en la señal de giro de entrada [S/U]	V_s	[0.1, 0.2, 0.3]	0.8
Amplitud de la variación de la componente en la señal de giro de salida [S/U]	V_c	[0.05, 0.1, 0.3]	0.05
Amplitud de la variación de la componente en la señal de engrane [S/U]	V_m	[0.3, 0.5, 0.7]	1.5
Amplitud generalizada de la TDF [S/U]	TDF_{A_m}	[0.6, 0.85, 1]	0.85
Amplitud de la variación generalizada de la TDF [S/U]	TDF_{V_m}	[0.4, 0.15, 0]	0.15

Tabla K.4: Parámetros de la simulación para el caso falla en aros

Parámetro	Símbolo	Valores probados	Valor seleccionado
Número de puntos de los datos [S/U]	L	[51200, 102400]	51200
Número de señales generadas [S/U]	N	[200, 600, 1800]	600
Variación de la frecuencia de engrane [Hz]	f_{var_m}	[0.5, 1, 1.5]	1.5
Variación de la frecuencia de entrada [Hz]	f_{var_s}	[0.5, 1, 1.5]	1.5
Variación de la frecuencia de salida [Hz]	f_{var_c}	[0.5, 1, 1.5]	1.5
Variación de la frecuencia de falla local [Hz]	f_{var_p}	[0.2, 0.3, 0.4]	0.4
Magnitud del ruido en la señal de giro de entrada [S/U]	E_s	[0.5, 1, 2]	2
Magnitud del ruido en la señal de giro de salida [S/U]	E_c	[0.5, 1, 2]	2
Magnitud del ruido en la señal de engrane [S/U]	E_m	[0.5, 1, 2]	2
Magnitud de la falla [g]	A_p	[0.5, 1, 2]	2
Amplitud de la componente en la señal de giro de entrada [S/U]	A_s	[0.1 , 0.2 , 0.3]	0.3
Amplitud de la componente en la señal de giro de salida [S/U]	A_c	[0, 0.1, 0.2]	0
Amplitud de la componente en la señal de engrane [S/U]	A_m	[1, 2, 3]	3
Amplitud de la variación de la componente en la señal de giro de entrada [S/U]	V_s	[0.1, 0.2, 0.3]	0.3
Amplitud de la variación de la componente en la señal de giro de salida [S/U]	V_c	[0.05, 0.1, 0.3]	0.05
Amplitud de la variación de la componente en la señal de engrane [S/U]	V_m	[0.3, 0.5, 0.7]	0.5
Amplitud generalizada de la TDF [S/U]	TDF_{V_m}	[0.6, 0.85, 1]	0.85
Amplitud de la variación generalizada de la TDF [S/U]	TDF_{V_m}	[0.4, 0.15, 0]	0.15

Apéndice L

CÓDIGO ANN

```
#%% FUNCION PARA GRAFICAR MATRIZ DE CONFUSION
# Esta funcion permite graficar la matriz de confusion en un formato mas adecuado para su analisis.

import numpy as np
def plot_confusion_matrix(cm,
                          target_names,
                          title='Confusion matrix',
                          cmap=None,
                          normalize=True):
    """
    given a sklearn confusion matrix (cm), make a nice plot

    Arguments
    _________
    cm:           confusion matrix from sklearn.metrics.confusion_matrix

    target_names: given classification classes such as [0, 1, 2]
                  the class names, for example: ['high', 'medium', 'low']

    title:        the text to display at the top of the matrix

    cmap:         the gradient of the values displayed from matplotlib.pyplot.cm
                  see http://matplotlib.org/examples/color/colormaps_reference.html
                  plt.get_cmap('jet') or plt.cm.Blues

    normalize:    If False, plot the raw numbers
                  If True, plot the proportions
```

```
28
29      Usage
30      _____
31      plot_confusion_matrix(cm           = cm,                   #
           confusion matrix created by
32                                                                  #
                                                                       sklearn
                                                                       .
                                                                       metrics
                                                                       .
