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

ANÁLISIS DE DISPONIBILIDAD DE EQUIPO DENSE PLASMA FOCUS MEDIANTE REDES NEURONALES

MEMORIA PARA OPTAR AL TITULO DE INGENIERO CIVIL MECÁNICO

DANIEL ESTEBAN ZANELLI SANHUEZA

PROFESOR GUÍA:
ENRIQUE LÓPEZ DROGUETT

MIEMBROS DE LA COMISIÓN:
LEOPOLDO SOTO NORAMBUENA
SERGIO COURTIN VEGA

Memoria realizada en conjunto con el Departamento de Plasmas TermoNucleares (DPTN)
de la Comisión Chilena de Energía Nuclear (CCHEN)

SANTIAGO DE CHILE
2017

RESUMEN DE LA MEMORIA PARA OPTAR AL
TÍTULO DE: Ingeniero Civil Mecánico
POR: Daniel Esteban Zanelli Sanhueza
FECHA: 22/10/2017
PROFESOR GUÍA: Enrique López Droguett

ANÁLISIS DE DISPONIBILIDAD DE EQUIPO DENSE PLASMA FOCUS MEDIANTE REDES NEURONALES

La búsqueda por una energía renovable prácticamente ilimitada ha sido uno de los focos principales de la motivación en la investigación de energía nuclear por fusión atómica. Sin embargo, uno de los principales problemas con la energía por fusión es ser capaz de mantenerla estable con una energía suficiente para conservar la reacción en cadena. Esto se pensaba que solo se podría lograr energizando un gas a tal nivel que se convierte en plasma y lograr que se mantenga estable. Sin embargo, hay otras aproximaciones que dan por sentado que el plasma colapsa y se intenta explotar esta fragilidad. El Dense Plasma Focus permite generar un plasma ascendente que dura centenas de nanosegundos antes de colapsar. Es de sumo interés para estudiar el comportamiento de este plasma en los equipos PF en diversos aspectos.

Debido a esto es imprescindible el estudio del comportamiento del plasma, por lo que se deben realizar experimentos para determinar las restricciones, condiciones necesarias y emisiones de energía que son propias del cuarto estado de la materia. Dentro de la CCHEN existe un departamento dedicado al estudio de las propiedades físicas del plasma, en función de los parámetros de los equipos usados para generarlas. Además, es de interés comprender las aplicaciones que puede tener el plasma en la ciencia de materiales, biología y eliminación de deshechos, entre otros. Sin embargo fallas en estos equipos son comunes e impiden que se puedan realizar los experimentos en el momento esperado, causando problemas posteriores en varios sentidos.

El análisis realizado por el memorista comprende un estudio integral del funcionamiento del sistema de los equipos de generación de plasma del tipo Dense Plasma Focus. Para lograr esto, en este análisis se determinan elementos críticos, se realizan diagramas de sistema y planos de equipo, además se obtienen parámetros que permiten evidenciar un avance en el deterioro del sistema, teniendo en cuenta las distribuciones de probabilidades correspondientes. Finalmente, se plantean propuestas de rediseño del sistema, con el objetivo de mejorar la confiabilidad y disponibilidad de este. Junto con esto, se proponen protocolos de mantenimiento y recomendaciones sobre monitoreo de variables clave en la detección de fallas del sistema.

De la presente memoria se concluye que manteniendo un monitoreo del crecimiento porcentual de Kurtosis, Shape Indicator, Clearance Indicator, Crest Indicator y Amortiguamiento de las señales características del sistema se puede estimar la cantidad de disparos desde el inicio de un experimento. Por otro lado, se concluye que distinguir los datos que presenten instancia de Dip permite establecer la probabilidad de ocurrencia como una distribución de Weibull. Distinguir datos que posean Dip de los que no puede ser realizado mediante un pre-tratamiento de las señales y posterior entrenamiento de una red neuronal de clasificación. El clasificador permite identificar con una asertividad del $93{,}25\,\%$ para datos del mismo tipo y $83{,}3\,\%$ para datos diferentes.

Tabla de Contenido

1. Introducción **1**
- 1.1. Objetivos . . . 1
- 1.2. Alcances . . . 1

2. Metodología **2**
- 2.1. Recopilaciones de Antecedentes . . . 2
- 2.2. Generación de Diagramas de Sistema . . . 2
- 2.3. Análisis FMEA/FMECA . . . 2
- 2.4. Análisis de Modos de Falla Críticos . . . 2
- 2.5. Experimentación Preliminar . . . 2
- 2.6. Análisis Preliminar . . . 2
- 2.7. Análisis de Mantenimiento Predictivo . . . 3
- 2.8. Experimentación Final . . . 3
- 2.9. Análisis Final y Gestión de Activos Físicos . . . 3

3. Antecedentes **4**
- 3.1. Dense Plasma Focus . . . 4
- 3.2. Comisión Chilena de Energía Nuclear (CCHEN) . . . 5
- 3.3. Funcionamiento de Sistemas PF . . . 5
- 3.4. FMEA / FMECA . . . 6
- 3.5. Distribuciones de Tiempos de Falla . . . 7

4. Funcionamiento PF-2J **10**
- 4.1. Antecedentes Específicos . . . 10
- 4.2. PF-2J . . . 10
- 4.3. Generación de Plasma . . . 14

5. Experimentación Preliminar **17**
- 5.1. Introducción . . . 17
- 5.2. Montaje Experimental . . . 17
- 5.3. Resultados Preliminares . . . 19
  - 5.3.1. Análisis FMECA . . . 19
  - 5.3.2. Resultados Experimentales . . . 22
- 5.4. Análisis de Resultados Preliminares . . . 29

6. Análisis de Correlación de Datos **32**
- 6.1. Estimación de N por Combinación Lineal . . . 32
- 6.2. Estimación de N Mediante Redes Neuronales . . . 35

7. Análisis de Disponibilidad y Predicción de Dip **38**
- 7.1. Fenómeno de Dip . . . 38
- 7.2. Disponibilidad de PF . . . 39
- 7.3. Pre-tratamiento de Señales . . . 40
- 7.4. Clasificación Usando Neural Networks . . . 44

8. Experimentación Final **47**
8.1. Montaje . . . 47
8.2. Resultados Finales . . . 48
8.2.1. Identificación de Dip Mediante NN . . . 48
8.2.2. Pronósticos de Disponibilidad . . . 49

9. Análisis Final **55**
9.1. Análisis de Resultados . . . 55
9.2. Propuestas de Mantenimiento . . . 55
9.3. Discusión . . . 57

10. Conclusiones **59**

11. Bibliografía **60**

12. Anexos **62**
A. Plano Conjunto de Equipo PF-2J . . . 62
B. Plano de Aislante de PF-2J . . . 63
C. Código Python para Ventana de Selección de Datos con 'Dip' . . . 64
D. Código Python para Feature Extraction de Perfiles Característicos . . . 72
E. Código Python para Limpieza de Datos Repetidos . . . 79
F. Código Python para Lectrura y Pre-tratamiento de Datos . . . 82
G. Código Python para Entrenamiento Red Neuronal . . . 88
H. Código Python para Comparar Predicciones con Datos Pre-clasificados . . . 92
I. Código Python para Pronosticos de Disponibilidad y Predicciónes de Falla . . . 95

Índice de Figuras

3.1. Diagrama Equipo PF[1] . . . 4
3.2. Generación de Pinch en PF[2] . . . 5
3.3. Sistema Dense Plasma Focus[3] . . . 6
3.4. Distribución de Probabilidad de Fallas[4] . . . 7
3.5. Tasa de Falla[5] . . . 8
4.1. Dense Plasma Focus 2J[6] . . . 10
4.2. Dibujo Isométrico PF-2J[7] . . . 11
4.3. Dibujo de Conjunto PF-2J[7] . . . 11
4.4. Dibujo en Corte de PF-2J[7] . . . 12
4.5. Detalle de Interior Cámara de Descarga[7] . . . 13
4.6. Fenómeno de z-pinch en PF[9] . . . 14
4.7. Diagrama de Bloques de PF 2J[10] . . . 15
5.1. Diagrama del Montaje Experimental[11] . . . 17
5.2. Montaje Experimental[12] . . . 18
5.3. Computador y Osciloscopio Contenidos en Jaula de Faraday[12] . . . 18
5.4. Matriz de Criticidad para PF-2J[18] . . . 21
5.5. Señales Características: Derivada de corriente (superior) y Voltaje (inferior), en función del tiempo[20] . . . 22

5.6. Señales Características con 'Dip': Derivada de corriente (superior) y Voltaje (inferior), en función del tiempo[21] . . . 23
5.7. GUI Desarrollada para Identificar 'Dip'[22] . . . 24
5.8. Distribución Estadística del Dip (cada barra corresponde a 200 disparos)[23] . . . 24
5.9. Peak Picking [24] . . . 26
5.10. Kurtosis y Crest Indictor en función de numero de disparos[25] . . . 27
5.11. Shape Indicator y Clearance Indicator en función de numero de disparos[25] . . . 27
5.12. Máximos y Mínimos globales en función de numero de disparos[25] . . . 28
5.13. Amortiguamiento y Entropía de Shannon en función de numero de disparos[25] . . . 28
6.1. Estimaciones de N Mediante Combinación Lineal de Parámetros[28] . . . 34
6.2. Diagrama de Red Neuronal Implementada[29] . . . 35
6.3. Sigmoide[30] . . . 36
6.4. Ejemplo de Backpropagation[31] . . . 37
6.5. Estimaciones de N Mediante Red Neuronal[32] . . . 37
7.1. Instancias de Dip Acopladas [33] . . . 39
7.2. Instancia de Dip Leve [33] . . . 39
7.3. Multiplicación de Densidades de Probabilidad Normales[34] . . . 41
7.4. Señales Caracterisiticas Pre-tratadas sin Insancia de Dip[35] . . . 42
7.5. Señales Caracterisiticas Pre-tratadas con Instancia de Dip[35] . . . 42
7.6. Resta de Señales con Curva $S(t)$ [36] . . . 43
7.7. Resta de Señales con Curva Filtrada [37] . . . 44
7.8. Red Neuronal para Clasificación de Dip[38] . . . 45
7.9. Comparación de Identificación de Dip en Datos Finales[39] . . . 46
8.1. Montaje Experimental Final [40] . . . 47
8.2. Error de Última capa de NN [41] . . . 48
8.3. Comparación de Identificación de Dip en Datos Finales [42] . . . 49
8.4. Pronosticos de Disponibilidad de Datos Preliminares (1)[43] . . . 50
8.5. Pronosticos de Disponibilidad de Datos Preliminares (2)[43] . . . 51
8.6. Pronosticos de Disponibilidad de Datos Finales (1)[44] . . . 52
8.7. Pronosticos de Disponibilidad de Datos Finales (2)[44] . . . 53
8.8. Predicciones de Momento de Falla de Datos Preliminares [45] . . . 54
8.9. Predicciones de Momento de Falla de Datos Finales [46] . . . 54
9.1. Sistema de Mantenimiento Predictivo Asistido [47] . . . 56
9.2. Sistema de Mantenimiento Predictivo Automático [48] . . . 57

Índice de Tablas

4.1. Detalle de Elementos PF-2J[8] . . . 12
5.1. Tabla de Valores de Severidad de Falla[13] . . . 19
5.2. Tabla de Valores de Probabilidad de Falla[14] . . . 19
5.3. Tabla de Valores de Detectabilidad de Falla[15] . . . 20
5.4. Tabla de Análisis FMECA PF-2J[16] . . . 20
5.5. Tabla 2 de Análisis FMECA PF-2J[17] . . . 21
5.6. Elementos Críticos PF-2J[19] . . . 22
5.7. Variación de Parámetros Durante la Vida del Equipo[26] . . . 30
5.8. Cambio Porcentual de Parámetros[27] . . . 31

1. Introducción

1.1. Objetivos

Objetivo General

El objetivo general de esta memoria es realizar un análisis exhaustivo sobre la disponibilidad de los equipos Dense Plasma Focus (PF) del Departamento de Plasmas Termonucleares de la CCHEN.

Objetivos Específicos

Los objetivos específicos de esta memoria son:

- Identificar modos de falla, criticidad de los elementos e implicancias en el sistema
- Realizar una recopilación de datos de fallas de los equipos, haciendo experimentos de fallas en caso de ser necesario
- Encontrar la disponibilidad del equipo
- Concluir con un análisis de resultados, propuestas de rediseño del sistema, sugerir especificaciones de elementos y/o sugerir estrategias de mantenimiento preventivo

1.2. Alcances

El alcance de este estudio es:

- Se enfocará el estudio del equipo PF-2J presente en el DPTN de la CCHEN
- El análisis está limitado a las condiciones administrativas y económicas del reactor, experimentación de fallas puede ser limitada
- La conclusión se focalizará en plantear propuestas de mantenimiento predictivo y herramientas para lograrlo.

2. Metodología

La metodología de trabajo a realizar durante el desarrollo de este trabajo de título es la siguiente:

2.1. Recopilaciones de Antecedentes

En esta primera parte se hace una recopilación de antecedentes referentes no solo a los aspectos técnicos del sistema, sino que también de métodos de determinación de las variables claves y de análisis similares.

2.2. Generación de Diagramas de Sistema

En esta etapa se considera un análisis exhaustivo del sistema, en el que se determinan los elementos, sus relaciones y el impacto que tienen en el funcionamiento del sistema. Consiste en un análisis preliminar que ayuda a enfocar el análisis posterior.

2.3. Análisis FMEA/FMECA

Este análisis se concentra en una comparación sinérgica de las causas de las fallas con las consecuencias que pueden generar. Además se estima una probabilidad de ocurrencia de cada uno de los métodos, culminando en una comparación de criticidad de elementos.

2.4. Análisis de Modos de Falla Críticos

Aquí se utilizan los análisis anteriores como supuestos necesarios para la determinación de un modelo de falla. Se analizan los elementos críticos del sistema y se obtienen correlaciones que permitan modelar el proceso de falla efectivo.

2.5. Experimentación Preliminar

Una parte fundamental de la memoria es el estudio de las señales características del equipo para correlacionar el desgaste del equipo en función del número de ciclos. Es por esto que se requiere una experimentación para comprender la naturaleza de estas señales para identificar parámetros clave que varían en función de la vida del producto. Aquí se realiza una recopilación de datos a lo largo de la vida útil del equipo PF.

2.6. Análisis Preliminar

Teniendo en cuenta la recopilación de datos preliminares se realiza un análisis sobre los parámetros clave del equipo, identificándolos y culminando con un diagnóstico del sistema.

2.7. Análisis de Mantenimiento Predictivo

Se analizan los patrones en los datos que permitan establecer una correlación con la vida útil del equipo. Se exploran diversos métodos, incluyendo Machine Learning.

2.8. Experimentación Final

En esta sección se considera un nuevo setup experimental que toma en cuenta las sensibilidades externas y de condiciones de laboratorio. En esta jornada experimental se busca demonstrar la validez de los modelos generados.

2.9. Análisis Final y Gestión de Activos Físicos

Culminado el estudio experimental final se realizan las sugerencias para determinar la disponibilidad del equipo, junto con una solución sistemática e integral de mantenimiento. Junto con esto se proponen modelos para comprender la naturaleza del equipo, su optimización y siguientes estudios a realizar.

3. Antecedentes

3.1. Dense Plasma Focus

Para el trabajo de tesis el estudiante se enfoca en específico un tipo de equipo para generación de plasma, conocido por la literatura como 'Dense Plasma Focus '(PF). El PF es una máquina que produce un plasma de corta vida, alto en densidad y temperatura a tal nivel de ser capaz de generar fusión nuclear, emisión de rayos X y neutrones.

Esto se lleva a cabo mediante aceleración electromagnética, generando una gran diferencia de potencial ente el ánodo central y los cátodos. Un diagrama de este equipo se puede apreciar en la siguiente figura:

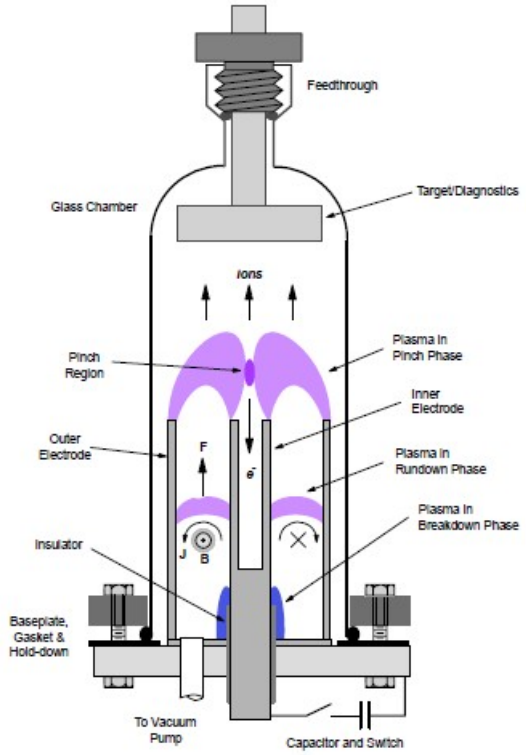


Figura 3.1: Diagrama Equipo PF[1]

Esta diferencia de potencial de gran escala genera un 'pinch', que corresponde a la compresión de un filamento conductor mediante fuerzas magnéticas y puede presentarse de varias formas (sheet pinch, theta pinch, Z pinch, screw pinch, entre otros). En el caso del equipo PF se produce un Z-pinch, creando una corriente a lo largo de las paredes del cilindro central (ánodo) de forma axial y un campo magnético de forma azimutal.

Al utilizar este equipo se requiere cargar un banco de capacitadores eléctricos, realizando una súbita descarga sobre el ánodo, lo que genera una descomposición del gas presente en la cámara y una corriente eléctrica escalando rápidamente a lo largo de las paredes de los electrodos, como se puede ver en las etapas A y B de la figura 3.2. Llegando a la etapa C el movimiento axial finaliza tras haberse desplazado a varias veces la

[1] `http://www.belljar.net/mini_f.htm`

velocidad del sonido en el gas presente en la cámara.

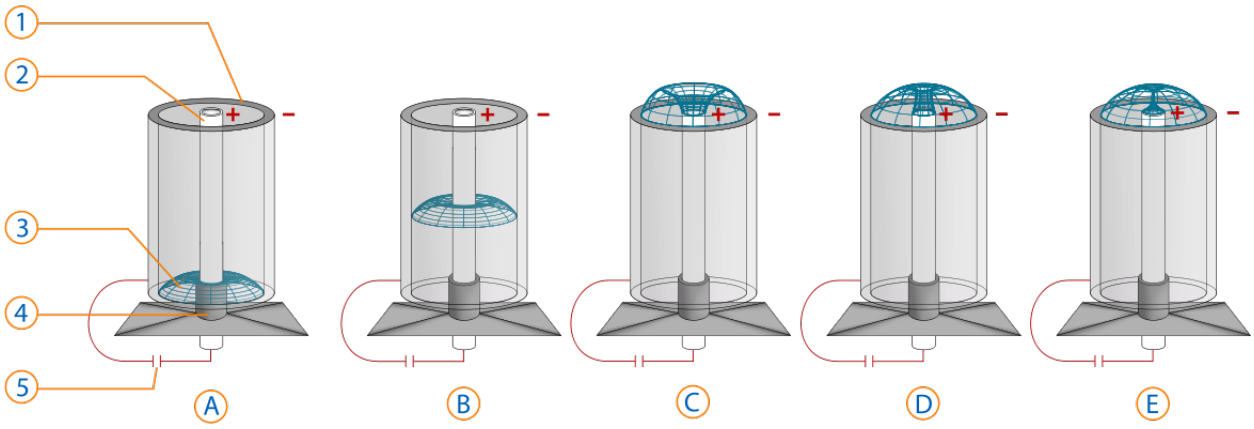


Figura 3.2: Generación de Pinch en PF[2]

La columna de plasma genera un pinch rápidamente, genera inestabilidades y colapsa, representado en las etapas C, D y E. En estas etapas se genera radiación electromagnética en escalas de tiempo de nanosegundos para PFs de baja energía (magnitud de [kJ]) y en escalas de tiempo de microsegundos para PFs de alta energía (magnitud de [MJ]).

3.2. Comisión Chilena de Energía Nuclear (CCHEN)

Cabe recordar que el análisis a realizar por el estudiante será llevado a cabo en el Departamento de Plasma Termonuclear (DPTN) de la CCHEN, dentro del cual es de principal importancia realizar mediciones científicas de variadas naturalezas, como efectos de radiación de plasma sobre materiales diversos y tejidos vivos, determinación del espectro fotónico de la radiación emitida, flujo de neutrones y descomposición de gases, entre otros.

Sin embargo, uno de los principales objetivos dentro de los proyectos planteados por el equipo de la CCHEN es la generación de una relación de escala entre equipos PF utilizando principios de similitud [11]. Todo esto con la misión de encontrar una relación de similitud para las diferentes magnitudes de energía de los equipos, permitiendo un análisis certero de reacciones a gran escala sin tener las restricciones económicas, energéticas, de seguridad y de horas-hombre asociadas a la operación de equipos de esta naturaleza. Por esta misma razón el DPTN cuenta con varios equipos de generación de plasma con un diseño de PF, desde magnitudes menores a 1 joule a cientos de kilojoules.

3.3. Funcionamiento de Sistemas PF

El funcionamiento de la unidad de PF no sería posible sin una serie de elementos aledaños que permitan tanto generar la corriente, como extraer, insertar y mantener gases deseados dentro de la cámara, junto con

[2] `https://lasttechage.wordpress.com/2015/01/02/fusion-paths-not-taken-3/`

proporcionar una estructura adecuada para montar los elementos y poder acceder la zona de generación de plasma para realizar mediciones.

Para poder generar la cantidad de diferencia de potencial requerida para generar el pinch, es necesario tener a disposición una gran cantidad de condensadores de alta capacitancia, junto con una unidad de Spark Gap (SG) que permita la descarga repentina al sobrepasar cierto umbral predefinido y, evidentemente, se requieren conectores eléctricos que permitan el cierre del circuito.

Junto con esto se necesita el acople de una bomba de vacío y fittings correspondientes para poder conectar el sistema a tanques que provean gases determinados para poder generar distintos tipos de plasmas, además de una fuente de poder, visores para la cámara de plasma y otros elementos opcionales que dependerán de la naturaleza del experimento. Para ilustrar de mejor manera, un sistema simple se aprecia en la figura 3.3.

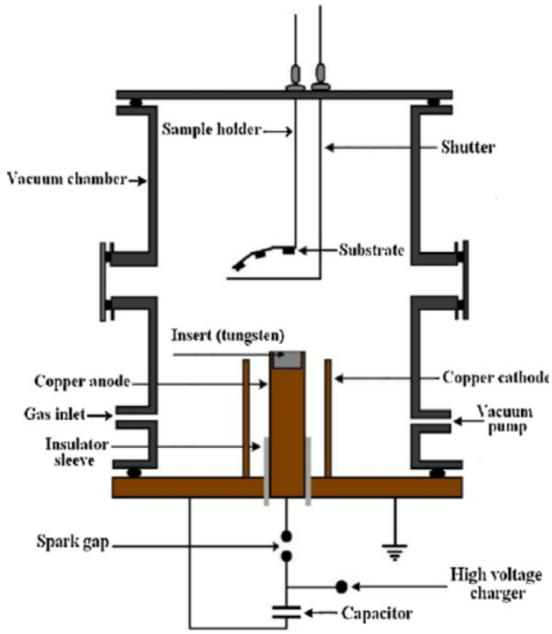


Figura 3.3: Sistema Dense Plasma Focus[3]

3.4. FMEA / FMECA

Failure Mode and Effects Analysis (FMEA) es un método para la identificación de fallas en un sistema cualquiera. Consiste en generar indicadores sobre el nivel de riesgo que presenta un sistema ante sus distintos métodos de falla, considerando tres aspectos:

- Severidad de la Consecuencia
- Probabilidad de Ocurrencia

[3] http://cpb.iphy.ac.cn/article/2015/cpb_24_6_065204.html

- Detectabilidad de la Causa

De acuerdo a esto se pueden identificar los elementos que ponderen más intensamente en la disponibilidad efectiva del sistema.

Este método permite un análisis cualitativo del sistema, siendo de alta utilidad en el momento de diseñar u optimizar un sistema para una mayor vida útil. Sin embargo, este sistema puede inducir a aproximaciones grosas sobre las probabilidades de ocurrencia.

En contraste, el método Failure Mode, Effects and Criticality Analysis (FMECA) considera todo lo anterior pero enfocándose en cómo se comporta la probabilidad de falla a partir de bases de datos y como esta se compara con la severidad de sus consecuencias, permitiendo una estimación de la vida útil del dispositivo en cuestión.

3.5. Distribuciones de Tiempos de Falla

De lo descrito anteriormente en los antecedentes generales se puede notar que la confiabilidad de un elemento o sistema se determina a partir de la distribución probabilística que siguen las fallas, como se ilustra en la figura 3.4 y ecuación 3.1:

$$f(t) = \frac{dF(t)}{dt} = -\frac{dR(t)}{dt} \tag{3.1}$$

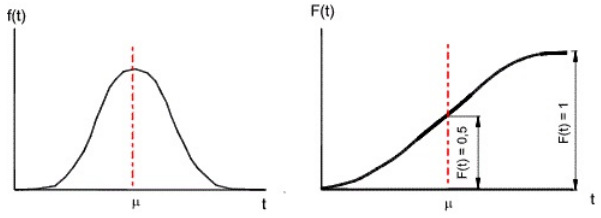


Figura 3.4: Distribución de Probabilidad de Fallas[4]

Ahora, la forma que toma esta distribución varía según el modelo, elemento, sistema y supuestos usados. En la literatura se destacan las distribuciones weibull, gamma, exponencial, binomial, normal, lognormal, chi-cuadrado, entre otras. Factor común de varias distribuciones es la tasa de falla o hazard rate, que determina la intensidad del crecimiento de la probabilidad de falla a lo largo del tiempo y se define como:

$$h(t) = \frac{f(t)}{R(t)} \tag{3.2}$$

Esta variable determina el proceso que está viviendo el dispositivo de acuerdo a si es decreciente, constante o creciente a lo largo del tiempo, como se muestra en la figura 3.5:

[4] http://www.roymech.co.uk/Useful_Tables/ARM/Failure_Distributions.html

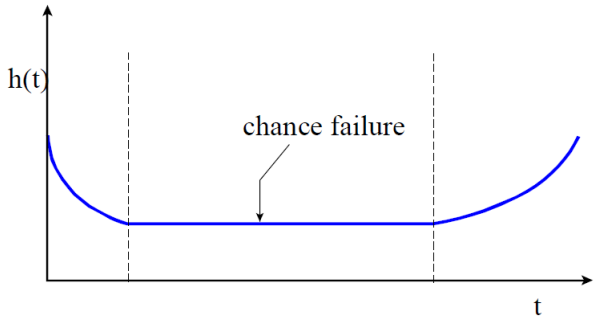


Figura 3.5: Tasa de Falla[5]

Resulta crucial este elemento para poder determinar la confiabilidad de un sistema, que puede hacerse mediante varias perspectivas, una de ellas consiste en una aproximación por ley de potencia:

$$h(t) = \lambda(t) = \beta \cdot \theta \cdot t^{\beta-1} \tag{3.3}$$

Donde:

- Si $\beta < 1$, el sistema está mejorando
- Si $\beta > 1$, el sistema se está deteriorando
- Si $\beta = 1$, corresponde a un Proceso Homogéneo de Poisson (HPP)

La obtención de estos parámetros puede realizarse a partir de bases de datos que incluyan bitácoras de fallas, explicitando la cantidad de fallas y los tiempos (o ciclos) en los que se presentaron las fallas.

Se pueden obtener estimaciones mediante las siguientes ecuaciones para datos de Tipo I, es decir, con un tiempo de experimento constante y contabilización de fallas:

$$\beta = \frac{n}{n\ ln(T) - \sum_{i=1}^{n} ln(t_i)} \tag{3.4}$$

$$\theta = \frac{n}{T^{\beta}} \tag{3.5}$$

Y, análogamente para datos de Tipo II (número de fallas de elementos constante, en un período de tiempo variable):

$$\beta = \frac{n}{(n-1)ln(T) - \sum_{i=1}^{n-1} ln(t_i)} \tag{3.6}$$

$$\theta = \frac{n}{T^{\beta}} \tag{3.7}$$

[5] http://www.weibull.com/hotwire/issue21/hottopics21.htm

Siendo T el tiempo total del experimento y t_i los tiempos en los que se presentaron las fallas, particularmente para datos Tipo I el tiempo T es fijado de forma arbitraria y en datos Tipo II $T = t_n$, siendo n fijado arbitrariamente.

En el Departamento de Plasmas Termonucleares (DPT), es común que los equipos de generación de plasma permanezcan indispuestos debido a fallas sorpresivas, falta de coordinación, reconocimiento de métodos de fallas bajo, escaso inventario de piezas de repuestos, monitoreo de equipos escaso o nulo, entre otros. Por esto es de vital importancia maximizar la disponibilidad de los equipos, es decir maximizar el porcentaje de tiempo que cierto equipo está disponible para utilizar. Considerando un sistema reparable, la disponibilidad de define como:

$$A = \frac{MTBF}{MTBF + MTTR} \tag{3.8}$$

Donde:

- A es la disponibilidad de un equipo
- MTBF es el tiempo medio entre fallas
- MTTR es el tiempo medio para la reparación

El tiempo medio para la reparación es inexacto, ya que depende de factores humanos y de gestión. Sin embargo, también depende fuertemente del tipo de elemento que falle y de las consecuencias que provocaron esta falla.

Ahora, el tiempo medio entre fallas es determinado por la confiabilidad que presentan los elementos que lo integran y como estos se conectan entre sí. La confiabilidad de un elemento se define como:

$$R(t) = Pr\{T > t\} \tag{3.9}$$

Donde:

- $R(t)$ es las confiabilidad de un elemento dado, en un tiempo t
- T es una variable aleatoria que representa el tiempo en el que este elemento falla

Resulta crucial entender el comportamiento de cada elemento del sistema, encontrando sus modos de falla y determinando entonces cual es la naturaleza de la distribución de la variable aleatoria T que determina cuando el sistema quedará indispuesto.

Además es de vital importancia entender los tipos de fallas que se presentan, analizando las consecuencias de estas y por qué se producen.

4. Funcionamiento PF-2J

4.1. Antecedentes Específicos

Para poder realizar un análisis a esta serie de equipos PF basta con analizar uno de estos, ya que el funcionamiento es el mismo. Entre un modelo y otro sólo cambia la escala y pequeñas modificaciones, como por ejemplo sensores adicionales para tomar datos específicos al experimento en cuestión. Es por este motivo que la memoria se enfocará en el estudio del equipo más pequeño, el $PF-2J$.

4.2. PF-2J

El equipo Dense Plasma Focus 2J corresponde a la versión más pequeña de la serie, con una capacidad energética de apenas $[2J]$. Una imagen de este equipo se puede apreciar en la figura 4.1.Cabe notar que las imágenes y dibujos mecánicos presentados a continuación fueron generados en el trabajo de memoria y no están basadas en referencias externas.

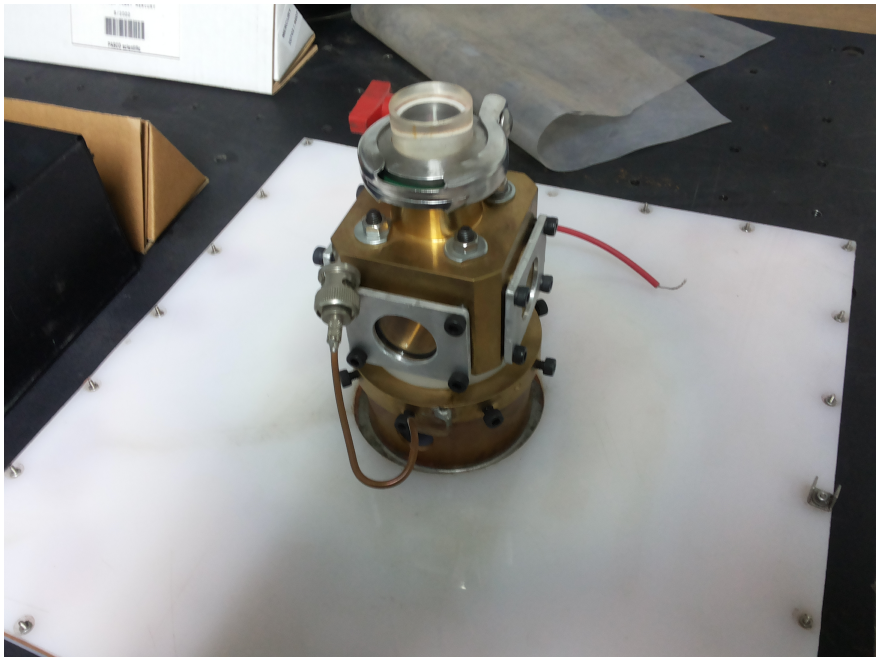


Figura 4.1: Dense Plasma Focus 2J[6]

El funcionamiento del equipo es mayoritariamente eléctrico, sin embargo todas las conexiones conforman un sistema que puede modelarse como un diagrama de bloques entre elementos.

Se ha realizado una digitalización del sistema físico a software CAD. De esta manera las piezas pueden ser usadas para futura referencia.

En la figura 4.2 se muestra el PF-2J en su vista isométrica, seguido por la figura 4.3 que muestra las dimensiones principales del equipo y la sección de corte de la figura siguiente, donde se muestra toda la configuración interior del generador.

[6]Imagen tomada en las dependencias del DPTN en el transcurso del trabajo de título

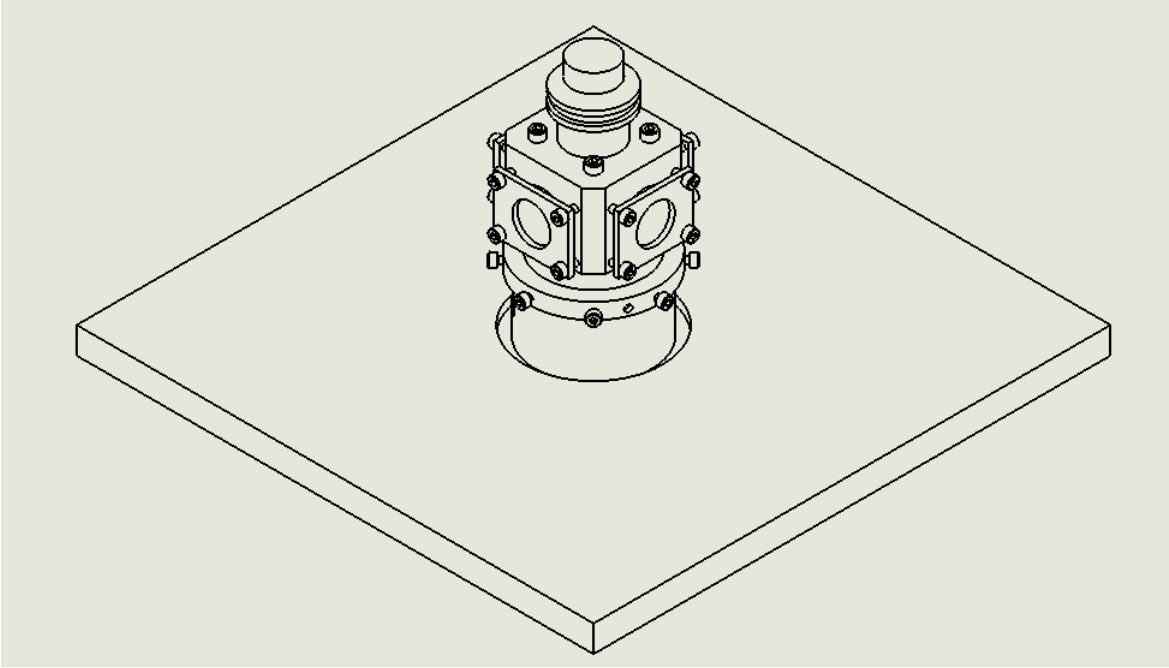


Figura 4.2: Dibujo Isométrico PF-2J[7]

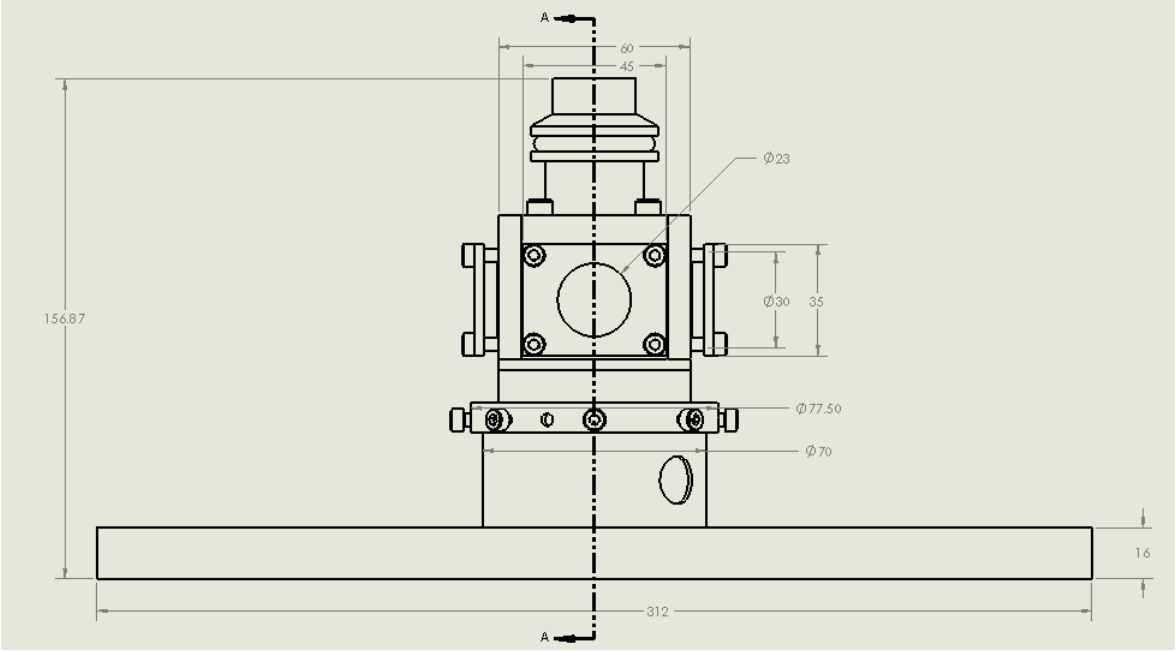


Figura 4.3: Dibujo de Conjunto PF-2J[7]

[7]Vistas de Modelo 3D del equipo PF-2J. Plano detallado en el Anexo A

Como se puede observar este equipo es relativamente pequeño, siendo el que tiene la menor capacidad energética, volumen y voltaje de umbral de toda la serie PF.

A continuación se presenta una vista en corte del equipo, junto con un listado de componentes:

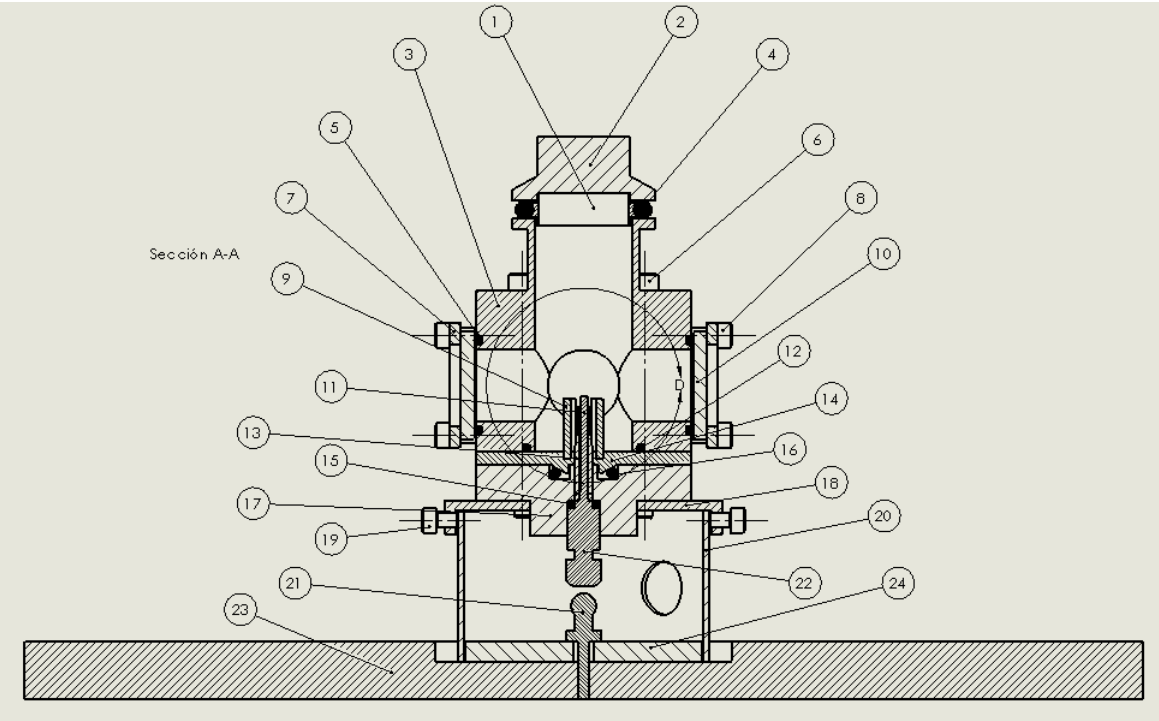


Figura 4.4: Dibujo en Corte de PF-2J[7]

Tabla 4.1: Detalle de Elementos PF-2J[8]

Referencia	Elemento		Referencia	Elemento
1	Acople O'ring		13	Aislante
2	Tapa Superior		14	Soporte Cátodos
3	Cuerpo Cámara		15	O'ring 8mm
4	O'ring 33mm		16	O'ring 14mm
5	O'ring 24mm		17	Soporte Ánodo
6	Perno M5x2.5"		18	Cubierta Spark Gap
7	Montura Visor		19	Perno M4x6mm
8	Perno M4x12mm		20	Cámara Spark Gap
9	Cátodos		21	Electrodo Inferior Spark Gap
10	Visor		22	Electrodo Superior Spark Gap
11	Ánodo		23	Condensador Plano
12	O'ring 30mm		24	Aislante Spark Gap

[7]Vistas de Modelo 3D del equipo PF-2J. Plano detallado en el Anexo A
[8]Tabla generada para detallar los elementos en el la figura 4.4

Ahora bien, en la figura 4.4 no permite ver con totalidad los elementos al interior de la cámara de descarga, por lo que se muestra el detalle en la figura 4.5.

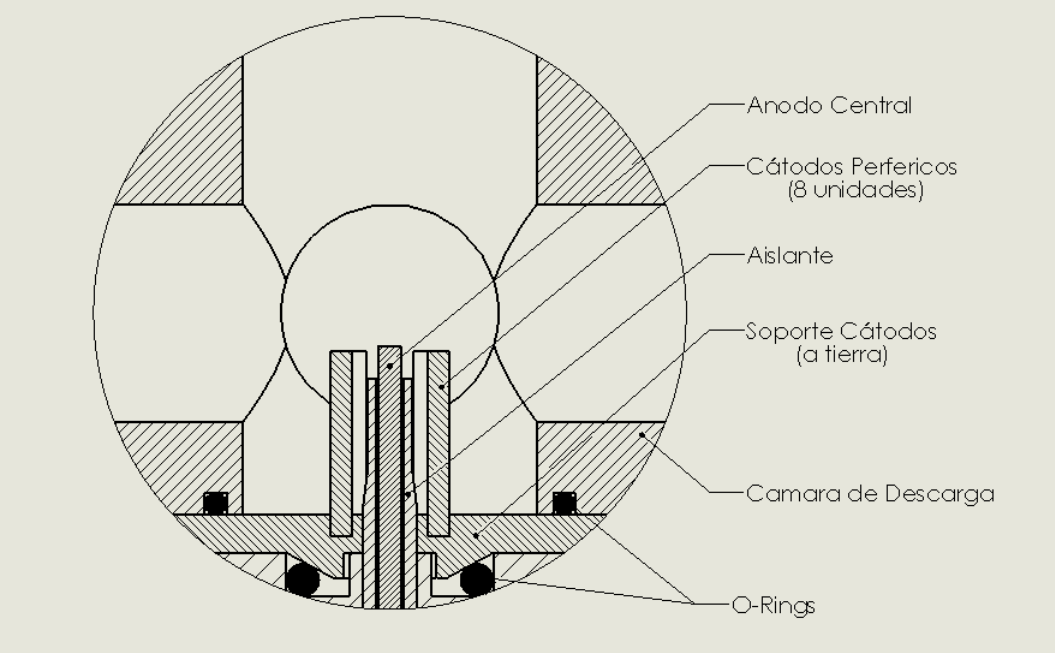


Figura 4.5: Detalle de Interior Cámara de Descarga[7]

Tal como se explica en la figura 4.7, el funcionamiento del equipo consiste en canalizar un flujo de corriente a través de los elementos claves mostrados en verde. En la figura 4.4 se pueden identificar estos elementos clave por los que fluye la corriente:

- 23 : Condensador
- $18, 20$ y 24 : Cama Spark Gap
- 21 y 22 : Spark Gap
- 11 : Ánodo
- 13 : Aislante
- $1, 2, 3, 4, 10$ y 14 : Cámara de Descarga
- 9 y 14 : Cátodos

[7]Vistas de Modelo 3D del equipo PF-2J. Plano detallado en el Anexo A

4.3. Generación de Plasma

Lo explicado anteriormente explica la estructura física del equipo, sin embargo estudiar su estructura no es suficiente para entender cómo opera y los mecanismos por los que falla.

Para lograr comprender esto vale la pena detallar el mecanismo en el que el plasma surge en el PF y como estalla. Para esto supongamos que el voltaje ha llegado al punto crítico en el condensador y este ha superado el umbral del spark-gap. Este escenario envía una corriente eléctrica de alto voltaje sobre el ánodo central.

Una vez las condiciones para generar el plasma en la cámara de descarga están satisfechas, se obtendrá un escenario como se muestra en la figura4.6. Aquí se puede ver como una línea de corriente surca a través de la atmosfera controlada, ionizando el gas cercano a este filamento y realizando una descarga hacia los cátodos con intensidad I.

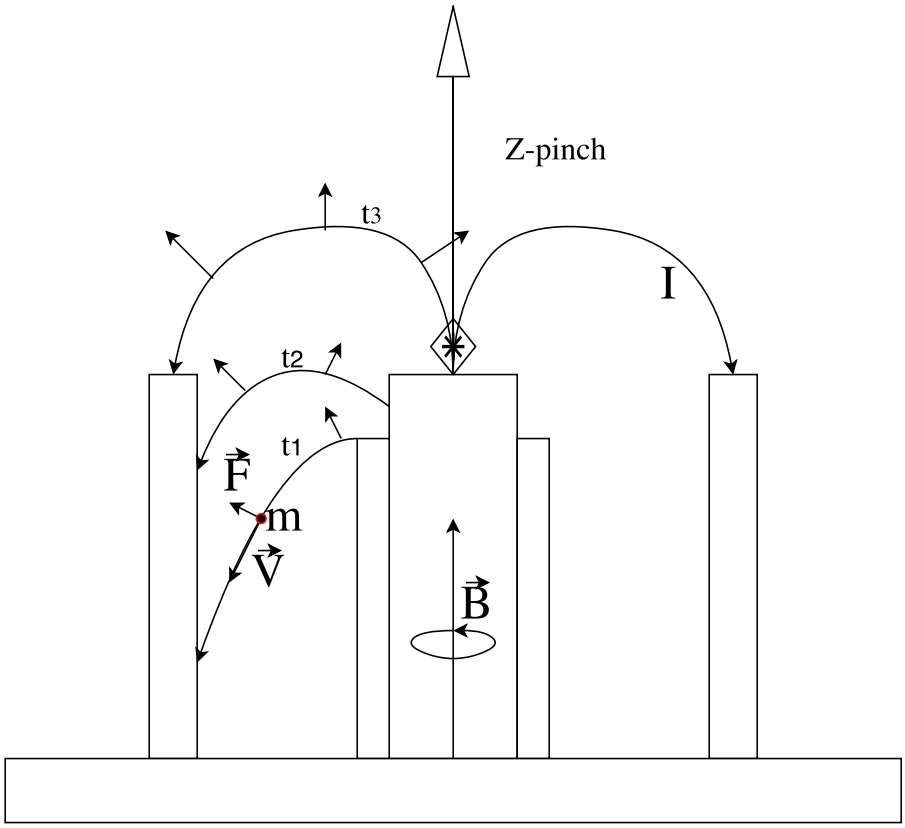


Figura 4.6: Fenómeno de z-pinch en PF[9]

Una corriente es una forma de entender el movimiento de cargas a través de un medio, en este caso este medio es el plasma. Si tomamos una de estas cargas que se desplazan hacia el cátodo, con una carga q, masa m y velocidad $\overrightarrow{V}$ se puede ver como se mueve a lo largo de su filamento en el momento t_1.

Al mismo tiempo, la corriente que surge a lo largo del ánodo central produce un campo magnético $\overrightarrow{B}$ como

[9]Diagrama generado usando `https://www.draw.io/`

se muestra en la figura. Este campo magnético impacta a la partícula m mediante la fuerza de Lorentz:

$$\overrightarrow{F} = q(E + \overrightarrow{V} \times \overrightarrow{B}) \tag{4.1}$$

Que mueve esta partícula en una dirección perpendicular a $\overrightarrow{V}$. Tomando en cuenta el movimiento caótico de las partículas ionizadas esto se traduce a una fuerza que mueve el filamento de plasma, subiendo el arco, como se puede ver en la figura en el momento t_2.

Al llegar al instante t_3 el filamento de plasma colisiona con los otros filamentos que conllevan a los distintos cátodos. Estos filamentos siguen estirándose comprimiendo la zona entre ellos, produciendo un plasma muy denso. Cabe destacar que el movimiento de los filamentos se desplaza a una velocidad varias veces la velocidad del sonido.

En estas condiciones se genera una explosión del plasma comprimido, liberando un jet de partículas verticalmente y generando fusión nuclear en algunas de ellas.

Esto explica el fenómeno de la generación del plasma en la cámara de descarga, que tiene una duración del orden de los nanosegundos y se percibe desde afuera como un sonido de estallido y una emanación de luz visiblemente purpura para el caso del H_2.

Sin embargo, para poder lograr esta generación de plasma es necesario que todo el equipo opere correctamente. También es importante comprender como los otros elementos del equipo operan y para eso se ha realizado el siguiente diagrama explicando sus interacciones:

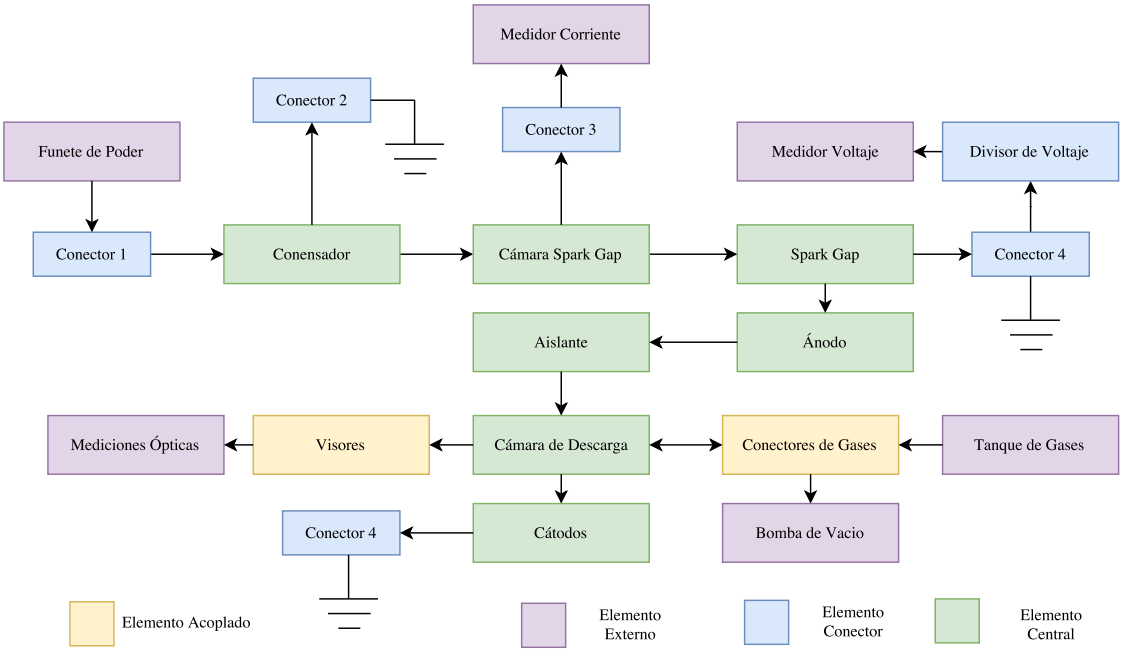


Figura 4.7: Diagrama de Bloques de PF 2J[10]

El funcionamiento del equipo considera elementos estructurales del sistema y de carácter esencial coloreados en verde en la figura 4.7. En este diagrama los bloques azules corresponden a conectores eléctricos, la

[10] Diagrama generado usando `https://www.draw.io/`

mayoría cables simples, a excepción del *Conector 4* que es un cable coaxial. En esta misma figura se consideran elementos externos al equipo PF de 2J, que se muestran en purpura.

Cabe notar que en la figura 4.7 el elemento *Conector 4* se encuentra repetido, esto se debe a que este elemento tiene la estructura de un cable coaxial. Esto quiere decir que este elemento cumple dos funciones, por una parte transmitir una línea viva del voltaje en la parte superior del Spark Gap, y por otra parte asegurar a tierra la parte externa de la estructura del equipo y los cátodos de descarga.

5. Experimentación Preliminar

5.1. Introducción

Una parte fundamental de la memoria el estudio de las señales características del equipo, es necesario contar con una gran cantidad de datos. En específico se necesitan datos que sean consistentes a lo largo de toda la vida útil del PF para correlacionar el desgaste del equipo en función del número de ciclos.

Es por esto que se requiere una experimentación para comprender la naturaleza de estas señales y así identificar parámetros clave que varían en función de la vida del producto.

Para poder modelar el desarrollo de la vida útil y su aproximación al punto de falla es necesario tener una gran cantidad de datos confiables. Por esto mismo se consideró una recopilación de datos experimentales anteriores al desarrollo de la memoria. Sin embargo, los datos a disposición en el DPT se encuentra en la forma de bitácoras de experimentación de cada equipo, por lo que explican cosas específicas del experimento y no se centran en la cantidad de veces que se ha disparado el equipo o el estado en el que se encuentra.

Junto con esto, las bitácoras presentan vacíos de largos periodos de tiempo (meses e incluso años) en los que no hay información y muestran cambios realizados al equipo a lo largo de su vida útil.

5.2. Montaje Experimental

Por las razones antes mencionadas es que se toma la decisión de no considerar las bitácoras como fuente confiable de datos de fallas y se opta por una recopilación de datos de fallas a través de un montaje experimental, en el cual se realizará una recopilación de datos a lo largo de la vida útil del equipo PF.