                                                                       confusion_mat
33                            normalize    = True,                 #
                                 show proportions
34                            target_names = y_labels_vals,        #
                                 list of names of the classes
35                            title        = best_estimator_name) #
                                 title of graph
36  .
37      Citiation
38      _________
39      http://scikit-learn.org/stable/auto_examples/model_selection/
           plot_confusion_matrix.html
40
41      """
42      import matplotlib.pyplot as plt
43      import numpy as np
44      import itertools
45
46      accuracy = np.trace(cm) / float(np.sum(cm))
47      misclass = 1 - accuracy
48
49      if cmap is None:
50          cmap = plt.get_cmap('Blues')
51
52      plt.figure(figsize=(8, 6))
53      plt.imshow(cm, interpolation='nearest', cmap=cmap)
54      plt.title(title)
55      plt.colorbar()
56
57      if target_names is not None:
58          tick_marks = np.arange(len(target_names))
59          plt.xticks(tick_marks, target_names, rotation=45)
60          plt.yticks(tick_marks, target_names)
61
62      if normalize:
63          cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
```

```
64
65
66      thresh = cm.max() / 1.5 if normalize else cm.max() / 2
67      for i, j in itertools.product(range(cm.shape[0]), range(cm.
           shape[1])):
68          if normalize:
69              plt.text(j, i, "{:0.4f}".format(cm[i, j]),
70                       horizontalalignment="center",
71                       color="white" if cm[i, j] > thresh else "
                            black")
72          else:
73              plt.text(j, i, "{:,}".format(cm[i, j]),
74                       horizontalalignment="center",
75                       color="white" if cm[i, j] > thresh else "
                            black")
76
77
78      plt.tight_layout()
79      plt.ylabel('Etiqueta correcta')
80      plt.xlabel('Etiqueta predicha \naccuracy={:0.4f}; misclass
           ={:0.4f}'.format(accuracy, misclass))
81      plt.show()
82
83  #%% Extraccion de los datos
84
85  import numpy as np
86  import matplotlib.pyplot as plt
87  import pandas as pd
88
89  import keras
90  from keras.models import Sequential
91  from keras.layers import Dense
92  from keras.layers import Dropout
93  from keras.utils import np_utils
94  from keras import optimizers
95  import pandas as pd
96  import numpy as np
97  import matplotlib.pyplot as plt
98  from scipy import stats
99  import tensorflow as tf
100 import pickle
101 import seaborn as sns
102 from pylab import rcParams
103 from sklearn.model_selection import train_test_split
104 from keras.models import Model, load_model
105 from keras.layers import Input, Dense
106 from keras.callbacks import ModelCheckpoint, TensorBoard
```

```
107 from keras import regularizers
108 from sklearn import preprocessing
109 import pyuff
110 import scipy.io as sio
111 import os
112
113 # ABRO LA CARPETA EN DONDE SE ENCUENTRAN LOS ARCHIVOS
114