En este montaje se prepara la utilización del PF-2J mediante conexiones a una bomba de vacío, barómetro, tanque de hidrógeno, divisor de corriente, osciloscopio y fuente de alto voltaje. La configuración del montaje se puede apreciar de mejor forma en el siguiente diagrama:

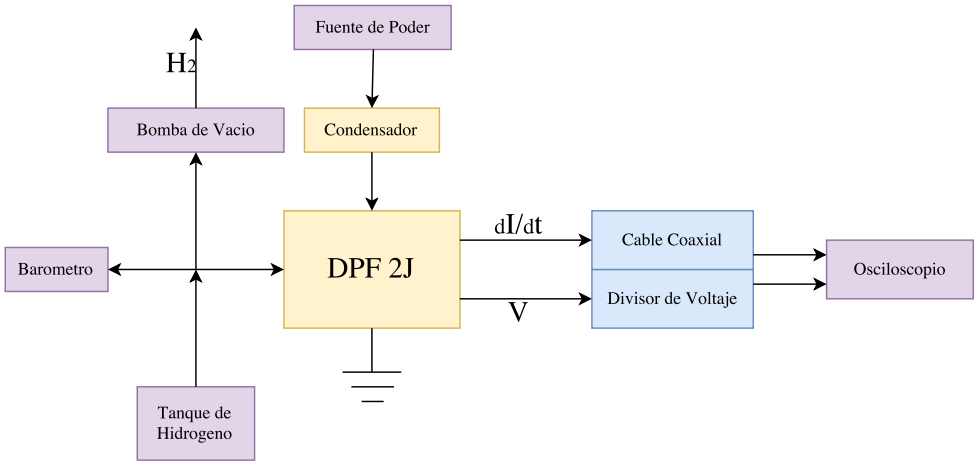


Figura 5.1: Diagrama del Montaje Experimental[11]

[11] Diagrama generado usando `https://www.draw.io/`

En la figura 5.2 se muestra la disposición de estos elementos: En el centro se encuentra el equipo PF-2J, conectado a un barómetro en la parte superior, una bomba de vacío a la izquierda y la entrada de H_2 a la derecha. Para asegurar que el osciloscopio y computador no sean dañados por las fuertes señales electromagnéticas que son liberadas en cada disparo se ha dispuesto una jaula de Faraday. Esta jaula también permite una reducción de ruido en las lecturas y se puede ver en el extremo derecho de la figura 5.2, y con mayor detalle en la figura 5.3.

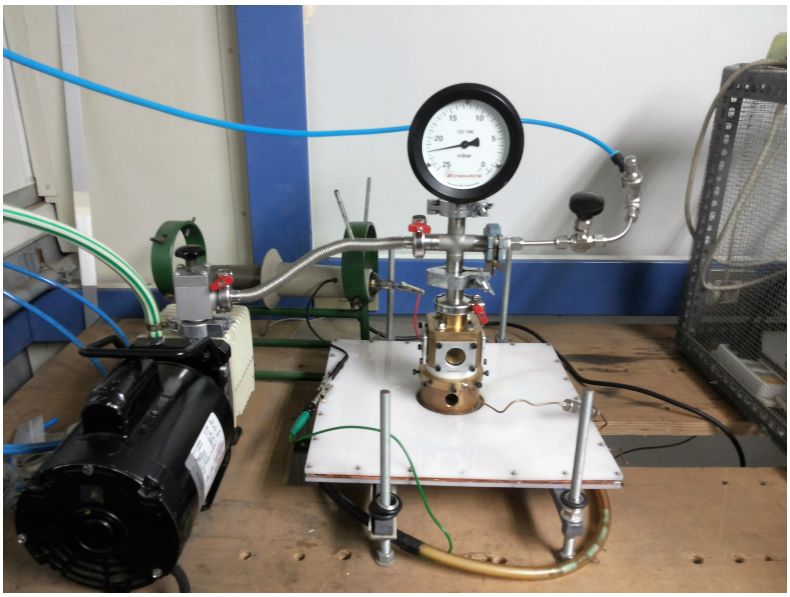


Figura 5.2: Montaje Experimental[12]

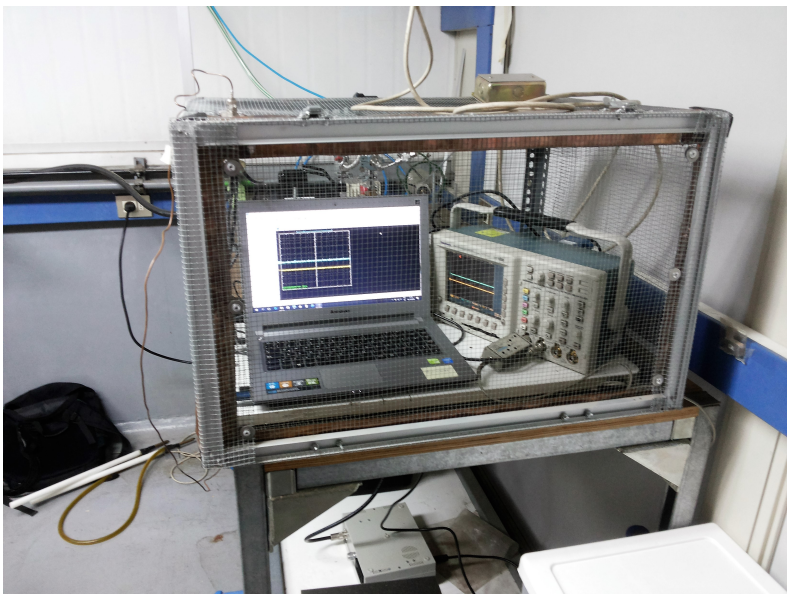


Figura 5.3: Computador y Osciloscopio Contenidos en Jaula de Faraday[12]

[12]Imágenes tomadas en las dependencias del DPTN

5.3. Resultados Preliminares

A continuación se presentan los resultados obtenidos a partir de la realización de los pasos propuestos en la sección de Metodología, en su orden respectivo.

5.3.1. Análisis FMECA

Tal como se ha explicado en los antecedentes de la sección de Introducción, FMECA es una sigla para *Failure Mode and Effects Criticality Analysis*, lo que en español quiere decir Análisis de Criticidad de Modos de Falla y Efectos.

Al realizar el análisis se consideraron una serie de elementos clave, con sus respectivos modos de falla, la severidad que conlleva este modo de falla junto con la detectabilidad de falla y probabilidad de esta.

Para hacer esto es de vital importancia establecer claramente la escala de los parámetros y que significan sus valores. Es por esto que se adjuntan los cuadros 5.1, 5.2 y 5.3:

Tabla 5.1: Tabla de Valores de Severidad de Falla[13]

Severidad	
1	Puede Alterar Resultados
2	
3	No Permite Funcionamiento Correcto
4	
5	No Permite el Funcionamiento
6	
7	Puede Causar Daño al Equipo
8	
9	Puede Causar Daño a Personas
10	

Tabla 5.2: Tabla de Valores de Probabilidad de Falla[14]

Probabilidad	
1	Extremadamente Poco Probable:
2	Menos del 0.1 % de probabilidad total
3	Remota:
4	Entre 0.1 % y 1 % de probabilidad total
5	Ocasional:
6	Entre 1 % y 10 % de probabilidad total
7	Razonablemente Probable:
8	Entre 10 % y 20 % de probabilidad total
9	Frecuente:
10	20 % o más de probabilidad total

[13] Tabla de Severidad de Falla generada en específico para PF-2J, según es recomendado en [10]

[14] Tabla de Probabilidad de Falla generada en específico para PF-2J, según es recomendado en [10]

Tabla 5.3: Tabla de Valores de Detectabilidad de Falla[15]

Detectabilidad	
1	Notorio a Simple Vista
2	
3	Detectable Mediante Observación
4	
5	Detectable con Instrumentos
6	
7	Detectable al Desensamblar
8	
9	Detectable con Instrumentos y Desensamblado
10	

A continuación se presentan las tablas 5.4 y 5.5, que muestran el análisis FMECA para el PF-2J. En la primera se especifican los elementos, sus modos de falla correspondientes y las consecuencias de falla de estos. En la siguiente tabla se especifican los valores de criticidad para cada modo de falla.

Tabla 5.4: Tabla de Análisis FMECA PF-2J[16]

Elementos Clave	**Modos de Falla**	**Consecuencias**
Condensador	Quemado	No Carga
	Corto Circuito	No Carga
Spark Gap	Corrosión Electrodos	No Alcanza Umbral
Ánodo	Corrosión	No Alcanza Umbral
Aislante	Deposición Capa Conductora	Descarga sin Dip
Cátodos	Corrosión	No Alcanza Umbral
Cámara Descarga	Filtraciones de Gases	Mediciones Alteradas
Cámara Spark Gap	Filtraciones de Gases	Mediciones Alteradas
Conectores Gases	Filtraciones de Gases	Mediciones Alteradas
Visores	Reducción de Visibilidad	Mediciones Alteradas
	Ruptura	No Aísla Medio
Conectores Eléctricos	Quemado	No Carga o No Descarga
	Corto Circuito	No Carga
Medidores	Falla Externa	Mediciones Alteradas
Fuente de Poder	Quemado	No Carga
Bomba de Vacío	Falla Externa	Mediciones Alteradas
Tanque de Gases	Falla Externa	Mediciones Alteradas

[15] Tabla de Detectabilidad de Falla generada en específico para PF-2J, según es recomendado en [10]
[16] Tabla generada a partir de modos de falla de elementos y sus consecuencias resultantes

Tabla 5.5: Tabla 2 de Análisis FMECA PF-2J[17]

Elementos	Modos de Falla	Severidad	Probabilidad	Detectabilidad	Criticidad
Condensador	Quemado	6	5	4	120
	Corto Circuito	5	6	5	150
Spark Gap	Corrosión Electrodos	5	3	4	60
Ánodo	Corrosión	5	3	3	45
Aislante	Deposición Capa Conductora	6	7	5	210
Cátodos	Corrosión	5	3	3	45
Cámara Descarga	Filtraciones de Gases	3	4	5	60
Cámara Spark Gap	Filtraciones de Gases	3	4	5	60
Conectores Gases	Filtraciones de Gases	3	4	5	60
Visores	Reducción de Visibilidad	2	3	1	6
	Ruptura	4	1	1	4
Conectores Eléctricos	Quemado	6	5	3	90
	Corto Circuito	5	5	3	75
Medidores	Falla Externa	2	4	5	40
Fuente de Poder	Quemado	5	2	5	50
Bomba de Vacío	Falla Externa	2	2	5	20
Tanque de Gases	Falla Externa	2	2	3	12

De estas tablas se puede ver claramente una tendencia de ciertos modos de falla a ser mucho más críticos que otros. Para poder ilustrar esta diferencia es recomendado utilizar la matriz de criticidad, que se puede ver en la figura 5.4:

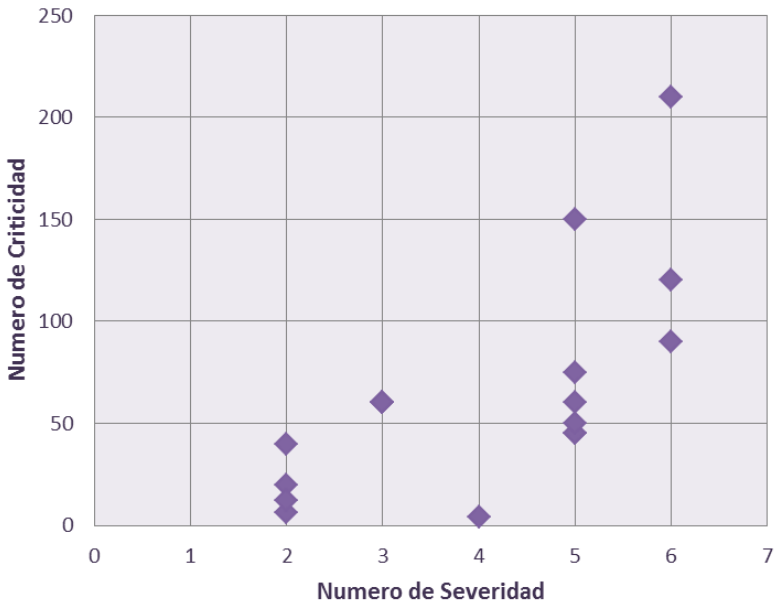


Figura 5.4: Matriz de Criticidad para PF-2J[18]

[17] Tabla FMECA generada en específico para PF-2J, según es recomendado en [10]
[18] Matriz de Criticidad generada en específico para PF-2J, según es recomendado en [10]

De este análisis preliminar se puede identificar la naturaleza de los modos de falla del sistema de forma semi cualitativa-cuantitativa. A partir de la figura 5.4 se nota que hay (a grandes rasgos) dos grupos de datos, que están separados básicamente por un valor de criticidad de 100.

Es decir, se pueden identificar los modos de falla más críticos del sistema. Un resumen de estos se puede apreciar en la tabla 5.6, donde se detallan los dos elementos más críticos:

Tabla 5.6: Elementos Críticos PF-2J[19]

Elementos Clave	Modos de Falla	Consecuencias	Criticidad
Aislante	Deposición Capa Conductora	Descarga sin Plasma	210
Condensador	Corto Circuito	No Carga	150
	Quemado	No Carga	120

5.3.2. Resultados Experimentales

Tal como se dijo anteriormente, el montaje experimental fue acomodado para poder generar dos curvas características a partir de cada disparo del equipo PF-2J. Estas curvas corresponden a la corriente que fluye a lo largo del ánodo central y que luego es dispersada a los distintos cátodos.

Los perfiles característicos del PF-2J se muestran a continuación en la figura 5.5. Cabe notar que estas curvas corresponden a una oscilación armónica amortiguada proveniente del circuito RLC que se forma en el sistema. Los siguientes perfiles característicos fueron generados usando python para implementar un sistema de reconocimiento detallado más adelante.

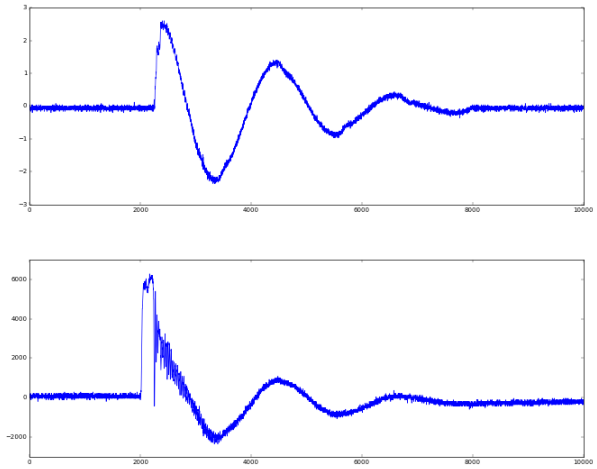


Figura 5.5: Señales Características: Derivada de corriente (superior) y Voltaje (inferior), en función del tiempo[20]

[19] Tabla generada para resumir elementos críticos
[20] Imagen tomada de programa desarrollado en Anexo B

Estas señales difieren levemente en el momento de efectuarse un pinch, que es necesario para que sea efectiva la fusión nuclear. El 'pellizcamiento' de plasma se evidencia en las curvas teniendo un pequeño quiebre en cierto tramo de la curva. El lugar de la curva donde se genera es casi siempre el mismo, pero eventualmente se presenta en otros momentos de tiempo. Un ejemplo característico de este fenómeno se puede ver en la figura 5.6.

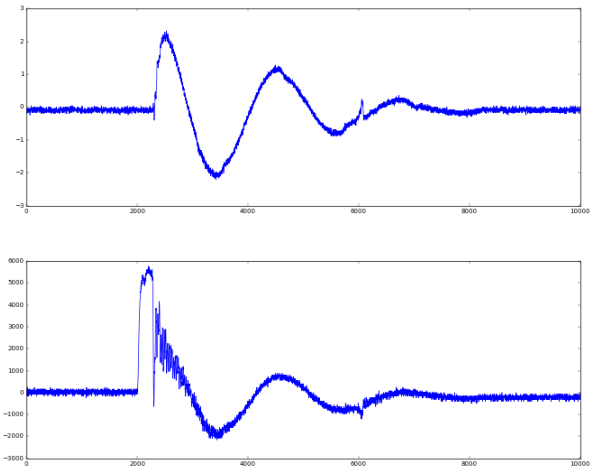


Figura 5.6: Señales Características con 'Dip': Derivada de corriente (superior) y Voltaje (inferior), en función del tiempo[21]

Los datos que presentan este comportamiento sólo pueden ser verificaos mediante una inspección visual de los perfiles característicos del disparo. Naturalmente habrá que reconocer cada uno de los disparos para comprender cuando es que se genera este fenómeno y cuando deja de suceder, concluyendo la vida útil del aislante.

Es por esto que el experimento fue hecho durante varias jornadas de toma de datos, hasta notar que el fenómeno 'dip' sea escaso o nulo. Cabe recordar que este fenómeno es lo que se busca estudiar en el DPTN y por ende es crucial entender la confiabilidad del equipo en función de la factibilidad y probabilidad de que ocurra 'dip' en el siguiente disparo. Teniendo esto en cuenta, los experimentos fueron realizados culminando en un total de $8{,}579\ disparos$.

Para poder diferenciar estos datos durante el estudio del trabajo de título y para poder ser usado en la implementación del sistema de mantenimiento se ha realizado un programa en $python$, usando $PyQt5$ para generar una ventana GUI interactiva. En el Anexo B se puede apreciar el código utilizado, que culmina en la ventana de la figura 5.7:

[21] Imagen tomada de programa desarrollado en Anexo B

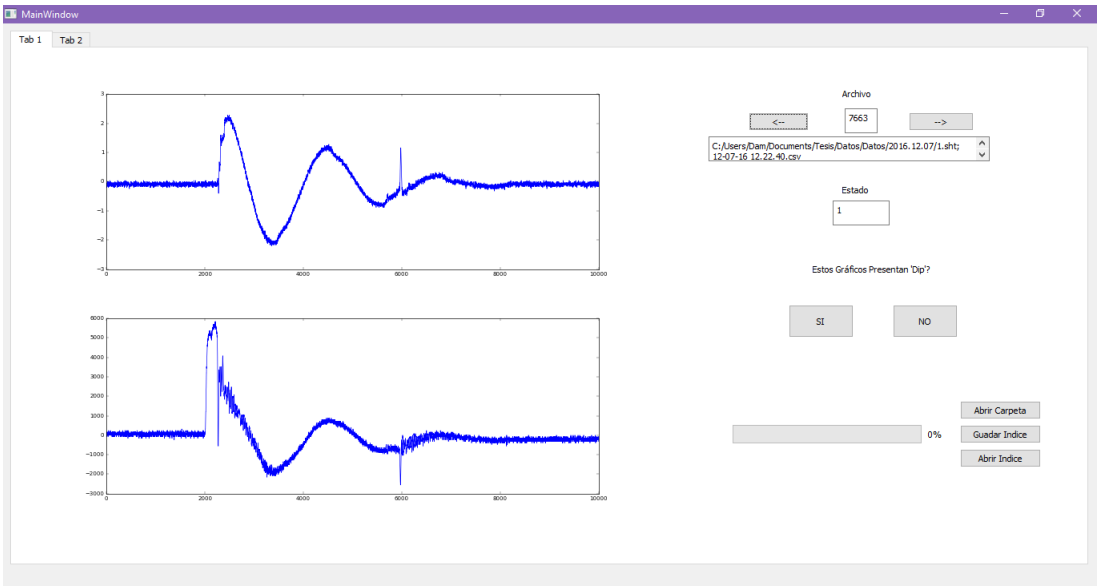


Figura 5.7: GUI Desarrollada para Identificar 'Dip'[22]

En este programa se selecciona una carpeta con los datos experimentales, divididos en subcarpetas diarias, siendo cada dato en formato '.csv'. Archivos con otras extensiones y archivos fuera de las subcarpetas no son tomados en consideración. Luego de seleccionados los datos el programa crea un índice que contiene el nombre de cada archivo, el número del disparo al que corresponde este y si se presenta 'dip' o no según tres posibles valores: 0, 1 y $pendiente$.

Cuando un archivo es categorizado esto se hace presente en el índice y se mueve al siguiente dato. Naturalmente cuenta con la posibilidad de volver a un dato anterior y avanzar a un dato siguiente. Después de haber categorizado suficientes datos se puede guardar el progreso en un archivo '.csv' que contiene la totalidad del índice, que posteriormente puede ser cargado y continuado en el primer archivo $pendiente$ del índice.

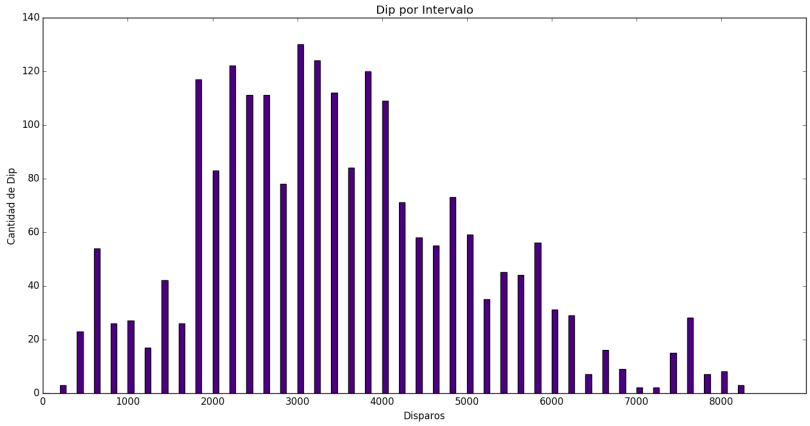


Figura 5.8: Distribución Estadística del Dip (cada barra corresponde a 200 disparos)[23]

[22] Imagen de programa desarrollado en Anexo B
[23] Gráfico generado en python a partir de resultados de programa en Anexo B

A partir de este análisis se pueden agrupar los datos en rangos de disparos, dentro de los cuales pueden ser contadas las veces que estos perfiles presentan 'dip'. Considerando un tramo de $200\ disparos$, en la figura 5.8 se puede ver cómo cambia la probabilidad de efectuarse 'dip' en función de los disparos, evidenciando la vida útil completa del producto.

Ahora bien, este análisis permite comprender el avance del estado del equipo a medida que se dispara mas veces, sin embargo, para poder predecir el estado del equipo es fundamental estudiar como los perfiles característicos se modifican al haber desgaste.

Por esto mismo se ha desarrollado otro programa en $python$ que permite analizar los perfiles y obtener así parámetros que lo describan en varios sentidos. Aquí entra el concetpo $Feature\ Extraction$, tal como es presentado en [6], donde se realizan análisis de datos sobre perfiles. En este caso, los parámetros escogidos son los siguientes:

- Kurtosis

$$K = \frac{\sum_{j=1}^{N}(n_j - u)^4}{(N-1)\sigma^4} \tag{5.1}$$

- Crest Indicator

$$CI = \frac{max|n|}{\sqrt{\frac{1}{N}\sum_{j=1}^{N}(n_j)^2}} \tag{5.2}$$

- Shape Indicator

$$SI = \frac{\sqrt{\frac{1}{N}\sum_{j=1}^{N}(n_j)^2}}{\frac{1}{N}\sum_{j=1}^{N}|n_j|} \tag{5.3}$$

- Clearance Indicator

$$CI = \frac{max|n|}{(\frac{1}{N}\sum_{j=1}^{N}\sqrt{|n_j|})^2} \tag{5.4}$$

- Máximo Global

$$Max = max(n_j) \tag{5.5}$$

- Mínimo Global

$$Min = min(n_j) \tag{5.6}$$

- Amortiguamiento

  Este valor fue calculado mediante un análisis de Fourier de los perfiles, utilizando el método 'Peak Picking' desarrollado en [8]. Una vez hecha la transformada de Fourier el amortiguamiento puede ser calculado usando la ecuación 5.7, donde las variables claves se pueden apreciar en la figura 5.9:

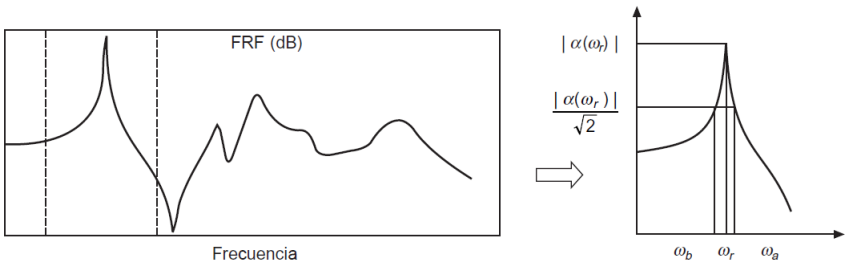


Figura 5.9: Peak Picking [24]

$$\xi_r = \frac{\omega_b^2 - \omega_a^2}{4\omega_r^2} \approx \frac{\omega_b - \omega_a}{2\omega_r} \tag{5.7}$$

- Shannon Entropy

  Para calcular este parámetro se utiliza también la transformada de Fourier, ya que esta permite determinar la frecuencia de ocurrencia de cada frecuencia de onda en la señal. De este modo se puede asignar la probabilidad que cierta frecuencia predomine, normalizando el espectro de Fourier a la unidad, luego de esto la entropía de la señal se calcula probabilísticamente usando la ecuación 5.8.

$$H = -\sum_i p_i ln(p_i) \tag{5.8}$$

Las ecuaciones 5.1 a 5.8, la variable σ corresponde a la desviación estándar, u corresponde al promedio de los datos, N es la cantidad total de datos y n_j corresponde al valor $j - esimo$ de los datos. Esta extracción de parámetros nos permitirá observar como estos cambian al avanzar el número de disparos del equipo, posiblemente estableciendo una correlación con el desgaste del mismo.

Para poder calcular todos estos parámetros se ha escrito un programa en $python$ que permita abrir una carpeta que contiene las señales de cada disparo durante la vida útil. Este programa se puede ver con mayor detalle en el Anexo C.

A continuación se muestran los resultados de la evolución de los parámetros antes señalados en función del número de disparos. En la figura 5.10 se muestran los parámetros de Kurtosis y de Crest Indicator para ambos perfiles, en la figura 5.11 se muestran los parámetros de Shape Indicator y de Clearance Indicator tanto para el voltaje como para la derivada de corriente. Análogamente, en la figura 5.12 se muestran los Máximos y Mínimos globales para ambos perfiles y en la figura 5.13 se muestra la entropía de señal y el amortiguamiento de la misma.

[24] Imagen extraída de [8]

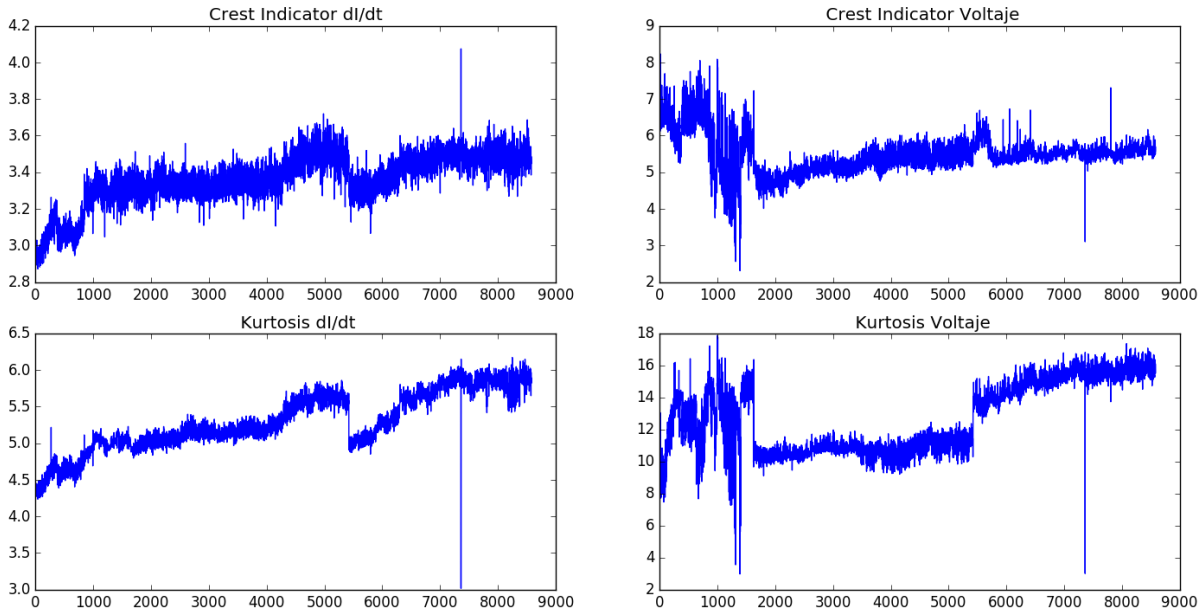


Figura 5.10: Kurtosis y Crest Indictor en función de numero de disparos[25]

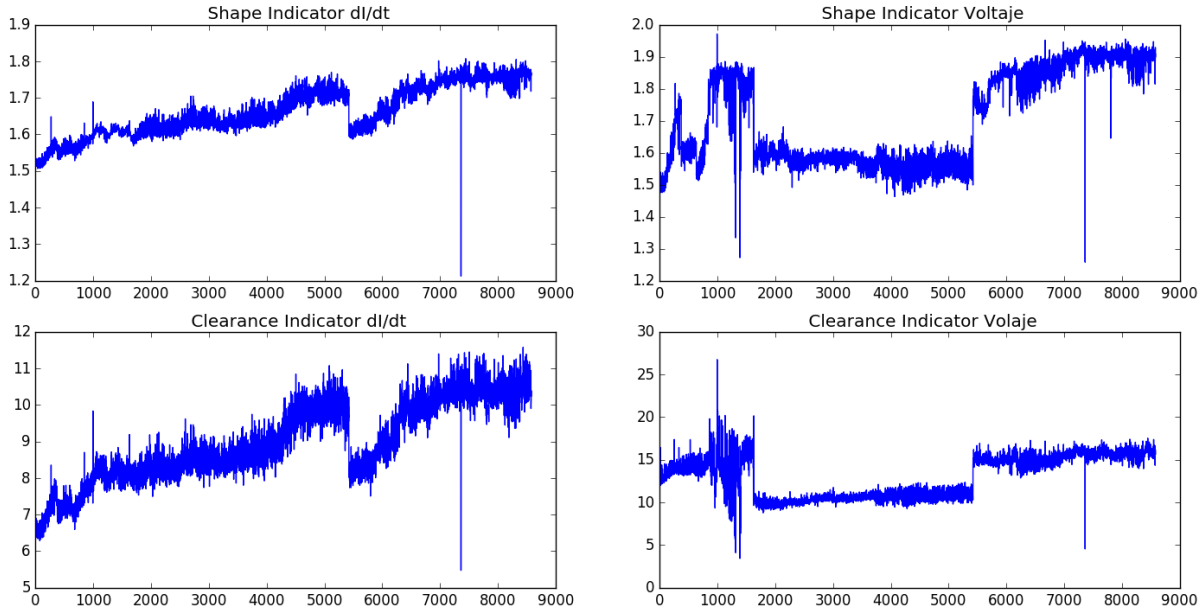


Figura 5.11: Shape Indicator y Clearance Indicator en función de numero de disparos[25]

[25]Figuras generadas en python a partir de resultados del programa detallado en el AnexoC)

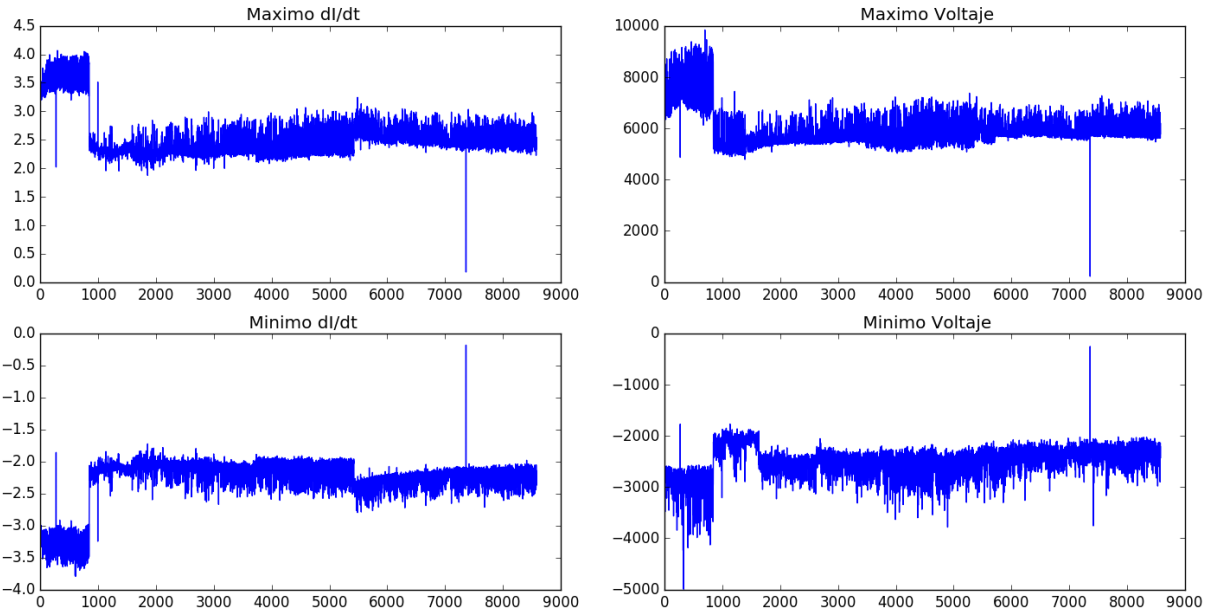


Figura 5.12: Máximos y Mínimos globales en función de numero de disparos[25]

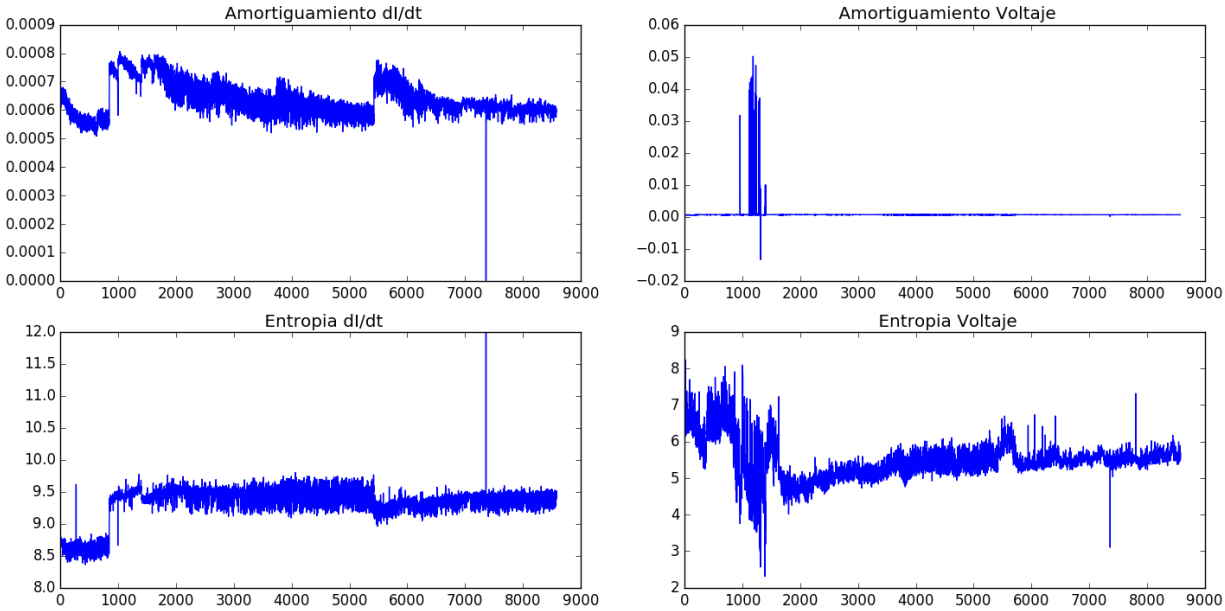


Figura 5.13: Amortiguamiento y Entropía de Shannon en función de numero de disparos[25]

[25]Figuras generadas en python a partir de resultados del programa detallado en el AnexoC)

5.4. Análisis de Resultados Preliminares

De los resultados mostrados anteriormente se pueden notar varios quiebres en los datos, por ejemplo en las vecindades de $N = 1000$ para los gráficos 5.12 y 5.13. Esto se debe a que la distancia del spark-gap fue modificada en ese momento con la finalidad de obtener un voltaje máximo del orden de $6kV$, debido a que el equipo está especificado para operar de forma óptima a este voltaje.

Por otro lado, el gráfico 5.8 permite visualizar de una forma bastante completa el deterioro del equipo y, tal como fue comentado por los científicos del laboratorio, el proceso de mejora del equipo durante los primeros ciclos. Tomando en cuenta las discontinuidades de operación antes señaladas que pueden afectar la forma de este gráfico, se puede ver una clara tendencia a un perfil probabilístico normal. También el gráfico puede corresponder a una distribución de Weibull, ya que el decaimiento hacia la derecha del punto de máxima probabilidad de Dip es menos pronunciado que el crecimiento a la izquierda de este.

Para realizar el experimento se deben fijar ciertos parámetros de experimentación, como por ejemplo la presión al interior de la cámara de descarga y el voltaje de carga del condensador. Estos parámetros, en especial la presión, son críticos en el momento de encontrar el Dip en los datos. Los científicos explican que en un inicio de una jornada de experimentación se busca la presión óptima para que el fenómeno sea más dominante.

Además, al acortar la distancia del spark-gap el umbral requerido para iniciar la descarga es reducido, lo que impacta en los resultados. Para poder modificar esta distancia (después de los 1000 disparos fue modificada a $3mm$) se separan los elementos $20, 21, 23$ y 24 del resto de ellos, soltando los pernos (elemento 19), los elementos son numerados anteriormente en la figura 4.4 y tabla 4.1. Esto permite acceder al elemento 21, que mediante contratuercas se fija a una altura arbitraria.

En el comienzo de esta experimentación se tomaron cientos de datos con distintas presiones en el rango de $25 - 6$ $mbar$. Posteriormente se estableció que la presión optima, es decir, donde el fenómeno de Dip era más recurrente era $P = 8mbar$. Posterior a esto se continuó utilizando solo esta presión.

Este procedimiento causa fuentes de error en los datos y debe ser evitado dentro de lo posible, sin embargo el elemento 21 tuvo que ser reajustado en las vecindades de $N = 5500$ debido a que el electrodo de carga del condensador había sido desacoplado. Aunque esta vez no se modificó la altura del spark-gap, la manipulación de este elemento muestra un impacto similar en los resultados, en especial en los gráficos 5.10, 5.11 y 5.13.

Estos casos demuestran la sensibilidad del equipo ante alteraciones externas, sea por motivos de reparación, ajuste o accidentales. Sin embargo, ciertos perfiles de los parámetros extraídos si muestran una clara tendencia a un crecimiento o decrecimiento al avanzar en número de disparos, correlacionándolos con el desgaste del equipo.

En específico, los parámetros Crest Indicator, Kurtosis, Shape Indicator y Clearance Indicator muestran un crecimiento constante hasta llegar al quiebre de $N = 5500$, después del cual continua el comportamiento creciente. Del mismo modo, el amortiguamiento tiene el comportamiento inverso, decrece hasta el quiebre para luego continuar su comportamiento decreciente. El quiebre en este momento fue causado por una accidental alteración del equipo por terceros, que requirió desarmar el equipo de forma similar a lo hecho a $N = 1000$.

En resumen, los parámetros Crest Indicator, Kurtosis, Amortiguamiento, Shape Indicator y Clearance Indicator muestran un comportamiento de variación monótona y pueden ser analizados de acuerdo a los momentos críticos de esta experimentación, siendo ellos $N = 0$, $N = 1000$, $N = 5500$ (antes y después del quiebre) y

$N = 8500$ (fin de vida útil).

Cabe destacar que las tendencias se notan en los perfiles correspondientes a $\frac{dI}{dt}$ con mayor definición que en los de V. Esto se debe a que en el montaje experimental la señal de $\frac{dI}{dt}$ se encuentra aislada por un cable coaxial que se conecta directamente a la jaula de Faraday, reduciendo el ruido. En cambio, la señal de V se conecta directamente al osciloscopio, operando como una suerte de antena que percibe las ondas electromagnéticas provenientes del disparo que son de una potencia no despreciable. Por eso mismo es que en los gráficos 5.5 se puede notar como el perfil del voltaje contiene una gran cantidad de ruido al inicio de la formación del plasma, a diferencia del perfil de derivada de corriente.

Conforme a esto y a los gráficos de todos los parámetros, se concluye que los parámetros asociados a las curvas de voltaje no entregan valores que puedan ser correlacionados al desgaste, al menos en esta experimentación preliminar.

Para poder comprender la relación entre el cambio de estos parámetros y la degradación del PF-2J es fundamental poder comparar los resultados en los momentos críticos del experimento. Como se observa en la figura 5.8, el equipo se encuentra en una etapa de degradación avanzada en el momento que ocurrió el quiebre de los parámetros, esto es a una cantidad de disparos de aproximadamente $N = 5500$. Sin embargo el equipo no se encuentra degradado completamente hasta alcanzar $N = 8500$, por lo que vale la pena comparar los parámetros en todos estos puntos críticos, un resumen de estos parámetros se puede apreciar en la tabla 5.7.

Tabla 5.7: Variación de Parámetros Durante la Vida del Equipo[26]

Disparos	Crest Indicator	Kurtosis	Amortiguamiento $(\times 10^{-4})$	Shape Indicator	Clearance Indicator
N=0	2.95	4.4	6.5	1.52	6.6
N=1000	3.25	4.9	7.8	1.60	8.0
N=5500$^-$	3.5	5.5	5.8	1.71	9.9
N=5500$^+$	3.3	5.0	6.9	1.60	8.2
N=8500	3.5	5.9	5.9	1.76	10.5

Considerando que el experimento presentó instancias de Dip desde las vecindades de $N = 1000$, se considera este momento ($N = 1000$) como el punto de referencia del inicio de la degradación, ya que los valores anteriores a este corresponden a datos de ajuste. Teniendo esto en cuenta, en la tabla 5.8 se compara el cambio proporcional al punto de referencia, siendo F un parámetro del sistema, se define el cambio porcentual como en la ecuación 5.9 .

$$\Delta F_x = \frac{F(N_x) - F(N = 1000)}{F(N = 1000)} \times 100 \tag{5.9}$$

[26]Tabla generada a partir de observación detallada de los gráficos de la sección de resultados preliminares

Tabla 5.8: Cambio Porcentual de Parámetros[27]

Parámetro	$N = 1000$ a $N = 5500$	$N = 1000$ a $N = 8500$	$N = 5500$ a $N = 8500$
$\Delta Crest$	7.69 %	7.69 %	0 %
$\Delta Kurtosis$	12.24 %	20.41 %	8.17 %
$\Delta Amortiguamiento$	-25.64 %	-24.36 %	1.28 %
$\Delta Shape$	6.87 %	10.00 %	3.13 %
$\Delta Clearance$	23.75 %	31.25 %	7.5 %

A partir de la tabla 5.8 se puede notar como los parámetros Crest Indicator y Amortiguamiento alcanzan sus valores máximo y mínimo antes de finalizar la vida útil del producto. Por otro lado los parámetros Kurtosis, Shape Indicator y Clearance Indicator muestran una variación mayor en esta última etapa de degradación (entre $N = 5500$ y $N = 8500$) .

Esto permitiría identificar ciertos cambios porcentuales de los parámetros para poder correlacionar distintas etapas de vida, haciendo posible una predicción sobre el estado del sistema. A pesar que los parámetros cambian rapidamente hasta el estado deteriorado de $N = 5500$, otros parámetros continuan en aumento hasta finalizar el ciclo de vida en $N = 8500$.

[27] Tabla generada a partir de los valores de la tabla 5.7

6. Análisis de Correlación de Datos

Según los datos que surgen a partir del experimento preliminar se pueden establecer relaciones entre la evolución de los distintos parámetros característicos en el tiempo con el avance en la vida útil del equipo.

Para abordar el modelo que permite correlacionar estos estados se puede establecer una relación analítica entre desde un espacio multidimensional, los parámetros, hacia un espacio unidimensional, el desgaste. Para esto se puede denotar $\overrightarrow{F}$ al vector que contiene todos los parámetros o $features$.

Sin embargo, para que este análisis sea menos dependiente de este experimento y pueda ser realizado para otras condiciones experimentales, así como otros equipos se consideran que estos parámetros se encuentran normalizados. Es decir, cada feature será trasladada para que comience desde 0 cuando se encuentre en su punto de inicio, que es considerado como referencia a la configuración del experimento ($N = 0$, $N = 1000$ como en el experimento preliminar, u otro determinado por la estabilización inicial del equipo).

Esto quiere decir que el vector $\overrightarrow{F}$ mide el incremento porcentual de cada parámetro para un numero de disparos arbitrario desde el inicio del experimento hasta que el equipo haya cumplido su vida útil:

$$\overrightarrow{F} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ \vdots \\ F_n \end{bmatrix} \quad \epsilon \ \Re^n \tag{6.1}$$

Entonces, suponiendo que existe una función $Y(\cdot)$ que cumple:

$$\begin{aligned} Y(\cdot) : \Re^n &\rightarrow \Re \\ Y(\overrightarrow{F}) &= N \end{aligned} \tag{6.2}$$

Donde $\overrightarrow{F}$ es el vector de features (es decir, Kurtosis, Crest Indicator, Amortiguamiento, ...) y N es el numero de disparos que ha ejecutado el equipo desde el inicio del experimento hasta un tiempo arbitrario. Esa función Y es lo que se busca encontrar o, más bien, aproximar mediante modelos matemáticos y computacionales.

6.1. Estimación de N por Combinación Lineal

La primera aproximación para acercarnos a esta función se puede optar por un comportamiento lineal. A pesar de que haya presente una pérdida de información dada por la aproximación de las curvas 5.10 a 5.13 a un comportamiento lineal, el hecho que el fenómeno se vea en varias dimensiones permite eliminar fuentes de ruido, estableciendo una aproximación simple y directa.

Para esto, se reduce el ruido de las curvas $F_i(N)$ utilizando la librería $Signal$ de python utilizando la función $signal.lfilter()$ y luego una interpolación lineal de tal forma de obtener una resolución analítica para la estimación de N, para cada una de las F_i. Es decir:

$$\overrightarrow{N}_{estimado} = \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ \vdots \\ N_n \end{bmatrix} = \begin{bmatrix} y_1(F_n) \\ y_2(F_n) \\ y_3(F_n) \\ \vdots \\ y_n(F_n) \end{bmatrix} \tag{6.3}$$

Donde $y_i = m_i \cdot F_i + n_i$ representa la ecuación de la recta interpolada para los gráficos inversos de $F_i(N)$.

De esta manera se puede obtener una estimación lineal del numero de disparos efectuados para cada parámetro. Sin embargo, ahora se tienen muchas estimaciones para el mismo resultado. Para esto se considera que es posible realizar una estimación global de N como una combinación lineal de las estimaciones de cada feature:

$$\begin{gathered} N_{est} = C_1 \cdot N_1^{est} + C_2 \cdot N_2^{est} + C_3 \cdot N_3^{est} + ... + C_n \cdot N_n^{est} \\ N_{est} = \sum_i C_i \cdot N_i^{est} \end{gathered} \tag{6.4}$$

En esta ecuación se conocen todos los N_i^{est} para cada uno de los n parámetros y N disparos, por lo que se pueden encontrar los coeficientes que mejor representan la realidad, contrastándolos con el numero real de disparos para el set de parámetros. Para lograr encontrar los coeficientes C_i de cada uno de estos parámetros se produce un problema crucial en el cambio de dimensionalidad, debido a que n corresponde al numero total de $features$ y N al numero total de $disparos$, hay $n \sim 10$ incógnitas y $N \simeq 8000$ ecuaciones. Por lo que se propone resolver el problema como si hubiesen n ecuaciones, siendo estas las primeras n, luego resolverlo nuevamente con las ecuaciones n a $2n$ y así sucesivamente hasta resolver todas las N ecuaciones utilizando N/n combinaciones lineales:

$$\begin{gathered} 1 = C_1 \cdot N_1^{est}(1) + C_2 \cdot N_2^{est}(1) + ... + C_n \cdot N_n^{est}(1) \\ 2 = C_1 \cdot N_1^{est}(2) + C_2 \cdot N_2^{est}(2) + ... + C_n \cdot N_n^{est}(2) \\ 3 = C_1 \cdot N_1^{est}(3) + C_2 \cdot N_2^{est}(3) + ... + C_n \cdot N_n^{est}(3) \\ ... \\ n = C_1 \cdot N_1^{est}(n) + C_2 \cdot N_2^{est}(n) + ... + C_n \cdot N_n^{est}(n) \end{gathered} \tag{6.5}$$

Y para las n siguientes ecuaciones lo mismo:

$$\begin{gathered} n+1 = C_1 \cdot N_1^{est}(n+1) + C_2 \cdot N_2^{est}(n+1) + ... + C_n \cdot N_n^{est}(n+1) \\ n+2 = C_1 \cdot N_1^{est}(n+2) + C_2 \cdot N_2^{est}(n+2) + ... + C_n \cdot N_n^{est}(n+2) \\ n+3 = C_1 \cdot N_1^{est}(n+3) + C_2 \cdot N_2^{est}(n+3) + ... + C_n \cdot N_n^{est}(n+3) \\ ... \\ 2n = C_1 \cdot N_1^{est}(2n) + C_2 \cdot N_2^{est}(2n) + ... + C_n \cdot N_n^{est}(2n) \end{gathered} \tag{6.6}$$

De esta manera se obtienen N/n sistemas lineales que se pueden resolver para obtener los n coeficientes C_i, N/n veces. Estos sistemas pueden expresarse de forma matricial para ser resueltos computacionalmente

de forma mas sencilla:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ n \end{bmatrix} = \begin{bmatrix} N_1^{est}(1) & N_2^{est}(1) & ... & N_n^{est}(1) \\ N_1^{est}(2) & N_2^{est}(2) & ... & N_n^{est}(2) \\ N_1^{est}(3) & N_2^{est}(3) & ... & N_n^{est}(3) \\ & & \vdots & \\ N_1^{est}(n) & N_2^{est}(n) & ... & N_n^{est}(n) \end{bmatrix} \times \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ \vdots \\ C_n \end{bmatrix} \tag{6.7}$$

$$\overrightarrow{N}_{real} = \overrightarrow{N}_{est} \times \overrightarrow{C} \tag{6.8}$$

Finalmente, para estimar los parámetros C_i que con mayor frecuencia resuelven estas ecuaciones simplemente son promediados para encontrar los n coeficientes finales que describen de mejor forma esta combinación lineal:

$$C_i = \frac{C_i(1,n) + C_i(n,2n) + C_i(2n,3n) + ... + C_i(N-n,N)}{N/n} \tag{6.9}$$

De esta manera se puede estimar el numero de disparos efectuados por el equipo mediante una simple combinación lineal de parámetros que pueden ser calculados de forma sencilla a partir de los mismos datos que los usuarios del equipo obtienen cada vez que lo operan.

$$N_{est} = \sum_i C_i \cdot F_i \tag{6.10}$$

Utilizando los datos de la experimentación preliminar se pueden encontrar las predicciones de N_{est} resolviendo el sistema lineal. A continuación se muestran los resultados de correlación entre los features. En este caso se han utilizado los features Kurtosis, Crest Indicator, Shape Indicator y Amortiguamiento, que se pueden ver en las figuras 5.10, 5.11 y 5.13, respectivamente. El problema algebraico se ha resuelto recorriendo los datos mediante una rutina en *python*, resultando en la figura 6.1. La recta roja corresponde al numero de disparos reales ($f(x) = x$), así comparando la estimación con la realidad.

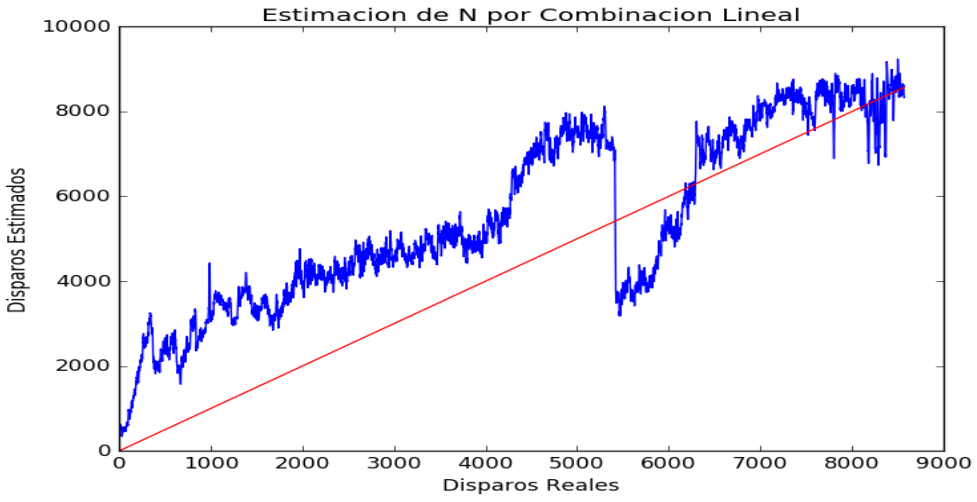


Figura 6.1: Estimaciones de N Mediante Combinación Lineal de Parámetros[28]

[28] Gráfico generado a partir de programa en python, siguiendo una combinación lineal de parametros

6.2. Estimación de N Mediante Redes Neuronales

Considerando que un análisis lineal es simple, rápido y da buenos resultados, es un análisis quizás muy básico para modelar un problema tan complejo. No solo estamos extrapolando los datos para alejarnos de la sensibilidad real una vez, sino que dos veces, gracias a las estimaciones de N_i^{est} y de C_i.

Esto conlleva a una pérdida de información que puede ser aprovechada si se supiera exactamente cual es la relevante y cual es ruido. Sin embargo, esto resulta una tarea extremadamente difícil en sistemas complejos como el estudiado en este documento.

Es por esto que se plantea un modelo de redes neuronales (en inglés Neural Networks, aquí referenciado como NN) que permitan relacionar de forma iterativa los datos recopilados, para así aproximar la función ideal $Y(\overrightarrow{F})$. En este estudio se utilizará una red neuronal de 4 capas, 2 visibles y 2 invisibles, tal como se muestra en la figura 6.2.

Los pesos de la red neuronal son las variables de estado del sistema, estos determinan cual es el resultado estimado para cada valor de $\overrightarrow{F}$. Estos son contenidos en las matrices W_0, W_1 y W_2, que determinan el estado de computo de la red después de entrenamiento prolongado.

Por simplicidad se optaron la cantidad de nodos en las capas 1 y 2 como la mitad de nodos de la capa anterior, así asegurando una transición suave en la reducción dimensional. Por lo tanto las dimensiones de las matrices W_0, W_1 y W_2 son $(n \times \frac{n}{2})$, $(\frac{n}{2} \times \frac{n}{4})$ y $(\frac{n}{4} \times 1)$, respectivamente.