115 os.chdir('C:/Users/galla/Desktop/MEMORIA V.D/[SIM] [3] RESULTADOS
       /[SIM] MED TDF NORM')
116
117 # SE EXTRAEN LAS TDF DE LOS DATOS SIMULADOS 1
118
119 X_SIM_MED_NORMAL_NORM = sio.loadmat('X_SIM_MED_NORMAL_NORM.mat')
120 X_SIM_MED_NORMAL_NORM_data_x = X_SIM_MED_NORMAL_NORM["
       X_SIM_MED_NORMAL_NORM"]
121
122 X_SIM_MED_PLANETA_NORM = sio.loadmat('X_SIM_MED_PLANETA_NORM.mat'
       )
123 X_SIM_MED_PLANETA_NORM_data_x = X_SIM_MED_PLANETA_NORM["
       X_SIM_MED_PLANETA_NORM"]
124
125 # SE EXTRAEN LAS TDF DE LOS DATOS SIMULADOS 2
126
127 X_SIM_MED_ARO_NORM = sio.loadmat('X_SIM_MED_ARO_NORM.mat')
128 X_SIM_MED_ARO_NORM_data_x = X_SIM_MED_ARO_NORM["
       X_SIM_MED_ARO_NORM"]
129
130 # SE EXTRAEN LAS TDF DE LOS DATOS SIMULADOS 3
131
132 X_SIM_MED_SOL_NORM = sio.loadmat('X_SIM_MED_SOL_NORM.mat')
133 X_SIM_MED_SOL_NORM_data_x = X_SIM_MED_SOL_NORM["
       X_SIM_MED_SOL_NORM"]
134
135 #%% SE ETIQUETAN LOS DATOS
136
137 # SE LE PONEN ETIQUETAS A LOS DATOS NORMALES
138
139 X_SIM_MED_NORMAL_NORM_data_y = 0 * np.ones((len(
       X_SIM_MED_NORMAL_NORM_data_x),1))
140
141 # SE LE PONEN ETIQUETAS A LOS DATOS FALLA DE ARO
142
143 X_SIM_MED_ARO_NORM_data_y = 1 * np.ones((len(
       X_SIM_MED_ARO_NORM_data_x),1))
144
145 # SE LE PONEN ETIQUETAS A LOS DATOS  FALLA DE SOL
```

```
146
147 X_SIM_MED_SOL_NORM_data_y = 1 * np.ones((len(
      X_SIM_MED_SOL_NORM_data_x),1))
148
149 # SE LE PONEN ETIQUETAS A LOS DATOS FALLA DE PLANETA
150
151 X_SIM_MED_PLANETA_NORM_data_y = 1  * np.ones((len(
      X_SIM_MED_PLANETA_NORM_data_x),1))
152
153 # SE GRAFICA
154
155 #plt.figure()
156 #for i in range(1,10):
157 #    plt.plot(X_SIM_MED_NORMAL_NORM_data_x[i,:])
158 #    plt.xlabel('Frequency  [Hz]')
159 #    plt.ylabel('Amplitude [g]')
160 #    plt.title('COMPARACION DE CASOS')
161 #plt.show()
162
163 #plt.figure()
164 #for i in range(1,10):
165 #    plt.plot(X_SIM_MED_PLANETA_NORM_data_x[i,:])
166 #    plt.plot(X_SIM_MED_ARO_NORM_data_x[i,:])
167 #    plt.plot(X_SIM_MED_SOL_NORM_data_x[i,:])
168 #    plt.xlabel('Frequency  [Hz]')
169 #    plt.ylabel('Amplitude [g]')
170 #    plt.title('COMPARACION DE CASOS')
171 #plt.show()
172
173 X_N = X_SIM_MED_NORMAL_NORM_data_x
174 X_A = X_SIM_MED_ARO_NORM_data_x
175 X_S = X_SIM_MED_SOL_NORM_data_x
176 X_P = X_SIM_MED_PLANETA_NORM_data_x
177
178 Y_N = X_SIM_MED_NORMAL_NORM_data_y
179 Y_A = X_SIM_MED_ARO_NORM_data_y
180 Y_S = X_SIM_MED_SOL_NORM_data_y
181 Y_P = X_SIM_MED_PLANETA_NORM_data_y
182
183 X_T = np.concatenate((X_N,X_A,X_S,X_P))
184 Y_T = np.concatenate((Y_N,Y_A,Y_S,Y_P))
185
186
187 #%% ETIQUETAS CATEGORICAS
188
189 from sklearn.model_selection import train_test_split
```

```
190 x_train, x_test, y_train, y_test = train_test_split(X_T, Y_T,
       test_size = 0.2, random_state = 0)
191
192 #%% PARTE 2: ARQUITECTIRA DE LA ANN
193
194 # SE IMPORTAN LOS PAQUETES DE KERAS QUE SON IMPORTANTES
195
196 from keras.models import Sequential
197 from keras.layers import Dense
198
199 # SE INICIALIZA LA RED NEURONAL
200
201 S = 7000
202
203 model = Sequential()
204
205 model.add(Dense(units = S, activation = 'relu', input_dim = S))
206
207 S1 = 4000