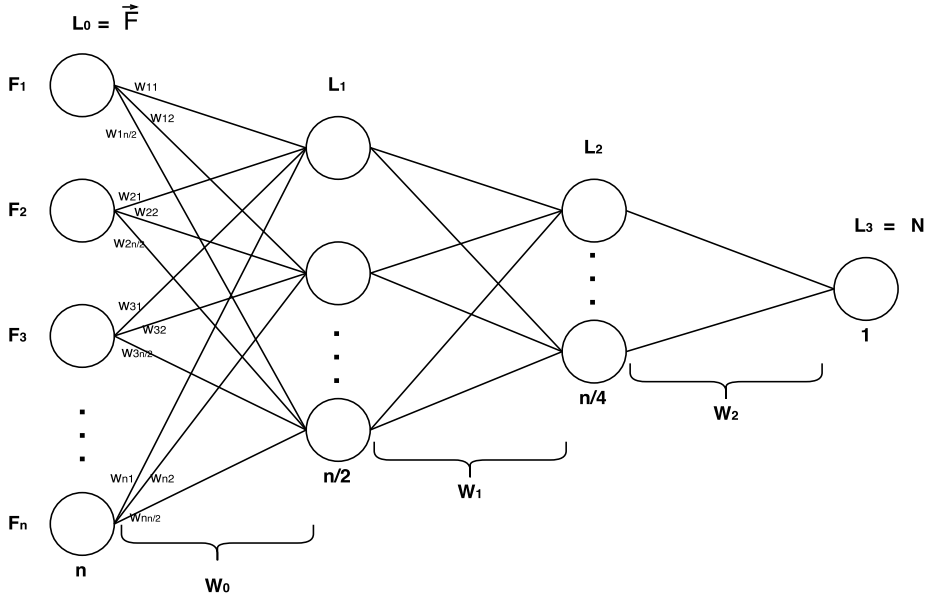


Figura 6.2: Diagrama de Red Neuronal Implementada[29]

Esta red neuronal consiste en un $Perceptron$ básico de dos capas ocultas, donde cada neurona de las capas ocultas actúa según la función sigmoide de la combinación lineal de cada F_i ponderada por cada uno de sus $pesos$, w_{ji}. Es decir:

[29] Red neuronal implementada en la regresión, figura generada en `draw.io`

$$L_1[i] = \sigma\left(\sum_j w_{ji} \cdot L_0[j]\right) = \sigma\left(\sum_j w_{ji} \cdot F_j\right) \tag{6.11}$$

$$L_2[i] = \sigma\left(\sum_j w'_{ji} \cdot L_1[j]\right) \tag{6.12}$$

$$N = L_3 = \sigma\left(\sum_j w''_j \cdot L_2[j]\right) \tag{6.13}$$

Donde la función sigmoide está definida como:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{6.14}$$

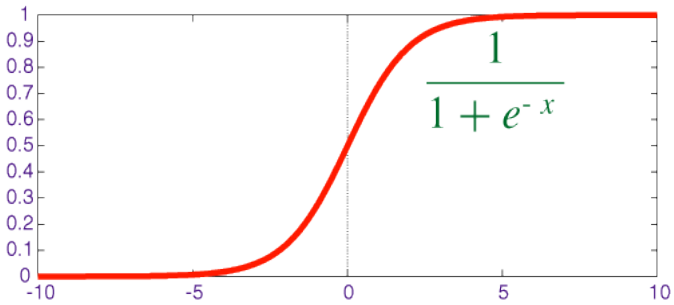


Figura 6.3: Sigmoide[30]

Esta función es comúnmente escogida como la función de una neurona base para una amplia gama de redes neuronales, debido a su habilidad de abarcar problemas no lineales.

Las redes neuronales operan inicialmente utilizando unas matrices de pesos generadas al azar, luego el error es calculado de acuerdo a la predicción que hizo contrastado con el valor real proporcionado de los datos de entrenamiento. Este error se propaga desde la ultima capa hasta cada una de los nodos, permitiendo saber cual el impacto de cada peso al error final mediante $Backpropagation$.

Este método permite saber la variación de un peso arbitrario w_i mediante el calculo de la derivada de el error respecto a este peso en específico. Debido a que casa operación entre neuronas es o una combinación lineal de neuronas anteriores o bien una función sigmoide, esta derivada siempre puede ser encontrada. Debido a que estos pesos están representados en matrices, $backpropagation$ funciona calculando el gradiente de estas.

Posteriormente el gradiente es utilizado para modificar el peso arbitrario w_i, reduciéndolo en la derivada de la función que le sigue (respecto al error) por el error y por el resultado de la capa anterior, tal como se muestra en la figura 6.4

[30] https://www.quora.com/What-is-the-sigmoid-function-and-what-is-its-use-in-machine-learnings-neural-networks

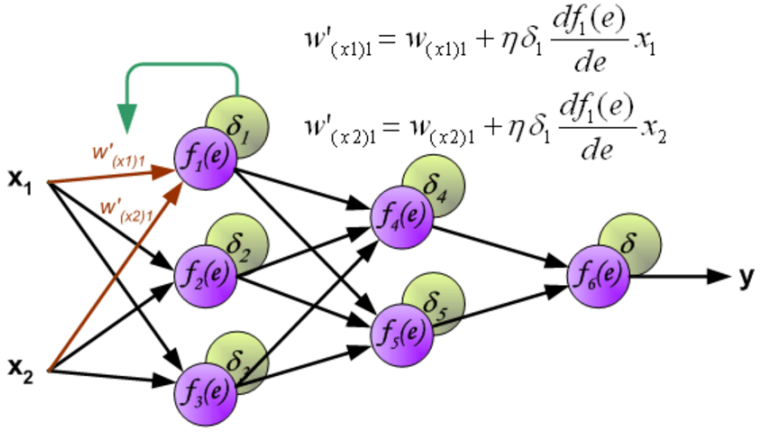


Figura 6.4: Ejemplo de Backpropagation[31]

De este modo el mismo análisis puede ser efectuado a los datos para poder obtener una estimación de N, al igual que en el análisis de combinación lineal. Utilizando los mismos parámetros estudiados en el modelo anterior se logra entrenar la red neuronal asociando la serie de $features$ a un valor de N.

Cabe destacar que los datos fueron entregados a la red neuronal en un orden al azar, sin perder la correspondencia entre una serie de valores $\overrightarrow{F}$ y el N asociado a el mismo. De esta forma se asegura que el aprendizaje no sea distorsionado por el orden cronológico de los datos.

Finalmente, y de forma análoga al modelo lineal, se obtienen las siguientes predicciones en la figura 6.5:

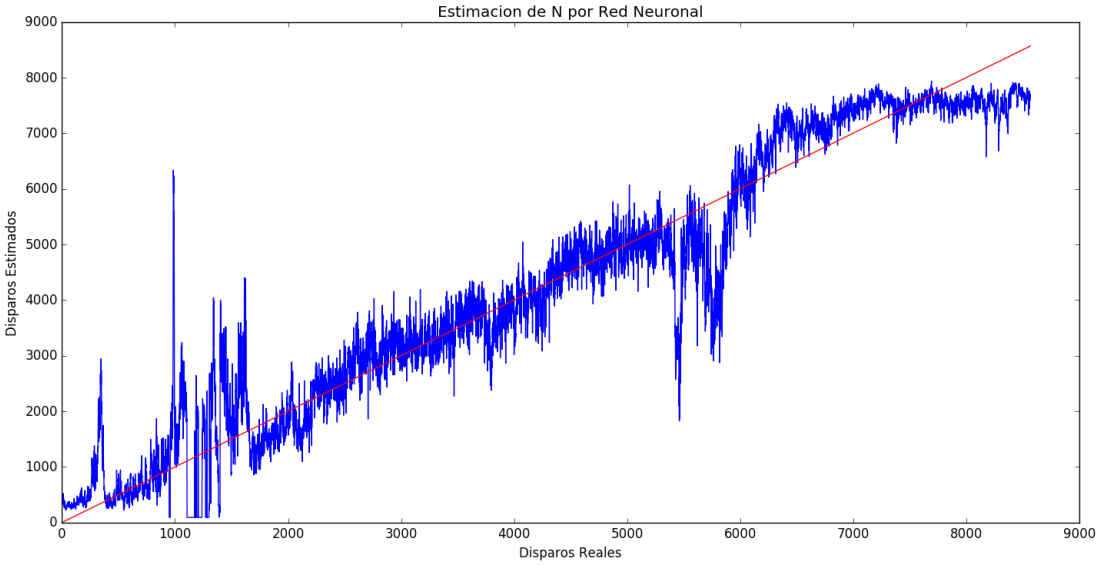


Figura 6.5: Estimaciones de N Mediante Red Neuronal[32]

[31] `http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html`
[32] Gráfico generado a partir de programa en python, siguiendo el modelo de red neuronal

7. Análisis de Disponibilidad y Predicción de Dip

A partir del análisis preliminar se pueden ver como ciertas correlaciones en los $features$ permiten establecer una estimación para el disparo actual en referencia al inicio del experimento, teniendo así una noción del progreso de la vida útil del equipo.

A pesar que esto es sumamente útil en un equipo que no ha sido monitoreado, el numero de disparos realizados resulta un parámetro fácil de obtener mediante un monitoreo mínimo del experimento. Por lo tanto, este enfoque no presenta un gran valor para un sistema de mantenimiento predictivo útil.

En este contexto vale la pena explorar otras perspectivas de los datos que pueden ser obtenidos mediante un monitoreo continuo del experimento, intentando determinar el transcurso de vida útil de forma mas precisa.

7.1. Fenómeno de Dip

Tal como se ha discutido en la sección de resultados preliminares y en particular basándose en el gráfico 5.8, se ha considerado que el mayor indicio de un buen funcionamiento del equipo es una alta $probabilidad\ de\ Dip$ en los perfiles de $\dot{I}(t)$ y $V(t)$ en un disparo arbitrario.

En los equipos PF el fenómeno de Dip significa que se ha generado un arco eléctrico que recorre el ánodo como un manto, tal como se describe en 3.2. Esta es una condición necesaria para poder generar el $Z-pinch$ y por ende es la base de la fusión nuclear en estos equipos. Realizando experimentos en equipos PF se busca estudiar la física involucrada en este fenómeno y por ende el equipo estará disponible para ser usado siempre que la probabilidad de ocurrencia de Dip sea considerable.

Para poder encontrar la probabilidad que ocurra este fenómeno en el próximo disparo, se requiere diferenciar si un dato arbitrario presenta dicha característica o no, y aplicar esto para cada dato en la sesión experimental. Posteriormente se pueden realizar gráficos como 5.8 para estimar la función de distribución de probabilidad del fenómeno y así predecir como se manifestará esta probabilidad en los disparos por venir.

El problema recae, tal como se ha mostrado en los resultados preliminares, que la categorización de existencia o no de Dip en los datos es un trabajo tedioso y largo, que en la práctica es casi imposible de implementar como una medida de monitoreo en tiempo real.

Sin embargo, si es posible de generar un algoritmo que libere al usuario de la categorización de este fenómeno, un sistema de monitoreo en tiempo real es una opción viable. Tal como se ha demostrado en la sección anterior, un sistema de redes neuronales puede predecir el suceso de un fenómeno real con una precisión mayor a los modelos lineales, esto se demuestra contrastando las figuras 6.1 y 6.5. Debido a esto es fundamental explorar la viabilidad de la predicción de Dip en una serie de datos.

Lamentablemente, este fenómeno se presenta en distintos instantes de tiempo en los perfiles característicos de un disparo, así como con distintas intensidades e incluso ocurren acoplamientos entre Dips cuando ocurren en una ventana de tiempo pequeña. Esto, junto con otras variables, como el ruido de la señal, hacen que la identificación sea subjetiva y dependiente de qué fenómeno físico es el que se busca (bias). En la figuras 7.1 y 7.2 se muestran ejemplos de distintos tipos de Dip que resultan difíciles de categorizar debido a estas ambigüedades.

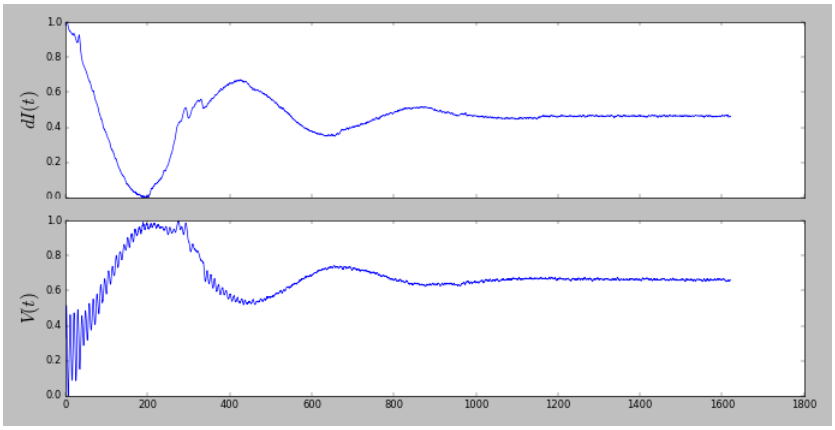


Figura 7.1: Instancias de Dip Acopladas [33]

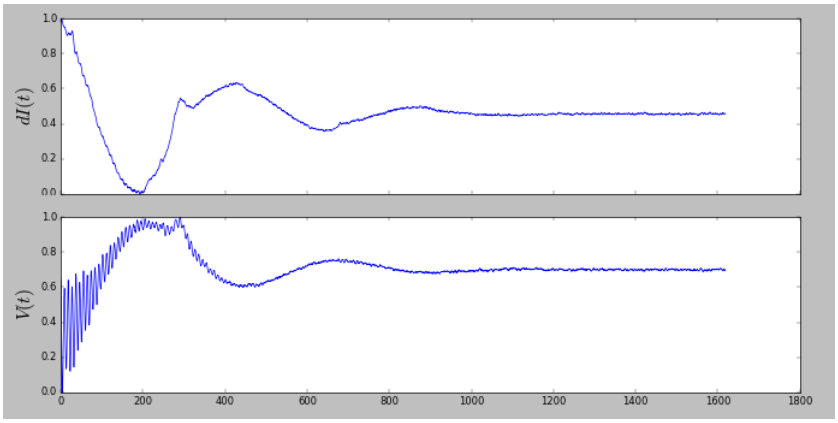


Figura 7.2: Instancia de Dip Leve [33]

En casos como estos resulta difícil la clasificación manual, debido a que el juicio humano no es completamente objetivo y esta sujeto a lo que se espera encontrar en los datos. Este tipo de datos ambiguos y percepciones humanas arbitrarias se presencian frecuentemente al clasificar y es una fuente de error que puede explicar las diferencias entre las clasificaciones manuales de las realizadas por un algoritmo de identificación.

Al tratarse de una clasificación mediante redes neuronales el principal factor que determina la eficiencia del algoritmo es la $asertividad$. La $asertividad$ es un porcentaje que determina cuantos de los elementos fueron clasificados correctamente. En este caso, el parámetro más importante es ser lo menos ambiguo posible en la selección y clasificación de datos de entrenamiento que son entregados a la NN.

7.2. Disponibilidad de PF

Suponiendo que el catastro de instancias de Dip es suficientemente objetivo, este catastro de los datos monitoreados se puede establecer el porcentaje de ocurrencia y, debido a que la presencia o no de este es una variable aleatoria, se puede extrapolar de estas frecuencias la distribución de probabilidad propia del experi-

[33] Dato observado durante categorización manual de Dip

mento.

Para estos equipos se puede definir la disponibilidad como esta distribución de probabilidad de Dip:

$$D(N) = Pr\{Disparo\ N\ Presente\ Dip\} \tag{7.1}$$

A partir de los datos preliminares y su categorización se sospecha que esta distribución sea una curva Gaussiana o Weibull. Debido a que las distribuciones de probabilidad Weibull presentan mayor versatilidad en cuanto a su forma resulta una buena primera aproximación en el análisis de disponibilidad. Por ende:

$$D(N) = \frac{k}{\lambda}\left(\frac{N-N_0}{\lambda}\right)^{k-1} \cdot e^{-\left(\frac{N-N_0}{\lambda}\right)^k} \;;\; \forall N > N_0 \tag{7.2}$$

A partir de esta curva se tiene conocimiento sobre la evolución del desempeño del equipo y por ende permite sostener un modelo de mantenimiento predictivo. Si se puede aproximar esta curva a partir de los disparos pasados en el mismo experimento se podrá conocer la cantidad de disparos restantes antes que esta probabilidad baje cierto umbral arbitrario.

Se puede, por ejemplo, estimar la cantidad de disparos restantes hasta que la disponibilidad baje del $5\,\%$ de ocurrencia. Definiendo arbitrariamente este punto como el momento de falla del equipo se pueden hacer estimaciones del numero de disparos hasta la falla del mismo, permitiendo organizar la logística del reparo con anticipación.

7.3. Pre-tratamiento de Señales

Debido a que el fenómeno de Dip ocurre en ciertas posiciones de las señales características y es evidente mediante un aumento pequeño pero agudo de amplitud, es una buena idea observar solo la sección de los perfiles en la que ocurre el fenómeno.

Se busca reducir lo mas posible la cantidad de datos $(t, f(t))$ que puedan determinar de la forma más limpia posible si el fenómeno esta tomando lugar o no en esta ventana de tiempo. Debido a que la discretización de datos dada por el osciloscopio caracteriza un disparo completo en $20{,}000$ datos discretizados y el fenómeno siempre ocurre a un tiempo específico después del máximo de la señal oscilatoria amortiguada, se puede reducir a una ventana de discretizaciones a $1{,}000$ puntos alrededor de un tiempo arbitrario. Esto permite mostrar si el fenómeno se presenta en este tiempo usando una cantidad reducida de datos. Esto reduce significativamente el tiempo de entrenamiento de la NN.

Por otro lado, y como se ve en los perfiles 5.5, toda la señal está acompañada de una cantidad de ruido no menor. Sin embargo, utilizando la librería $scipy.signal$ y tomando los primeros 30 datos de estos perfiles se puede suavizar la curva para minimizar el ruido entre datos de Dip, sin eliminar el fenómeno en sí.

Además, las curvas $\dot{I}$ y V presentan el fenómeno de Dip en sentidos contrarios en sus perfiles. Por lo tanto, es conveniente invertir una de las señales para que el Dip se muestre en la misma dirección para ambos perfiles. Es posible hacer esto dejando ambos perfiles con valores positivos.

Una vez hechos todos estos procesos de reducción de ruido y simetría, es conveniente que esta ventana de datos de los perfiles en el tramo de interés sea normalizada de tal forma que el micro-perfil de dip se encuentre siempre en el rango $(0, 1)$. La utilidad de esto recae en el campo de la estadística, ya que mientras ambas curvas tienen formas arbitrarias, en caso de existir un evento de Dip, a las oscilaciones amortiguadas se les suma una alza similar a una curva gaussiana o normal.

Muchas veces las curvas de distribución normal son atribuidas a eventos de este tipo, y debido a que se poseen dos curvas gaussianas se considera la multiplicación de estas para una acentuación del efecto Dip en caso que haya. Esto recae en el efecto de la multiplicación de dos curvas normales en el rango $(0, 1)$, como se ilustra en la figura 7.3.

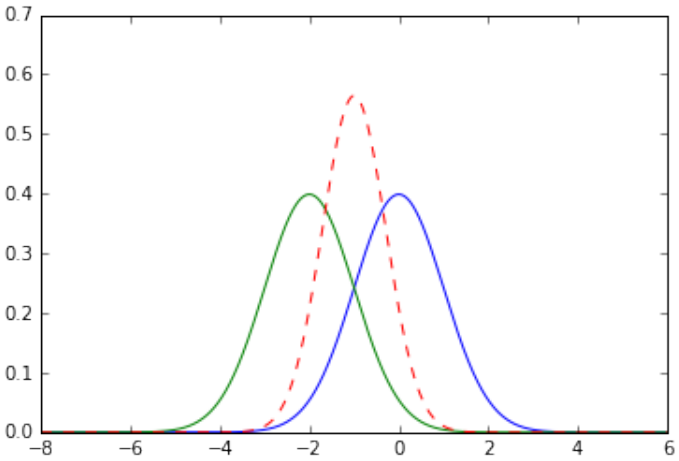


Figura 7.3: Multiplicación de Densidades de Probabilidad Normales[34]

Donde la curva resultante es una distribución gaussiana de media y varianza:

$$\mu = \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \tag{7.3}$$

$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \tag{7.4}$$

De esto se desprende que la nueva curva gaussiana posee una media entre las dos curvas (que es irrelevante, debido a que el Dip se presenta en la misma posición horizontal) y con una varianza $menor$ (debido a que la suma de dos valores en $(0, 1)$ es mayor a la multiplicación de estos).

A pesar que el fenómeno de Dip no es exactamente una curva gaussiana, es similar en forma y conlleva a las mismas consecuencias de multiplicación entre estas curvas, ya que las funciones multiplicadas se encuentran en el rango $(0, 1)$ y el Dip se presenta exactamente en el mismo instante de tiempo en ambos perfiles.

De esta forma se puede computar una curva de la zona de interés con una intensificación del fenómeno de Dip cuando se presenta. Esto se puede ver en las figuras 7.4 y 7.5:

[34] https://stats.stackexchange.com/questions/221195/pointwise-multiplication-of-two-gaussians-and-normalization-in-py

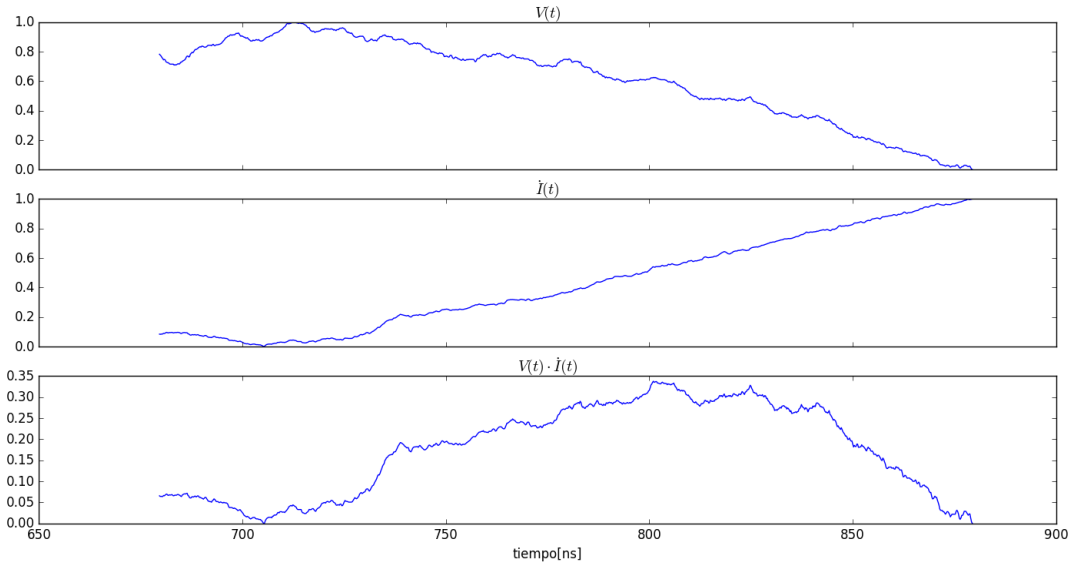


Figura 7.4: Señales Caracterisiticas Pre-tratadas sin Insancia de Dip[35]

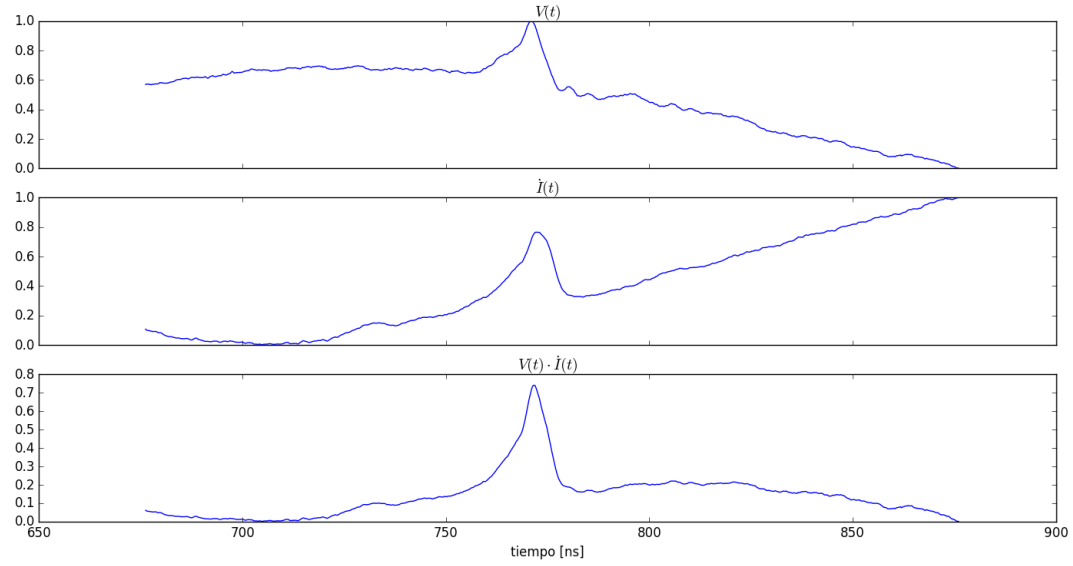


Figura 7.5: Señales Caracterisiticas Pre-tratadas con Instancia de Dip[35]

Cabe destacar que los gráficos de las curvas multiplicadas en esta ventana de tiempo se encuentran en amplitudes menores a $0{,}4$ en caso de no presenciar Dip y alcanzan amplitudes de $0{,}7$ o más en caso de ocurrir. Este cambio no se presenta en las curvas previas a la multiplicación, comprobando que el procedimiento de acentuación del fenómeno funciona.

[35] Gráficos generados en python mediante multiplicación de zonas de interés normalizadas en ambos perfiles característicos

Por otro lado, las instancias de Dip corresponden a un pico localizado en una zona específica de las curvas V e $\dot{I}$. Este instante de tiempo se asume constante a lo largo de un experimento, sin embargo este se puede desplazar de acuerdo a las condiciones experimentales, presentándose en distintos lugares al variar la presión de la descarga, la geometría de los electrodos, tipo de gas, entre otros.

En base a esto es conveniente poder separar las instancias de Dip de la curva sinusoidal amortiguada, ya que esto permitiría analizar nuevos datos que presentan Dip en otros instantes de tiempo sin tener que entrenar una red neuronal específica para cada una de estas instancias.

Para poder aislar el Dip de la curva sinusoidal de la señal se pueden tomar las curvas las curvas V e $\dot{I}$, para luego encontrar los parámetros que rigen este movimiento de tal modo que corresponda a la curva:

$$S(t) = A \cdot e^{-\gamma t} \cdot \cos(\omega t) \tag{7.5}$$

Tomando en cuenta que el disparo en sí comienza cuando la sinusoidal amortiguada se encuentra en su amplitud máxima, no es necesario mantener los datos anteriores a este tiempo y puede ser considerado como cero, por ende la función $S(t)$ tendría fase cero. Para generar esta ecuación se crea una función en python con los mismos parámetros, solo que normalizados, esto se debe a que la función de optimización es sensible a escala y puntos iniciales. Así se encuentran los parámetros óptimos que minimicen el error entre la curva S normalizada y los datos normalizados. Posteriormente estas curvas S_V y $S_{\dot{I}}$ son escaladas a la magnitud de la señal y restadas de ésta para así poder obtener el perfil en el que solo se encuentran otros fenómenos, como el Dip. En la figura 7.6 se muestra la resta entre señales mediante la curva S con la observación de ventana de tiempo en las vecindades de $t = 300[ns]$:

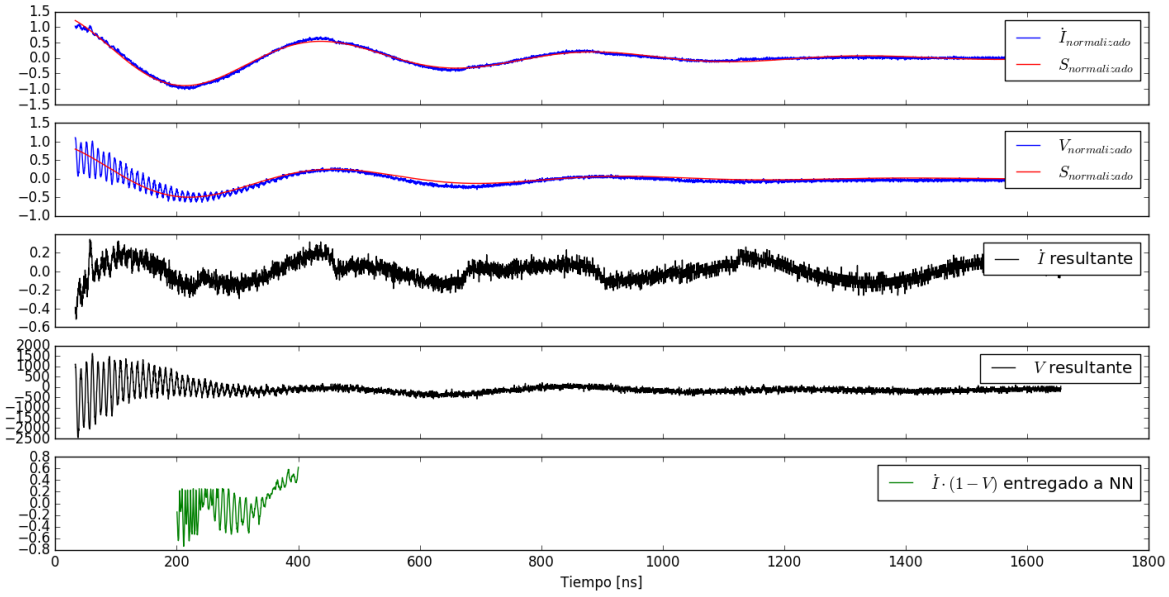


Figura 7.6: Resta de Señales con Curva $S(t)$ [36]

[36] Gráficos generados en python mediante resta de función optimizada usando $curve - fit$

A pesar de reducir significativamente el movimiento sinusoidal este método es costoso en tiempo de computación. Tratandose de hacer una lectura de los datos para los miles de disparos, el tiempo en que esta lectura opera debe ser mínimo. Por otro lado, está la posibilidad de que un dato peculiar no permita el algoritmo a converger, encontrando parámetros de la curva S que no sean correctos o bien no encontrando ninguno, distorsionando los datos.

Sin embargo, otra manera de lograr un resultado similar es pasar las señales V e $\dot{I}$ por un filtro más severo que elimine todos los movimientos por fuera de la sinusoidal amortiguada. De esta forma nos aseguramos tener siempre una curva que sigue la forma de la señal y en un tiempo de cómputo menor. Esto se puede apreciar en la figura 7.7:

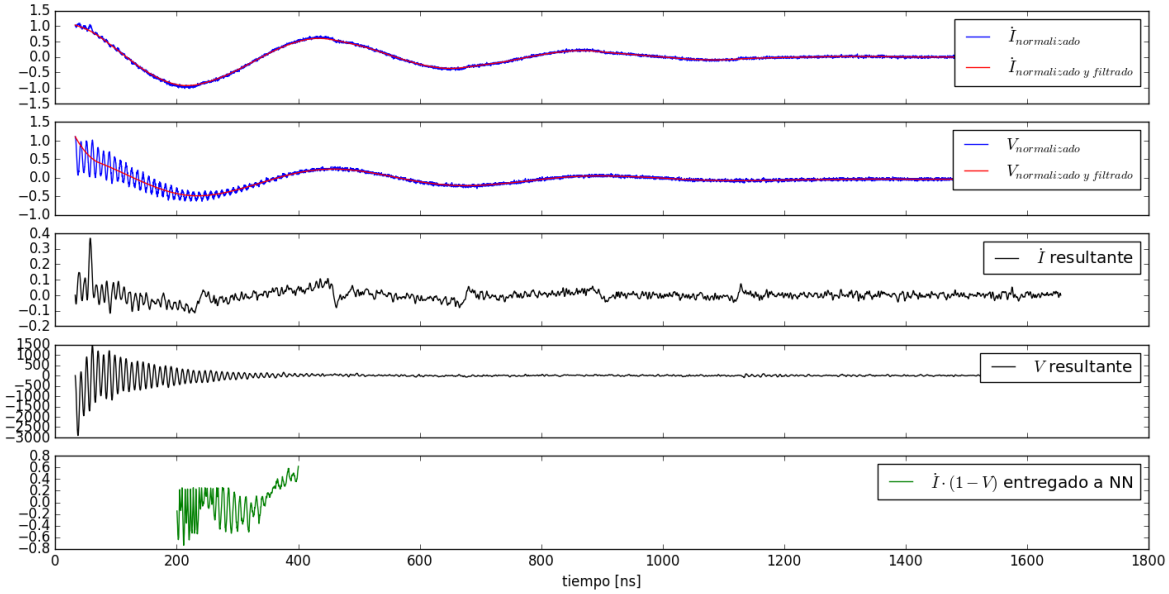


Figura 7.7: Resta de Señales con Curva Filtrada [37]

7.4. Clasificación Usando Neural Networks

Esta vez se utilizará un modelo de red neuronal similar al utilizado para la regresión del numero de disparos de la sección anterior, salvo que esta vez será usada para clasificar la ocurrencia de un fenómeno y debido al formato de los datos, las dimensiones de la red serán distintas.

Para esta NN la primera capa es la función de multiplicación de señales características normalizadas discutidas anteriormente, que para este tipo de Dip se considera una ventana de tiempo de aproximadamente 1000 puntos discretos. Sin embargo esta cantidad de puntos en insecesaria y basta con recrear la misma curva con $n \sim 100$ puntos.

[37] Gráficos generados en python mediante resta de función filtrada usando $signal.filtfilt$

Esto se hace seleccionando un dato de cada diez para representar el perfil. De esta manera se puede reducir la cantidad de datos, haciendo que el proceso iterativo de la red neuronal sea mas rápido sin alterar la forma del perfil. Además de esto, la red se ha optimizado agregando un nodo auxiliar por capa de tipo $Bias$. Esto se refiere a un nodo que siempre posee un valor de $+1$, con el fin de regularizar la variación del error. Muchas veces ocurre que, cuando el error es cercano a cero, el sistema de $backpropagation$ se descompensa y termina quedándose atascado en un mínimo local, esta neurona previene este fenómeno. Con todo esto considerado, la red neuronal utilizada para esta clasificación se puede ver en la figura 7.8:

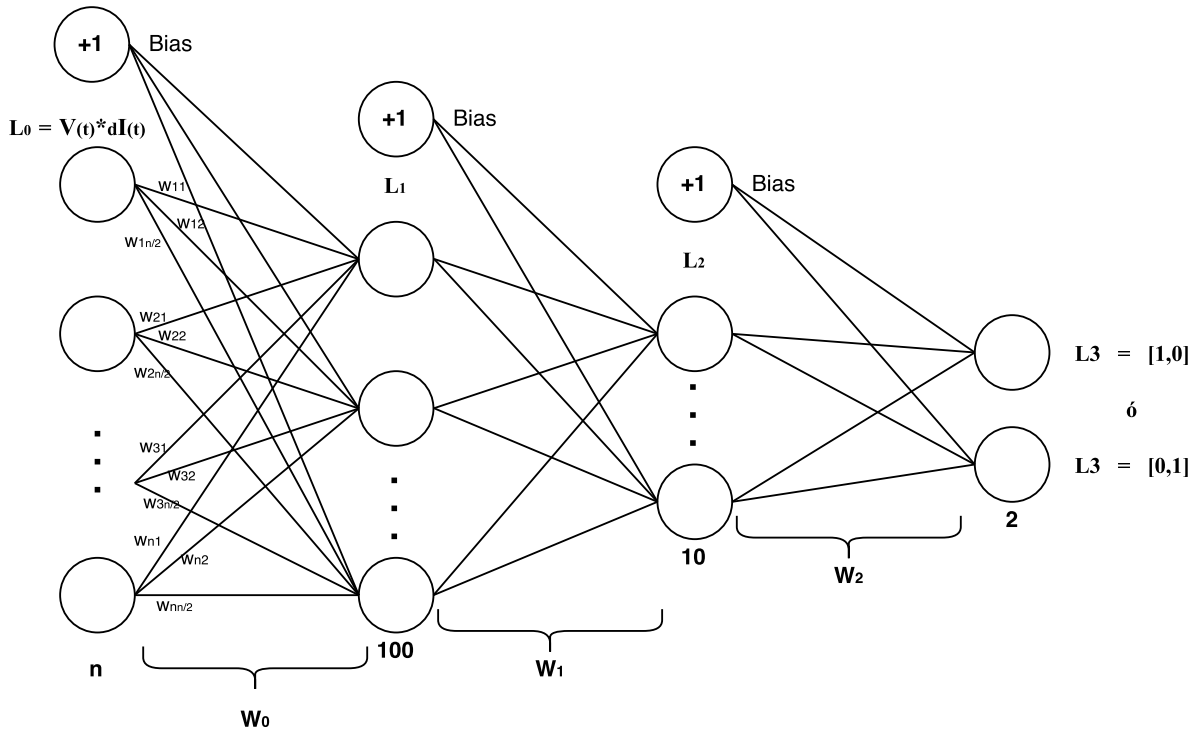


Figura 7.8: Red Neuronal para Clasificación de Dip[38]

Para el entrenamiento de esta red neuronal se considera que la capa final se puede encontrar en dos posibles estados, $[1, 0]$ o $[0, 1]$. Donde el primer valor corresponde a una neurona que busca clasificar como 1 a los perfiles que presenten Dip y 0 los que no, mientras que la segunda neurona hace el proceso inverso. De esta manera se obtienen dos indicadores para clasificar un dato, que se encuentran entre 0 y 1. De este modo, también se puede estimar la seguridad con la que la red neuronal predice un estado u otro.

Esta red neuronal entrenada puede ser utilizada para predecir si un dato nuevo es considerado como Dip o no mediante la multiplicación del dato que describe el tramo donde ocurre el Dip por las tres matrices de la red (W_0, W_1 y W_2). Estas matrices definen la red y están determinadas según la cantidad de datos de entrenamiento, la cantidad de iteraciones, el parámetro α (determina la velocidad de acercamiento en el proceso de backpropagation) y por sobre todo el tipo de datos con los que fue entrenada (forma, escala, dimensiones, ruido, entre otros factores).

Visualizando los datos del análisis preliminar se puede notar que el fenómeno de Dip se produce en los alrededores de $t = 750ns$, por lo que se puede tomar una ventana de tiempo alrededor de este valor como

[38] Diagrama de red neuronal de clasificación generado en `www.draw.io`

dato fundamental de entrenamiento y posterior predicción. Utilizando 1500 datos tomados al azar de todos los preliminares para entrenar la red neuronal a lo largo de 2000 iteraciones y con un parámetro $\alpha = 1$ se logran obtener los valores de la matrices características de la red. Con estas se pueden predecir las instancias de Dip a lo largo de todo el set de datos, incluyendo los aproximadamente 7000 datos que la red no ha visto anteriormente. Las predicciones obtenidas mediante este procedimiento se pueden ver en la figura 7.9. En esta figura se han contado la cantidad de instancias de dip cada 100 disparos para así obtener una estimación para la probabilidad de obtención de Dip a lo largo del experimento. Se encuentra que la red neuronal clasifica con un $93{,}25\,\%$ de asertividad los datos preliminares.

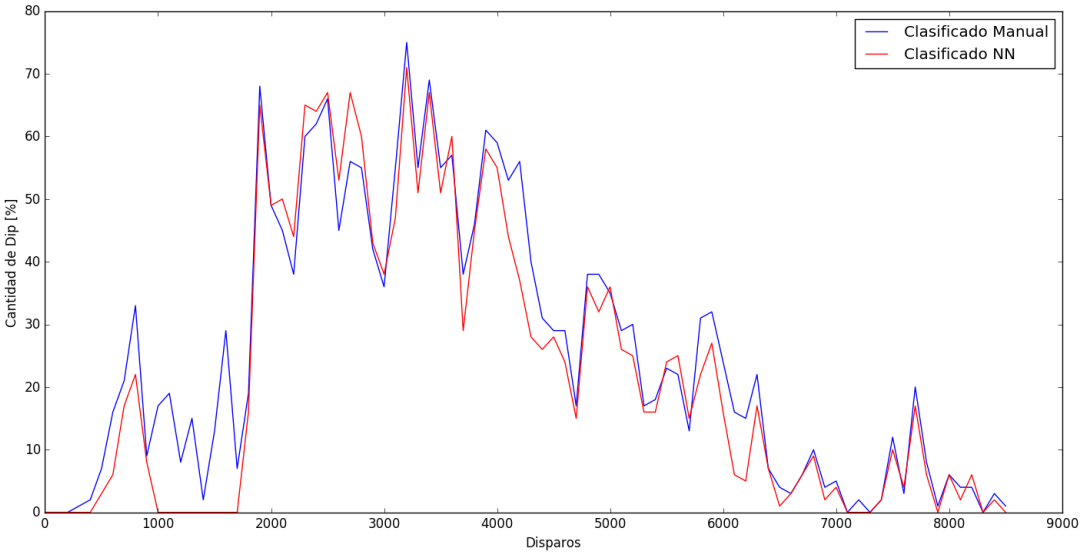


Figura 7.9: Comparación de Identificación de Dip en Datos Finales[39]

El perfil de distribución probabilidades de ocurrencia de Dip describe la disponibilidad del equipo PF, por lo tanto es central en el análisis poder aproximar este perfil y predecir con la mayor exactitud posible cuando esta distribución se encontrará bajo un umbral de falla. De este modo se pueden estimar los disparos restantes hasta la falla del equipo, en tiempo real.

[39] Gráfico generado a partir de datos preliminares

8. Experimentación Final

8.1. Montaje

Una vez acabada la vida útil del equipo PF-2J durante la experimentación preliminar, fue requerido reparar este mismo para poder generar nuevos datos experimentales. Sin embargo, debido principalmente a problemas de manufactura, se realizaron múltiples disparos sin poder comenzar la vida útil nuevamente. Es decir, los disparos fueron hechos sin presentar Dip.

Debido a esto se ha optado por un rediseño del ánodo central, que anteriormente era compuesto de un tubo de cobre que descansa sobre una estructura metálica que permite el paso de la descarga desde el spark-gap hacia el ánodo. Esto fue reemplazado por un ánodo de SAE316 fijado en su descanso por medio de una rosca. En la misma linea, se realizaron planos más específicos de las dimensiones requeridas por el aislante de vidrio.

Estos cambios en el equipo dieron fruto puesto que en la siguiente ronda de disparos si se presentó Dip. Sin embargo, debido a los cambios y a la sensibilidad de la física que rodea la descarga, esta vez el fenómeno de Dip se presenta en un instante de tiempo cercano a los $t = 300[ns]$ y sólo estando sometido a presiones bajas, específicamente, menores a $5[mbar]$.

Teniendo esto en cuenta se procede a realizar una nueva toma de datos, el montaje experimental se puede apreciar en la figura 8.1.

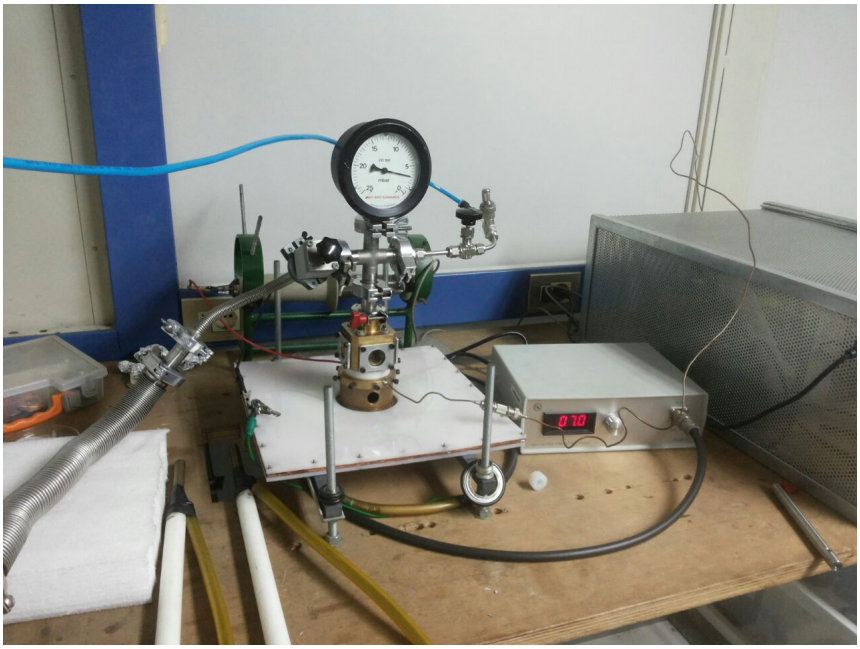


Figura 8.1: Montaje Experimental Final [40]

Los parámetros experimentales de este montaje son los siguientes:

- La presión de trabajo es de $3[mbar]$

[40] Imagen de montaje PF-2J para la toma de datos finales

- El gas utlizado es H_2
- El voltaje de carga del condensador es de $7kV$
- Se utiliza osciloscopio $TDS3064B$ para toma de datos y mediante una tarjeta de adquisición estos son almacenados en un computador

Bajo estos parámetros se realizan miles de disparos con el equipo a lo largo de su vida útil. Sin embargo, esta vez el equipo supera con creces la vida útil de 8500 disparos que presentó en la experimentación preliminar. Por este motivo y los problemas explicados anteriormente el experimento concluyó antes de alcanzar el fin de la vida útil.

Sin embargo, los datos recopilados permiten validar el análisis estipulado anteriormente sin problemas, cumpliendo así su rol de experimentación final de todas maneras.

8.2. Resultados Finales

A continuación se presentan los resultados obtenidos a partir del análisis realizado sobre la disponibilidad del equipo PF-2J utilizando como base los datos obtenidos en la experimentación final.

8.2.1. Identificación de Dip Mediante NN

Utilizando como base los datos preliminares se realiza el entrenamiento de la red neuronal tomando como referencia la clasificación de los datos preliminares. En este caso la red fue entrenada utilizando 2000 de los 8500 datos, realizando 6000 iteraciones de aprendizaje sobre ellas con un factor de velocidad de convergencia $\alpha = 0{,}3$ para asegurar que el progreso sea gradual y evitar la divergencia o estancamiento. En el gráfico 8.2 se puede apreciar como varia el error de la capa final de la red a lo largo de las iteraciones.

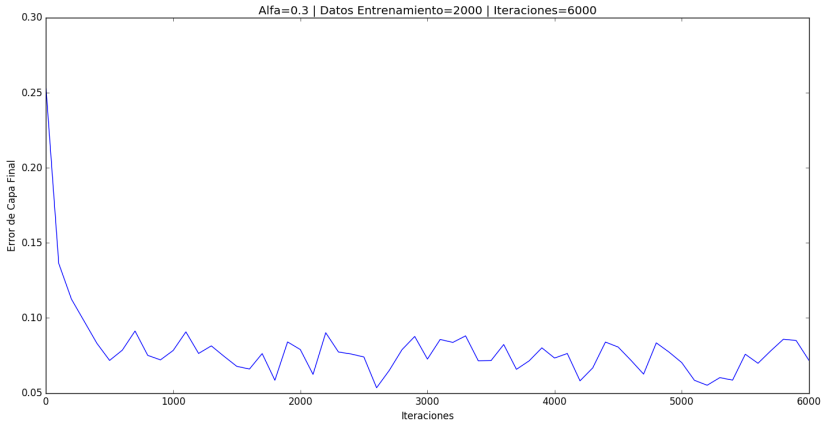


Figura 8.2: Error de Última capa de NN [41]

[41] Red entrenada a partir de datos preliminares pre-tratados y clasificados manualmente

Tras realizar aproximadamente $N = 8{,}000$ disparos de esta ronda experimental se procede a utilizar estos datos para realizar predicciones de ocurrencia de Dip en base a la red neuronal entrenada con los datos preliminares ya clasificados. Es decir, se utiliza una red neuronal entrenada con datos experimentales previos, por lo que en el entrenamiento de esta red neuronal no fue incluido ninguno de los datos evaluados por ella.

Posteriormente, estos disparos fueron catalogados manualmente del mismo modo que los primeros, utilizando el programa disponible en el Anexo B. Esto permite comparar la identificación de Dip de la red neuronal con la clasificación manual de cada disparo en el tramo $(0, 8500)$, tal como se puede apreciar en la figura 8.3:

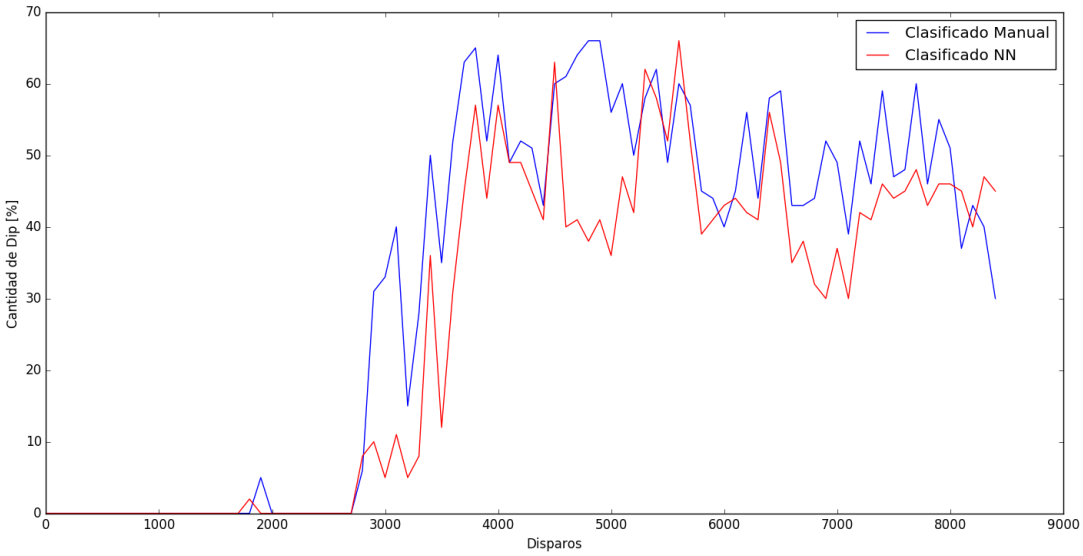


Figura 8.3: Comparación de Identificación de Dip en Datos Finales [42]

En este caso de predicción de instancias de Dip de datos nuevos la red neuronal consigue un $83{,}3\,\%$ de asertividad.

8.2.2. Pronósticos de Disponibilidad

Considerando que estas predicciones son suficientemente asertivas, se realizan predicciones de la curva de Disponibilidad del equipo mediante la extrapolación de una curva Weibull. De forma similar a la extrapolación generada en 7.6, los porcentajes de Dip identificados fueron escalados para poder utilizar la función $curve_fit()$ en $python$. Se incluye el momento de falla arbitrario, cuando la Disponibilidad alcanza el $5\,\%$ de probabilidad de Dip.

Claramente la curva será mas confiable a medida que transcurra el experimento, debido a que tendrá mas datos para realizar la extrapolación. Por esta razón se presentan extrapolaciones de una porción de los datos clasificados, emulando una implementación de monitoreo en tiempo real. En las figuras 8.4 y 8.5 se realizó este análisis para lo datos preliminares. Esto con la finalidad de contrastar con las figuras 8.6 y 8.7, en donde

[42] Gráfico generado a partir de datos finales y red neuronal entrenada con datos preliminares

se realiza para los datos experimentales finales.

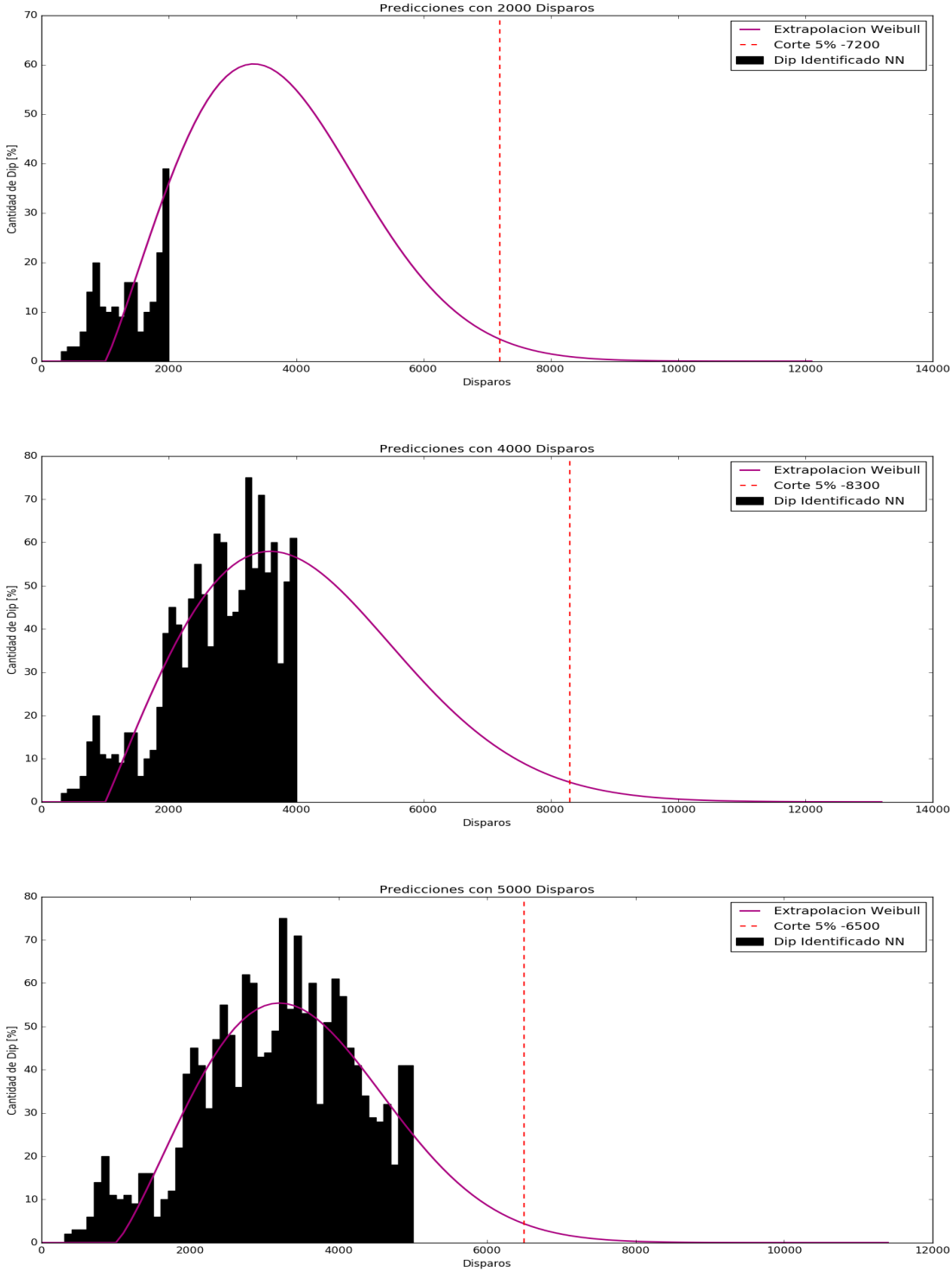
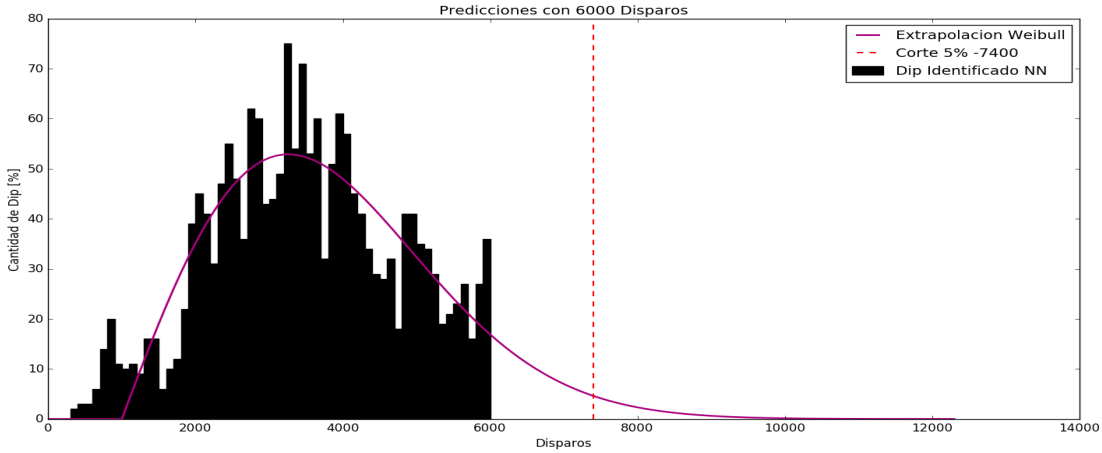
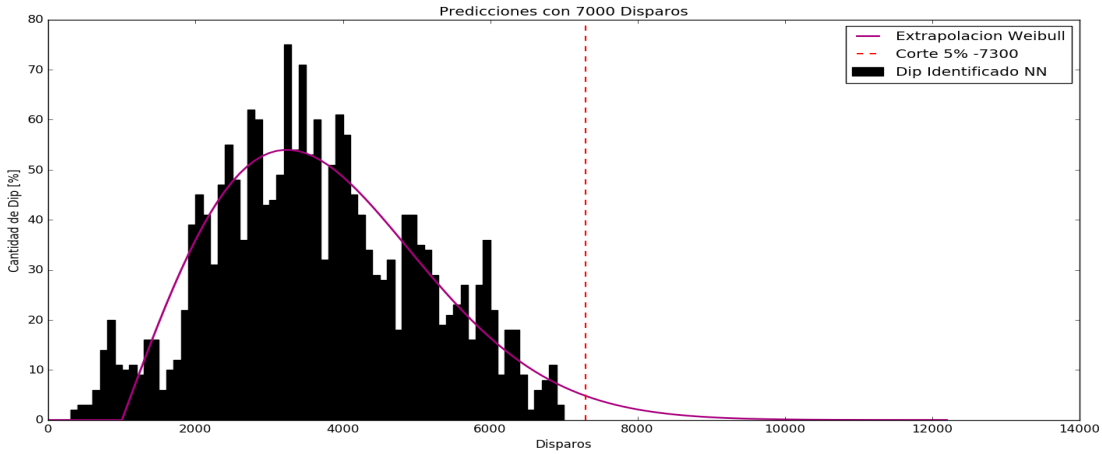