208 S2 = 2000
209 S3 = 500
210 S4 = 200
211 S5 = 50
212
213 # SE AGREGA LA PRIMERA CAPA A LA RED NEURONAL
214
215 model.add(Dense(units = S1, activation = 'relu'))
216 model.add(Dropout(rate=0.2))
217 # SE AGREGA LA SEGUNDA CAPA A LA RED NEURONAL
218
219 model.add(Dense(units = S2, activation = 'relu'))
220 model.add(Dropout(rate=0.2))
221
222 # SE AGREGA LA TERCERA CAPA A LA RED NEURONAL
223
224 model.add(Dense(units = S3, activation = 'relu'))
225 model.add(Dropout(rate=0.2))
226 # SE AGREGA LA CUARTA CAPA A LA RED NEURONAL
227
228 model.add(Dense(units = S4, activation = 'relu'))
229 model.add(Dropout(rate=0.2))
230 #model.add(Dropout(rate=0.5))
231
232 model.add(Dense(units = S5, activation = 'relu'))
233 model.add(Dropout(rate=0.2))
234 #model.add(Dropout(rate=0.5))
235
```

```
236 # SE AGREGA LA CAPA FINAL A LA RED
237
238 model.add(Dense(units = 1, activation = 'sigmoid'))
239
240 # SE COMPILA LA ANN
241
242 adam = keras.optimizers.Adam(lr=1e-5, beta_1=0.9, beta_2=0.999,
       epsilon=None, decay=0.0, amsgrad=False)
243
244 model.compile(optimizer = adam , loss = 'binary_crossentropy',
       metrics = ['accuracy'])
245
246 # SE HACE EL FIT CON EL TRAININ SET
247
248 model_history = model.fit(x_train, y_train, batch_size = 400 ,
       epochs = 80, validation_split = 0,validation_data=(x_test,
       y_test))
249
250 # SE CLASIFICA Y DIAGNOSTICA PARA VER LA EFECTIVIDAD DE LA RED
251
252 y_pred = model.predict_classes(x_test)
253
254 test_loss,test_accuracy = model.evaluate(x_test, y_test,
       batch_size = 400)
255
256 #%% SE VISUALIZA EL RENDIMIENTO DE LA RED
257
258 # CLassification/Diagnostics
259
260 # To visualize the performance of the model for a classification/
       diagnostics task, ...
261 # ... a confusion matrix C_ij is generated, where i corresponds
       to the real labels and j the predicted labels.
262
263 from sklearn.metrics import confusion_matrix
264 conf_matrix = confusion_matrix(y_test, y_pred)
265 #print('Confusion Matrix:\n',conf_matrix)
266 plot_confusion_matrix(cm           = conf_matrix,
267                       normalize    = False,
268                       target_names = ['Equipo sano', 'Equipo con
                          fallas'],
269                       title        = "Confusion Matrix [N de
                          casos]")
270
271 # ABRO LA CARPETA EN DONDE SE ENCUENTRAN LOS ARCHIVOS
272
```

```
273 os.chdir('C:/Users/galla/Desktop/MEMORIA V.D/[SIM] [4] MACHINE
       LEARNING/[SIM] FIGURAS ANN')
274
275 plt.savefig('CONFUSION_MATRIx.png')
276
277 #%% SE GRAFICA LA FUNCION DE PERDIDA
278
279 plt.figure()
280 plt.plot(model_history.history['loss'])
281 plt.plot(model_history.history['val_loss'])
282 plt.title('ANN model loss')
283 plt.ylabel('Loss')
284 plt.xlabel('Epoch')
285 plt.legend(['train', 'validation'], loc='upper right')
286 plt.show()
287
288 plt.savefig('LOSS.png')
289
290 #%% SE GUARDAN LOS DATOS EN EL MODELO
291
292 # ABRO LA CARPETA EN DONDE SE ENCUENTRAN LOS ARCHIVOS
293
294 os.chdir('C:/Users/galla/Desktop/MEMORIA V.D/[EXP] [4] MACHINE
       LEARNING/[EXP] ANN')
295
296 model.save('BINARIO.h5')
```