Figura 8.4: Pronosticos de Disponibilidad de Datos Preliminares (1)[43]





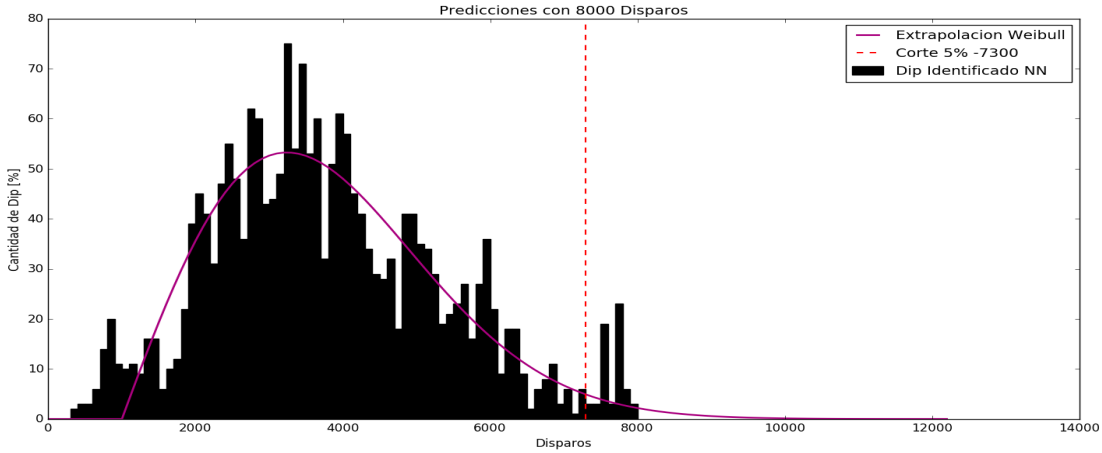
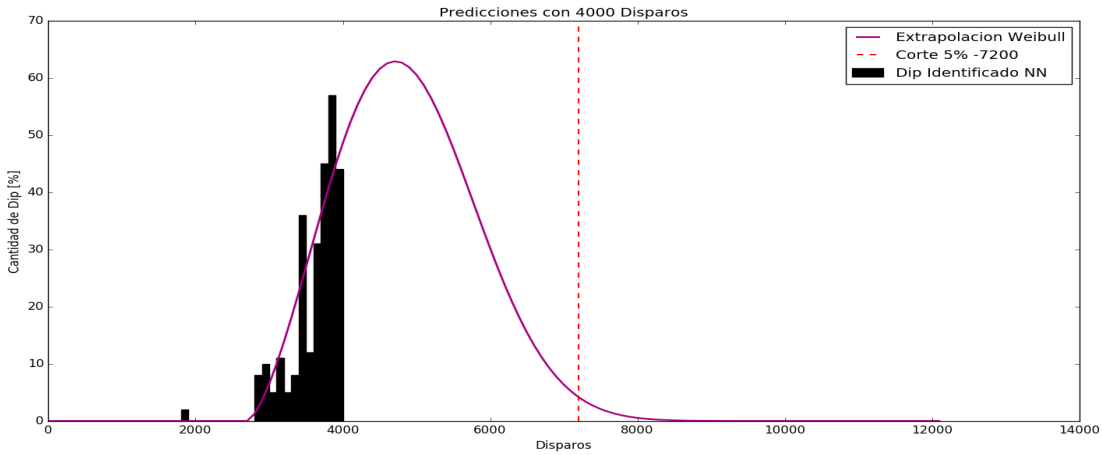
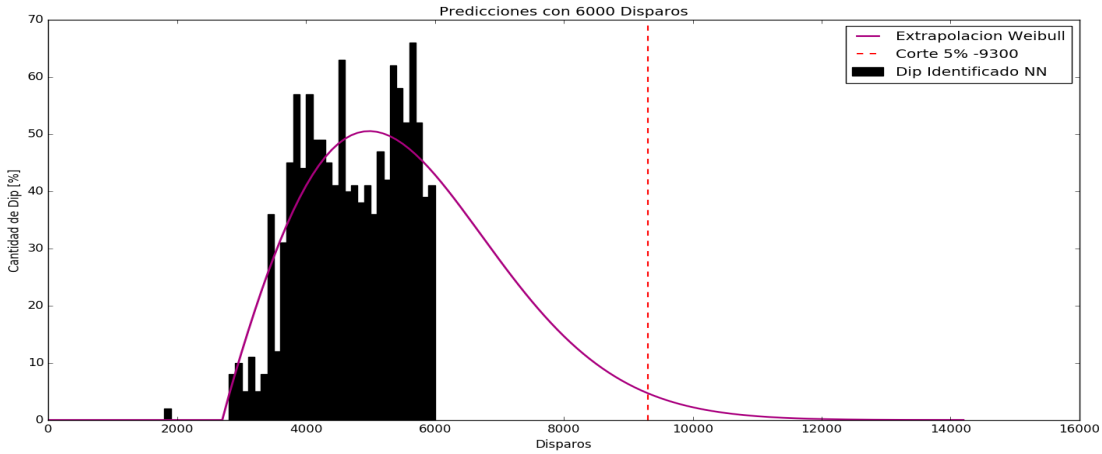


Figura 8.5: Pronosticos de Disponibilidad de Datos Preliminares (2)[43]

[43]Pronósticos en base a identificación de datos preliminares mediante NN entrenada con una porción estos mismos





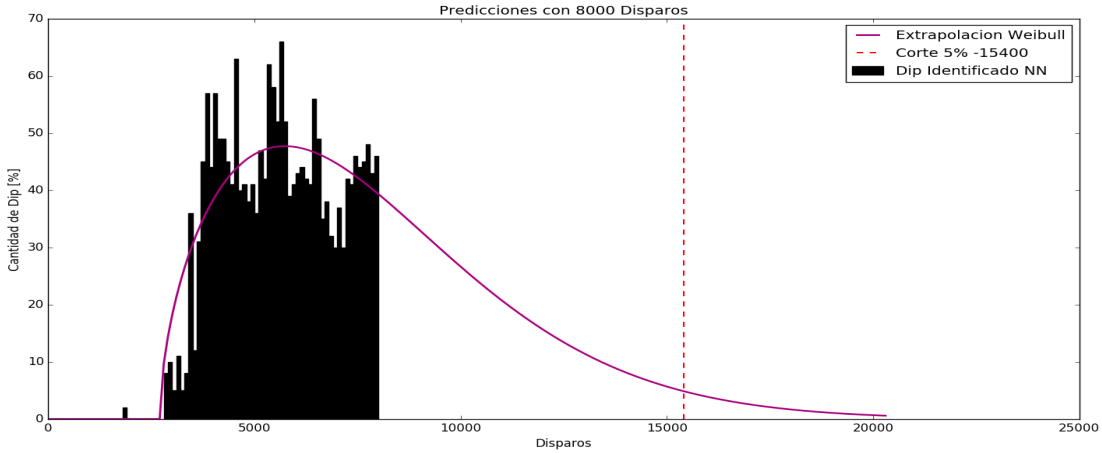
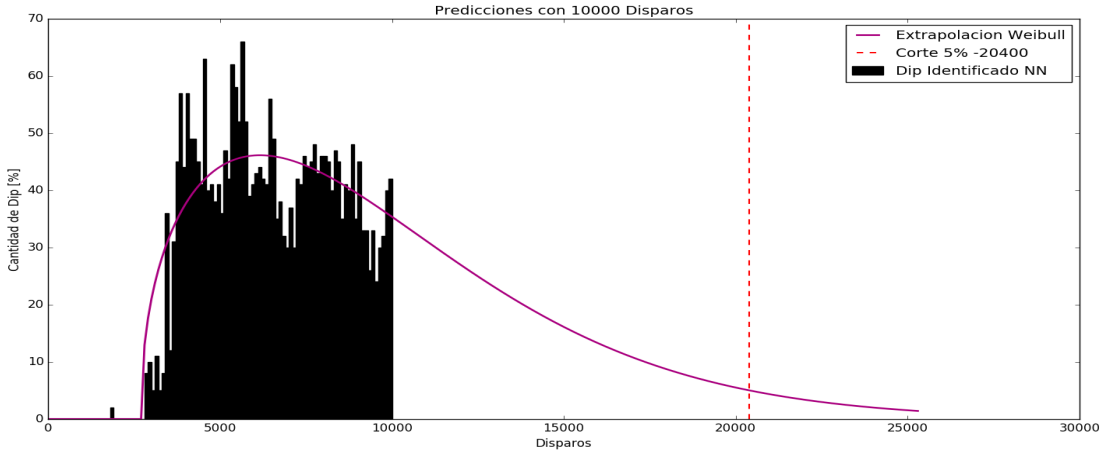
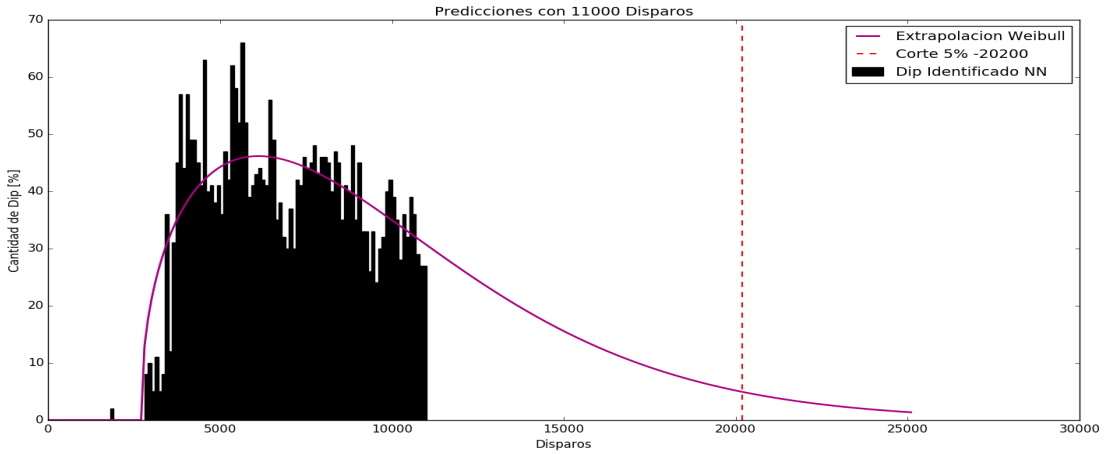


Figura 8.6: Pronosticos de Disponibilidad de Datos Finales (1)[44]

[44]Pronósticos en base a identificación de datos finales mediante NN entrenada con datos preliminares





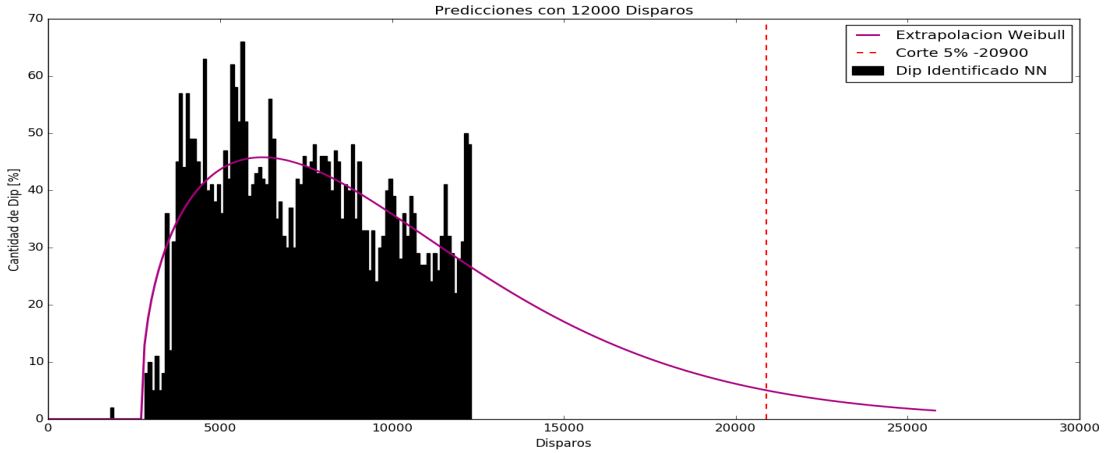


Figura 8.7: Pronosticos de Disponibilidad de Datos Finales (2)[44]

En estos pronósticos de la función de Disponibilidad en las figuras 8.4 a 8.7 se obtienen predicciones sobre el momento de corte con $5\,\%$, que pueden considerarse como predicciones de tiempos de falla. En las figuras

8.8 y 8.9 se pueden ver como evolucionan las predicciones de este momento a medida que se clasifican más datos experimentales preliminares y finales, respectivamente.

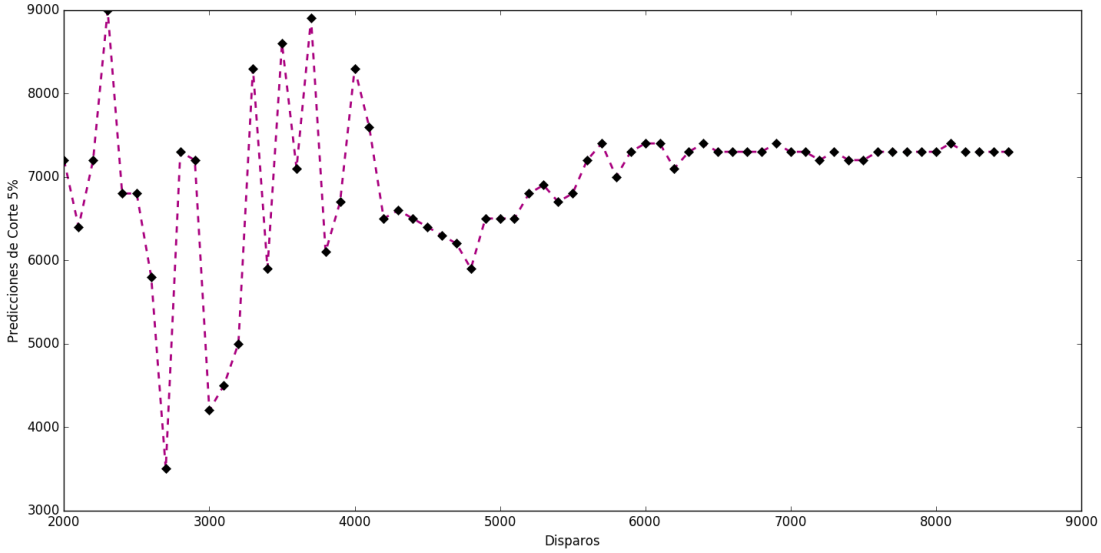


Figura 8.8: Predicciones de Momento de Falla de Datos Preliminares [45]

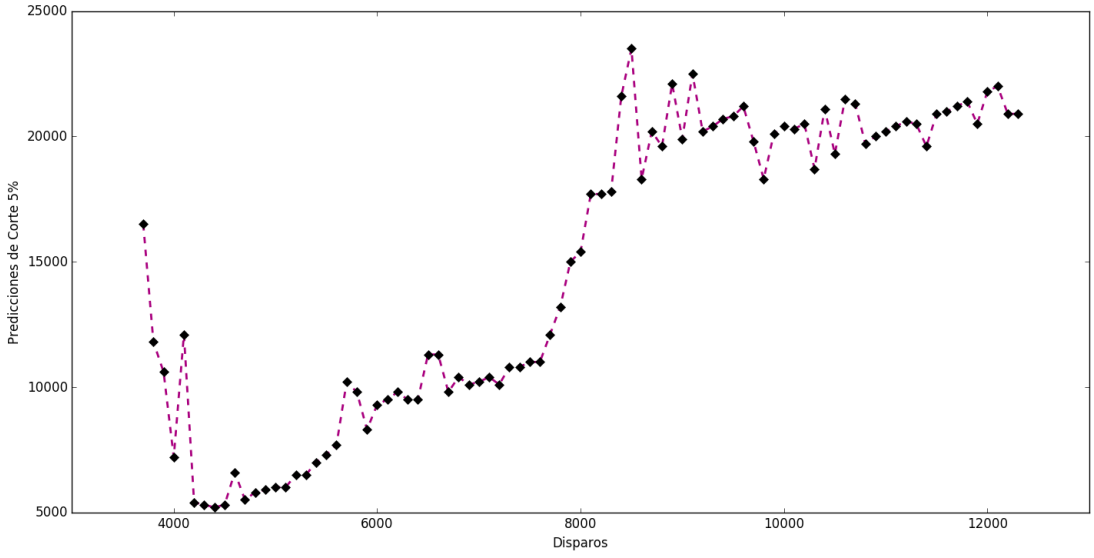


Figura 8.9: Predicciones de Momento de Falla de Datos Finales [46]

[45] Predicciones de falla en base a identificación de datos preliminares mediante NN entrenada con una porción de estos
[46] Predicciones de falla en base a identificación de datos finales mediante NN entrenada con datos preliminares

9. Análisis Final

9.1. Análisis de Resultados

De los resultados finales obtenidos a partir del análisis de disponibilidad de los datos experimentales finales, contrastado con el mismo análisis de los datos preliminares se pueden hacer varias observaciones.

En primer lugar, en base a las figuras 7.9 y 8.3, ambas realizadas con el entrenamiento de una red neuronal basada en $2{,}000$ datos preliminares, se puede notar que el pre-tratamiento de las señales efectivamente permite aislar el fenómeno de Dip de la curva sinusoidal. Del mismo modo se puede ver que basta con entrenar una red neuronal para analizar la ocurrencia de Dip en un tiempo arbitrario, debido a que la curva no se encuentra deformada por la sinusoidal amortiguada característica.

En la misma linea, se puede ver que se obtuvieron asertividades de identificación del fenómeno del $93{,}25\,\%$ para los mismos datos de entrenamiento y de $83{,}3\,\%$ para datos nuevos. Dando una buena base para asegurar que las predicciones realizadas tienen correlación directa con la realidad.

En segundo lugar, los gráficos 8.4 a 8.7 muestran como una distribución de probabilidad de Weibull se ajusta de buena manera a ambos experimentos. Teniendo en cuenta que se ajusta de buena manera para parámetros de forma y escala muy diferentes en ambos experimentos. Esto confirma en parte la aseveración que la curva de Disponibilidad de los equipos PF sigue esta distribución, a pesar de requerirse de más experimentos para identificar la naturaleza de esta curva de forma mas detallada.

En tercer lugar, se puede ver que las clasificaciones de NN permiten extrapolar estos perfiles probabilísticos para este tipo de experimentos. Fijando un limite de probabilidad arbitrario, las predicciones de falla observadas en los gráficos 8.8 y 8.9 muestran como evoluciona la predicción del momento de falla a medida que se cuenta con mas datos a analizar. Se puede ver como al principio estos pronósticos de falla presentan una variabilidad alta, sin embargo tanto para los datos preliminares como los datos finales, estas predicciones convergen a un valor específico. Notando la consistencia de esta predicción para los datos preliminares con el momento de falla real en este experimento se demuestra que el algoritmo permite prever con anticipación el momento de falla. Esto se puede extender a los datos finales, que a pesar de ser un experimento inconcluso, presenta una convergencia de los pronósticos de falla a alrededor de los $20{,}000$ disparos.

Por último, se puede ver que el proceso de categorización de datos se puede realizar en tiempo real a lo largo de un experimento. Realizando así aproximaciones de la Disponibilidad del equipo, basándose netamente en categorizaciones previas de instancias de Dip y el instante $t_{Dip}[ns]$ en el disparo que se presenta el fenómeno en el experimento de estudio.

9.2. Propuestas de Mantenimiento

En base a lo discutido sobre la información que se puede obtener a partir de clasificaciones de Dip previas y un pre-tratamiento de señales adecuado aplicado a nuevos datos experimentales, se puede establecer un sistema de mantenimiento predictivo mediante el monitoreo de las curvas V e $\dot{I}$ en los disparos de equipos PF.

Considerando los resultados obtenidos se pueden ver los alcances y ventajas del modelo de predicciones en función de la variabilidad de las predicciones de falla presentes en los gráficos 8.8 y 8.9. Esta información junto con la representación de la curva de Disponibilidad pueden ser generados en tiempo real, dando cuenta

del pronóstico del equipo y la precisión de este pronóstico.

En base a esto se realizan dos propuestas de mantenimiento predictivo, apreciables en las figuras 9.1 y 9.2. Ambos consisten en el uso de los distintos programas en $python$ realizados en el transcurso de la memoria.

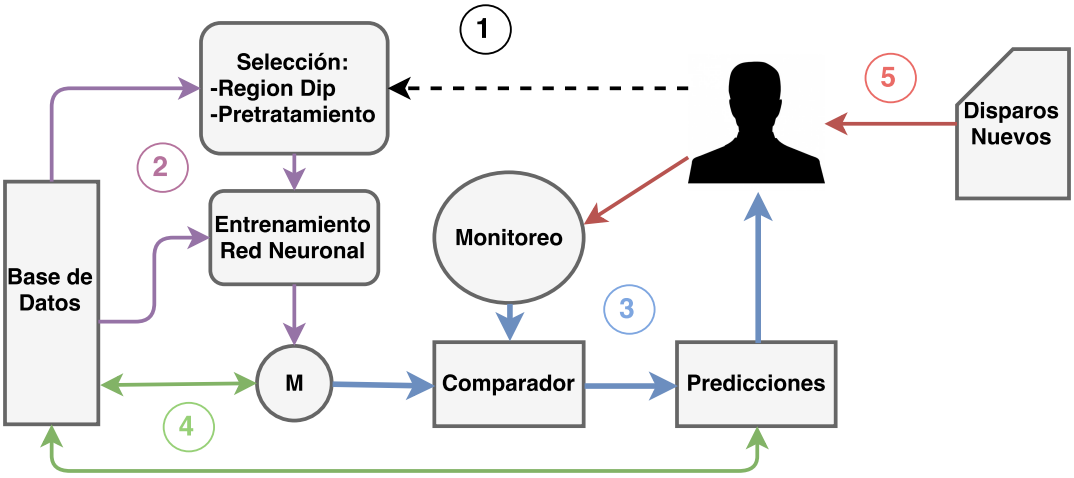


Figura 9.1: Sistema de Mantenimiento Predictivo Asistido [47]

En el primer sistema se contempla la interacción del usuario en 1, donde se debe elegir el instante temporal t_{Dip} de las señales en la que se desea analizar el fenómeno de Dip. Esto depende de cada experimento y modelo de PF, por lo que es necesario el suministro de esta información por el usuario. En este instante se pueden seleccionar otros métodos de pre-tratamiento de las señales V e $\dot{I}$, capaces de identificar y acentuar los patrones que permiten identificar el Dip.

Considerando estos parámetros seleccionados, se requiere procesar los datos para obtener los datos de entrenamiento y la generación de las matrices M que definen una NN entrenada en 2. Posteriormente, el usuario debe suministrar manualmente los nuevos datos experimentales de 5 a 3, para realizar la clasificación de estos y la posterior extrapolación de Disponibilidad en las predicciones. En 4 se contempla el almacenamiento de distintas redes neuronales entrenadas, así como distintas predicciones realizadas, de tal forma de no requerir repetir innecesariamente los pasos 2 y/o 3. Esto además permite contrastar diferentes NN y pre-tratamientos con sus correspondientes predicciones con facilidad.

En los Anexos C a I se pueden ver en mayor detalle todos los programas desarrollados en python usados para implementar este sistema de mantenimiento predictivo.

[47] Diagrama generado en `draw.io`

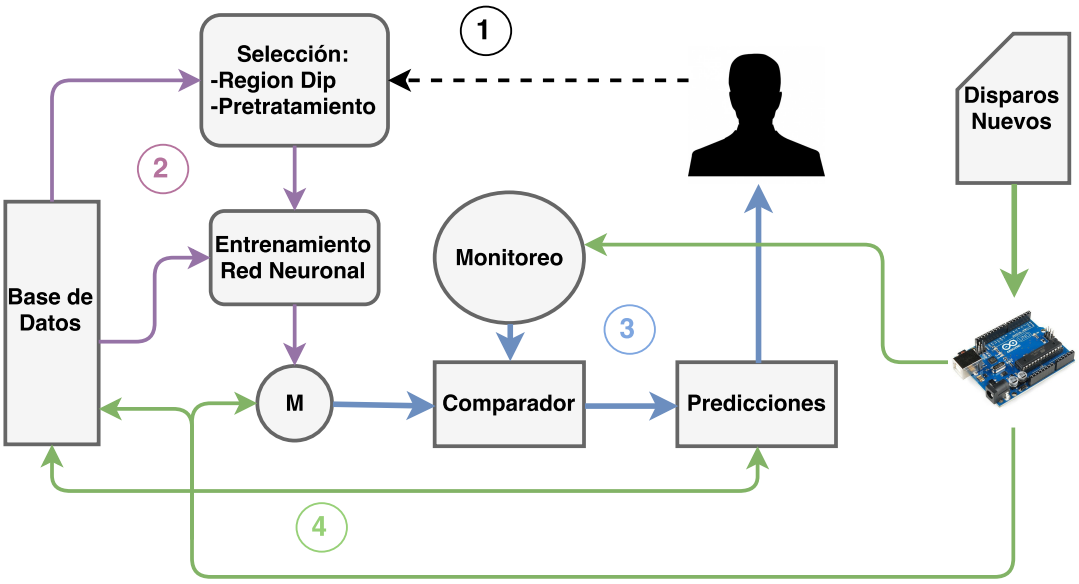


Figura 9.2: Sistema de Mantenimiento Predictivo Automático [48]

En el sistema ilustrado en 9.2 se contemplan los mismos procesos, a diferencia de 4 y 5. El usuario puede ser librado del suministro manual de los datos al sistema, que puede ser realizado de forma automática mediante una interfaz entre el osciloscopio y el usuario. Esto potencialmente libraría al usuario no solo de suministrar los datos de forma manual para obtener las predicciones, sino que también le permitiría almacenarlos de forma automática para su posterior análisis científico, que actualmente es la finalidad de estos equipos.

9.3. Discusión

Respecto al análisis de correlación de datos se puede ver en las figuras 5.10 a 5.13 como evolucionan los parámetros característicos de las curvas V e $\dot{I}$. Muchos de estos no presentan correlaciones directas en la evolución del experimento, a pesar de demonstrar una correlación en conjunto con el transcurso del experimento. Probablemente esta correlación puede ser mejorada mediante un pre-tratamiento de estas curvas, sea mediante filtros, observaciones de secciones reducidas de los perfiles, integración de nuevos parámetros, entre otros. Estas posibilidades no fueron exploradas debido a que la necesidad de estimar el numero de disparos transcurridos es nula en un sistema de monitoreo continuo en el que se tiene un catastro de todos los datos y, por ende, los disparos transcurridos. Sin embargo este análisis puede servir para estimar a grandes rasgos la vida transcurrida de un equipo que no esté siendo monitoreado, a partir de la variación porcentual de los parámetros.

Por otro lado, el fenómeno de Dip es de interés no solo porque presenta un indicador del estado del equipo PF, sino que es una área de estudio que presenta mucho interés en la física nuclear y tiene implicancias en muchas otras áreas de la física. El análisis aquí presentado permitiría un nuevo enfoque a las condiciones que causan este fenómeno. Mediante la variación de parámetros experimentales se pueden obtener distintas curvas de probabilidad de Dip, esto se puede ver en las diferencias de forma y escala de estas en los experimentos preliminares y finales. La diferencia en las predicciones de disparos antes de la falla se puede deber a que en el experimento final se operó a una presión de $3[mbar]$ a diferencia de los $8[mbar]$ del experimento preliminar. Mayores densidades de gases conllevan a mas reacciones entre partículas en el momento de la ionización, que

[48] Diagrama generado en `draw.io`

pueden explicar el hecho que el experimento preliminar tuvo una vida útil mucho menor que el posterior, comprendiendo que el modo de falla es la deposición de partículas y la erosión superficial del aislante a medida que se dispara. Esto también podría explicar el hecho que el aislante permite el fenómeno de Dip en un instante posterior en el experimento final ($N = 3000$) comparado con el preliminar ($N = 1000$).

En cuanto al entrenamiento de las redes neuronales es posible identificar distintos tipos de Dip en los experimentos. Por ende se puede categorizar ciertos datos que presenten un tipo de Dip particular, como los ilustrados en 7.1 y 7.2 e identificar como varían estos a lo largo de los experimentos para poder concluir más sobre la física involucrada y los distintos parámetros que la condicionan.

10. Conclusiones

A partir del análisis preliminar se puede ver que a pesar de haber varias fuentes de ruido en los datos, junto con eventos que han generado quiebres en estos, se puede notar de los gráficos presentados en la sección de resultados preliminares que efectivamente existen parámetros que muestran comportamientos monótonamente crecientes o decrecientes.

Considerando que el amortiguamiento de la señal disminuye y los indicadores de forma (shape), despeje (clearance) y kurtosis aumentan; se puede caracterizar este fenómeno como uno mismo: la señal se estira, aumentando su período levemente. Esto explica la evolución de estos parámetros de forma simultánea.Por otro lado, resulta una buena opción basarse en una gama de parámetros que hacen evidencia del mismo fenómeno ya que aseguran que sea una tendencia global y no una corrupción de los datos. Esto es esencial para realizar una predicción utilizando NN, ya que minimiza las fuentes de ruido.

De el análisis preliminar se puede concluir que se pueden realizar predicciones a grandes rasgos sobre los disparos efectuados mediante la observación de ciertos parámetros clave. En particular, de Kurtosis, Shape Indicator, Clearance Indicator, Crest Indicator y Amortiguamiento de las curvas V e $\dot{I}$.

De la ocurrencia de Dip se puede concluir que la probabilidad de este fenómeno llega a un máximo bastante posterior a la puesta en marcha del equipo, para luego deteriorarse y decrecer de forma gradual.

En cuanto al análisis final de este fenómeno se puede ver como esta probabilidad corresponde a la disponibilidad del equipo a lo largo de un experimento. En particular, esta distribución de probabilidad sigue un perfil de Weibull cuyos parámetros dependen de las condiciones experimentales.

Las curvas de Disponibilidad extrapoladas del equipo pueden ser obtenidas a lo largo de los disparos, volviéndose cada vez mas certeras para luego converger a la distribución final. Lo mismo se puede asegurar de las predicciones de momentos de falla del equipo, dado un umbral arbitrario. En este análisis se contempla un umbral de probabilidad de Dip del $5\,\%$ y se puede ver como los pronósticos realizados convergen a una predicción de falla certera mucho antes de que se alcance esta última. En cuanto a las predicciones de falla de los datos finales se estima que el experimento continuará reduciendo su Disponibilidad hasta alcanzar este límite alrededor de los $20{,}000$ disparos.

De los entrenamientos de las redes neuronales se puede concluir que el pre-tratamiento de las señales características es fundamental para una convergencia mínima de los errores de ultima capa durante el entrenamiento. En la misma linea, se puede concluir que mediante el entrenamiento realizado en base a $2{,}000$ de los datos preliminares se obtuvieron asertividades de $93{,}25\,\%$ para los mismos datos y $83{,}3\,\%$ para los datos de la experimentación final.

En base de todos los análisis presentados en este informe se logran concluir sistemas de mantenimiento predictivo que permiten predecir los momentos de falla del equipo PF-2J, en conjunto con mostrar las distribuciones de probabilidad de ocurrencia de Dip instantáneas a lo largo de un experimento, evidenciando el desgaste sobre el aislante.

11. Bibliografía

[1] Charles E. Ebeling. 1997. *An Introduction to Reliability and Maintainability Engineering*

[2] D. H. Stamatis. 1995. *Failure Mode and Effect Analysis: FMEA from Theory to Execution*

[3] Raymond J. Mikulak, Robin McDermott, Michael Beauregard. 1996. *The Basics of FMEA, 2nd Edition*

[4] Grandell, J. 1997. *Mixed Poisson Processes*

[5] Mahadevan Krishnan. 2012. *The Dense Plasma Focus: A Versatile Dense Pinch for Diverse Applications*

[6] Kwok L. Tsui, Nan Chen, Qiang Zhou, Yizhen Hai, Wenbin Wang. 2015. *Prognostics and Health Management: A Review on Data Driven Approaches*

[7] Leopoldo Soto, Cristian Pavez, José Pedreros, Jalaj Jain, Fermín Castillo,Patricio San Martín, Fabian Reyes, Luis Altamirano. 2016. *Plasma Dynamics in an Extreme Miniaturized Plasma Focus Operating at 2 Joules*

[8] Viviana Meruane. *Dinamica Estructural, Apuntes para el Curso ME706*

[9] Elaheh Rabiei, Enrique Lopez Droguett, and Mohammad Modarres. 2016. *A prognostics approach based on the evolution of damage precursors using dynamic Bayesian networks*

[10] Enrique López Droguett. *ME 5702 – Physical asset management, system reliability analysis*

[11] Leopoldo Soto, Cristian Pavez, Ariel Tarifeño, José Moreno and Felipe Veloso, Plasma Sources Sci. and Technol. 2010. *Studies on scalability and scaling laws for the plasma focus: similarities and differences in devices from 1MJ to 0.1J*

[12] Leopoldo Soto, Plasma Physics and Controlled Fusion. 2005. *New Trends and Future Perspectives on Plasma Focus Research*

[13] ", Cristian Pavez, Leopoldo Soto, José Moreno, Ariel Tarifeño and Gustavo Sylvester, Journal of Physics: Conf. Series. 2008. *Progress in Z-pinch research driven by the mega-ampere device SPEED2*

[14] P. Silva, J. Moreno, L. Soto, L. Birstein, R. Mayer, W. Kies, Applied Physics Letters. 2003. *Neutron Emission from a Fast Plasma Focus of 400 Joules*

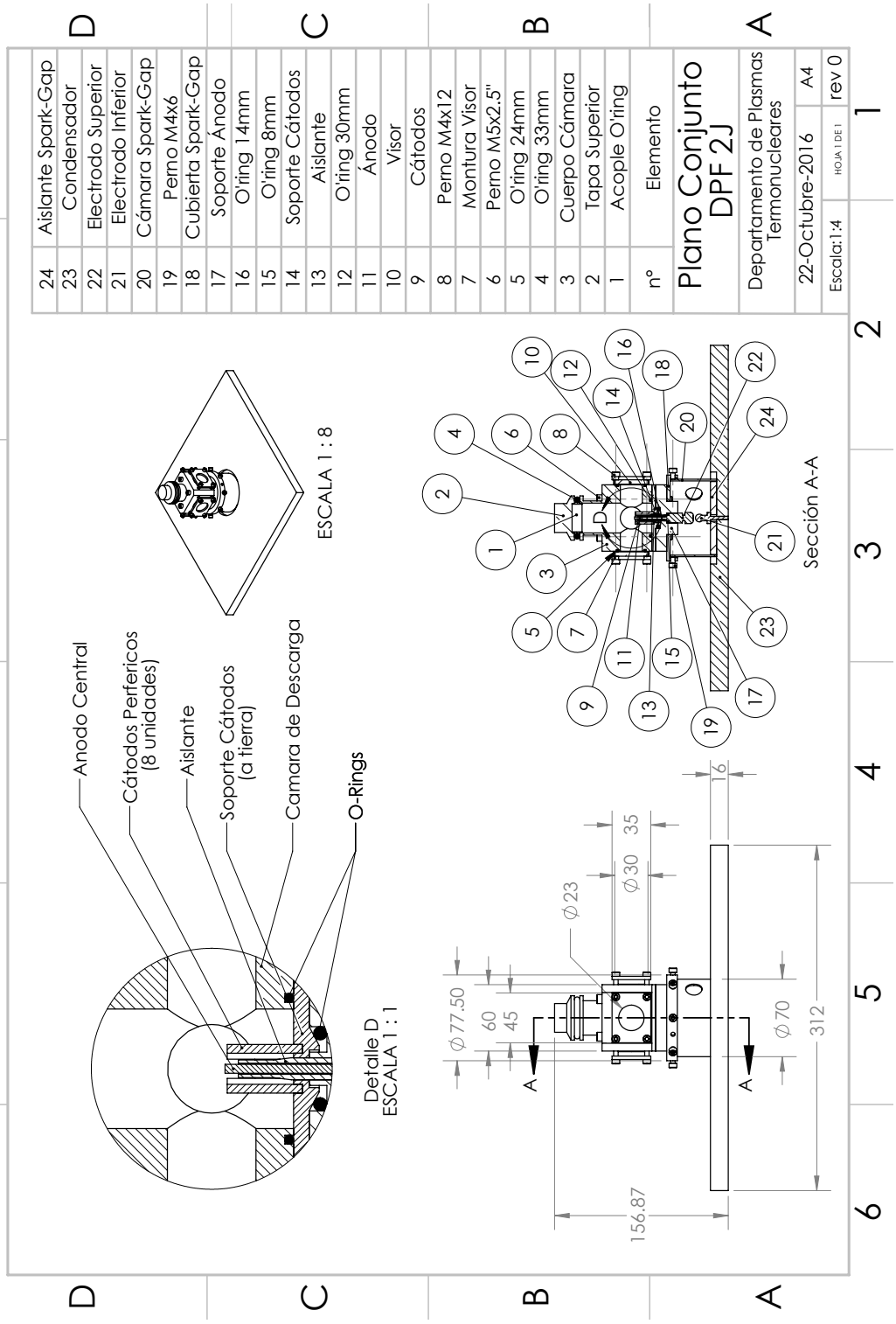
[15] Leopoldo Soto, Patricio Silva, José Moreno Moreno, Marcelo Zambra, Walter Kies, Roberto E. Mayer, Alejandro Clausse, Luis Altamirano, Cristian Pavez, and Luis Huerta J. Phys. D: App. Phys. 2008. *Demonstration of neutron production in a table top pinch plasma focus device operated at only tens of joules*

[16] ", L Soto, C Pavéz, J Moreno, J Pedreros and L Altamirano, Journal of Physics: Conference Series 511. 2014. *Non-radioactive Source for Field Applications Based in a Plasma Focus of 2J: Pinch evidence*

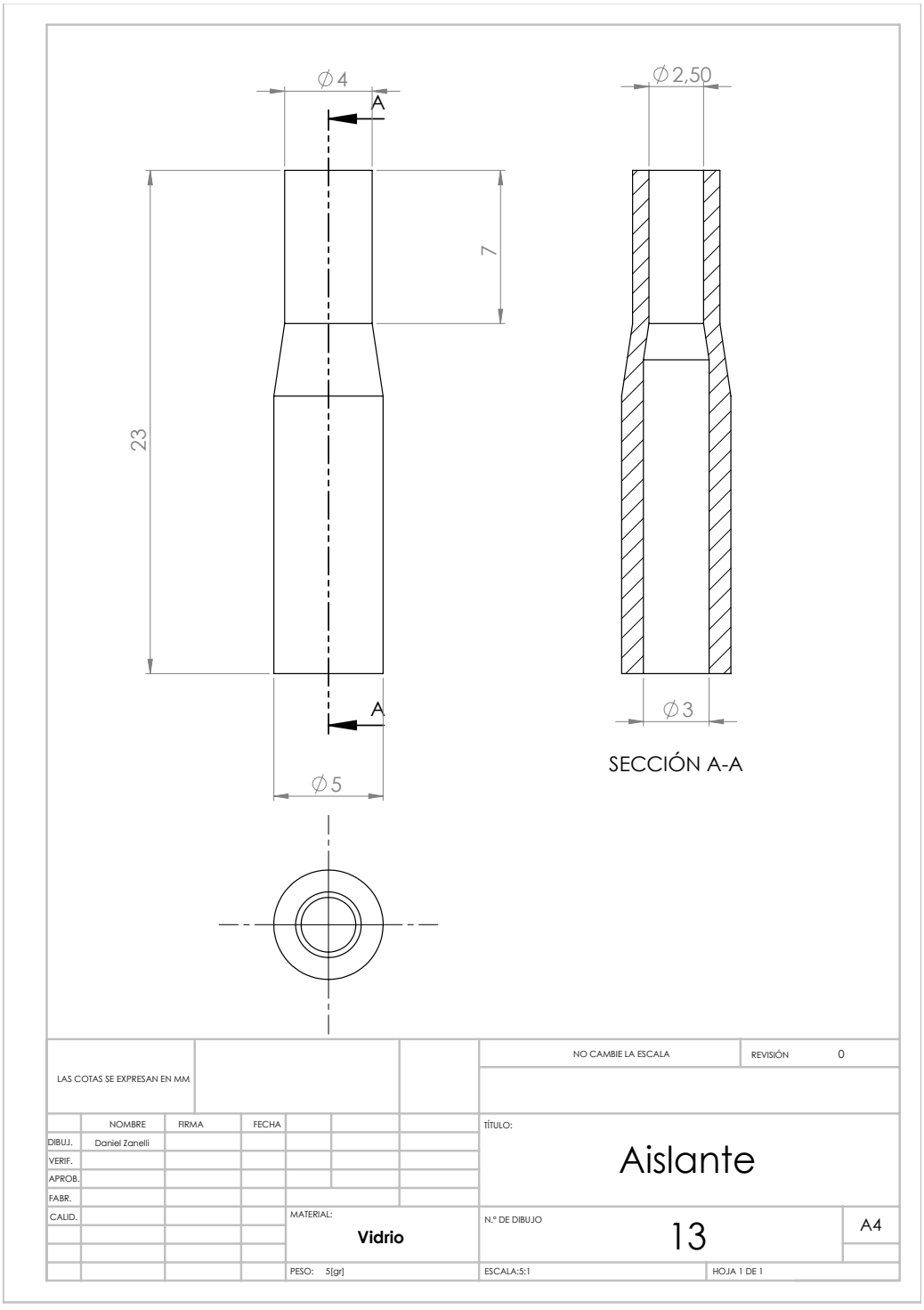
[17] Leopoldo Soto, Cristian Pavéz, José Moreno, Luis Altamirano, Luis Huerta, Mario Barbaglia, Alejandro Clausse, and Roberto E. Mayer, Physics of Plasma 24. 2017. *Evidence of nuclear fusion neutrons in an extremely small plasma focus device*

12. Anexos

A. Plano Conjunto de Equipo PF-2J



B. Plano de Aislante de PF-2J



C. Código Python para Ventana de Selección de Datos con 'Dip'

```
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 12 15:30:34 2016

@author: Daniel Zanelli
"""

import sys
import matplotlib
matplotlib.use("Qt5Agg")
from PyQt5 import QtCore, QtWidgets
from matplotlib.figure import Figure

import csv
import os

import numpy
from Tkinter import Tk
import tkFileDialog

import threading

from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg

class MatplotlibWidget(QtWidgets.QWidget):
    def __init__(self, parent=None):
        super(MatplotlibWidget, self).__init__(parent)
        self.figure = Figure((80.0,50.0),dpi=50)
        self.canvas =FigureCanvasQTAgg(self.figure)
        self.axis = self.figure.add_subplot(111,label=self.label)
        self.layoutVertical = QtWidgets.QVBoxLayout(self)
        self.layoutVertical.addWidget(self.canvas)

class Ui_MainWindow(object):

    def __init__(self, parent=None, width=100, height=80, dpi=30):
        fig = Figure(figsize=(width, height), dpi=dpi)
        self.axes = fig.add_subplot(111)
        self.axes.hold(False)
        self.Indice=[[],[],[]]
        self.disparo=0
        self.dip='Pendiente'
        self.filename=''

    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1544, 816)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout = QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout.setObjectName("gridLayout")
        self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)
```

```
self.tabWidget.setObjectName("tabWidget")
self.tab = QtWidgets.QWidget()
self.tab.setObjectName("tab")

self.plt1 = MatplotlibWidget(self.tab)
self.plt1.setGeometry(QtCore.QRect(10, 20, 811, 300))
self.plt1.setObjectName("Grafico1")


self.plt2 = MatplotlibWidget(self.tab)
self.plt2.setGeometry(QtCore.QRect(10, 300, 811, 300))
self.plt2.setObjectName("Grafico2")


self.numdisp = QtWidgets.QTextBrowser(self.tab)             #Numero de Archivo
self.numdisp.setGeometry(QtCore.QRect(1040, 75, 41, 31))
self.numdisp.setObjectName("spinBox")
self.numdisp.setText('')

self.archivo = QtWidgets.QTextBrowser(self.tab)         #Nombre de archivo
self.archivo.setGeometry(QtCore.QRect(870, 110, 351, 31))
self.archivo.setObjectName("textBrowser")
self.archivo.setText('')


self.label = QtWidgets.QLabel(self.tab)
self.label.setGeometry(QtCore.QRect(1037, 50, 46, 13))
self.label.setObjectName("label")


self.pushButton = QtWidgets.QPushButton(self.tab)           #BOTONES
self.pushButton.setGeometry(QtCore.QRect(970, 320, 81, 41))
self.pushButton.setObjectName("pushButton")
self.pushButton.clicked.connect(self.SI)

self.pushButton_2 = QtWidgets.QPushButton(self.tab)
self.pushButton_2.setGeometry(QtCore.QRect(1100, 320, 81, 41))
self.pushButton_2.setObjectName("pushButton_2")
self.pushButton_2.clicked.connect(self.NO)

self.pushButton_3 = QtWidgets.QPushButton(self.tab)
self.pushButton_3.setGeometry(QtCore.QRect(1184, 440, 101, 23))
self.pushButton_3.setObjectName("pushButton_3")
self.pushButton_3.clicked.connect(self.Abrir)

self.pushButton_4 = QtWidgets.QPushButton(self.tab)
self.pushButton_4.setGeometry(QtCore.QRect(1184, 470, 101, 23))
self.pushButton_4.setObjectName("pushButton_4")
self.pushButton_4.clicked.connect(self.Guardar)

self.pushButton_5 = QtWidgets.QPushButton(self.tab)
self.pushButton_5.setGeometry(QtCore.QRect(1120, 80, 81, 23))
self.pushButton_5.setObjectName("pushButton_5")
```

```
        self.pushButton_5.clicked.connect(self.Siguiente)

        self.pushButton_6 = QtWidgets.QPushButton(self.tab)
        self.pushButton_6.setGeometry(QtCore.QRect(920, 80, 75, 23))
        self.pushButton_6.setObjectName("pushButton_6")
        self.pushButton_6.clicked.connect(self.Anterior)

        self.pushButton_7 = QtWidgets.QPushButton(self.tab)
        self.pushButton_7.setGeometry(QtCore.QRect(1184, 500, 101, 23))
        self.pushButton_7.setObjectName("pushButton_7")
        self.pushButton_7.clicked.connect(self.Abririndice)


        self.estado = QtWidgets.QTextBrowser(self.tab)              #Estado de datos
        self.estado.setGeometry(QtCore.QRect(1025, 190, 71, 31))
        self.estado.setObjectName("textBrowser_2")
        self.estado.setText('')

        self.label_2 = QtWidgets.QLabel(self.tab)
        self.label_2.setGeometry(QtCore.QRect(1037, 170, 46, 13))
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(self.tab)
        self.label_3.setGeometry(QtCore.QRect(1000, 260, 151, 31))
        self.label_3.setObjectName("label_3")
        self.progressBar = QtWidgets.QProgressBar(self.tab)
        self.progressBar.setGeometry(QtCore.QRect(900, 470, 271, 23))
        self.progressBar.setProperty("value", 0)
        self.progressBar.setObjectName("progressBar")
        self.tabWidget.addTab(self.tab, "")
        self.tab_2 = QtWidgets.QWidget()
        self.tab_2.setObjectName("tab_2")
        self.tabWidget.addTab(self.tab_2, "")
        self.gridLayout.addWidget(self.tabWidget, 0, 1, 1, 1)
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1044, 21))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        self.tabWidget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.label.setText(_translate("MainWindow", "Archivo"))
        self.pushButton.setText(_translate("MainWindow", "SI"))
        self.pushButton_2.setText(_translate("MainWindow", "NO"))
        self.pushButton_3.setText(_translate("MainWindow", "Abrir Carpeta"))
```

```
        self.pushButton_4.setText(_translate("MainWindow", "Guadar Indice"))
        self.label_2.setText(_translate("MainWindow", "Estado"))
        self.label_3.setText(_translate("MainWindow", "Estos Gráficos Presentan \'Dip\'?"))
        self.pushButton_5.setText(_translate("MainWindow", "-->"))
        self.pushButton_6.setText(_translate("MainWindow", "<--"))
        self.pushButton_7.setText(_translate("MainWindow", "Abrir Indice"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab), _translate("MainWindow", "Tab 1"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_2), _translate("MainWindow", "Tab 2"))

    def launch_Thread(self):
        t = threading.Thread(target=self.Plot)
        t.start()

    def SI(self):
        self.dip=1
        self.Indice[2][self.disparo]=self.dip
        self.Siguiente()

    def NO(self):
        self.dip=0
        self.Indice[2][self.disparo]=self.dip
        self.Siguiente()


    def Siguiente(self):
        print len(self.Indice[1])
        if self.disparo+1>self.Indice[1][len(self.Indice[1])-1]:
            print 'NO HAY SIGUIENTE'
        else:
            self.disparo+=1
            self.dip=self.Indice[2][self.disparo]
            self.filename=self.Indice[0][self.disparo]
            self.numdisp.setText(str(self.disparo))
            self.archivo.setText(self.Indice[0][self.disparo])
            self.estado.setText(str(self.dip))
            self.Plot()
            print self.disparo

    def Anterior(self):
        print self.disparo
        if self.disparo==0:
            print 'NO HAY ANTERIOR'
        else:
            self.disparo+=-1
            self.dip=self.Indice[2][self.disparo]
            self.filename=self.Indice[0][self.disparo]
            self.numdisp.setText(str(self.disparo))
            self.archivo.setText(self.Indice[0][self.disparo])
            self.estado.setText(str(self.dip))
            self.Plot()

    def Abrir(self):
```

```
    print'Abriendo Archivos'
    Tk().withdraw()
    self.foldername = tkFileDialog.askdirectory()
    print self.foldername
    self.Indice=[[],[],[]]
    folders=[]
    i=0
    for root, dirs, files in os.walk(self.foldername, topdown=True):
        for name in dirs:
            folders.append(name)
        for filename in files:
            if filename[(len(filename)-3):]=='csv':
                i=i+1
    total_archivos=i
    i=0

    for carpeta in folders:
        for root, dirs, files in os.walk(self.foldername+'/'+carpeta, topdown=True):
            for filename in files:
                if filename[(len(filename)-3):]=='csv':

                    print 'Archivo numero ', i
                    print 'Avance: ', int(100*float(i)/float(total_archivos)), '%'
                    self.Indice[0].append(self.foldername+'/'+carpeta+'/'+filename)
                    self.Indice[1].append(i)
                    self.Indice[2].append('Pendiente')
                    self.progressBar.setValue(int(100*float(i)/float(total_archivos)))
                    i=i+1

    print 'Nuevo Indice ha sido creado'
    self.filename=self.Indice[0][0]
    self.disparo=0
    self.dip='Pendiente'
    self.numdisp.setText(str(self.disparo))
    self.archivo.setText(self.Indice[0][self.disparo])
    self.estado.setText(str(self.dip))
    self.Plot()

def Plot(self):

    print'Ploteando'

    i=0
    QtCore.QCoreApplication.processEvents()
    print self.filename
    t=[]
    V=[]
    dI=[]
    titulo=True
    i=-1

    with open(self.filename,'rb') as csvfile:
        spamreader = csv.reader(csvfile, delimiter=',', quotechar='.')
```

```
            for row in spamreader:
                if titulo==True:
                    titulo=False
                    continue
                if titulo==False:
                    t.append(float(row[0])*10**9)
                    V.append(row[1])
                    dI.append(row[3])
                    i=i+1
        i=0
        for x in V:
            V[i]=float(x)
            i+=1


        i=0
        for x in dI:
            dI[i]=float(x)
            i+=1


        Vprom=numpy.mean(V)
        i=0
        for valor in V:
            V[i]=float(valor)-Vprom
            i=i+1

        dIprom=numpy.mean(dI)
        i=0
        for valor in dI:
            dI[i]=float(valor)-dIprom
        i=i+1

        self.V=V
        self.dI=dI
        self.plt1.axis.clear()
        self.plt1.axis.plot(self.dI)
        self.plt1.canvas.draw()
        self.plt2.axis.clear()
        self.plt2.axis.plot(self.V)
        self.plt2.canvas.draw()

        self.progressBar.setValue(0)


    def Abririndice(self):

        print'ABRIENDO INDICE'
        Tk().withdraw()
        self.file = tkFileDialog.askopenfilename( defaultextension=".csv")
        self.Indice=[[],[],[]]
        titulo=True
```

```
        i=0
        with open(self.file,'rb') as csvfile:
            spamreader = csv.reader(csvfile, delimiter=',', quotechar='.')
            for row in spamreader:

                if titulo==True:

                    titulo=False
                    continue
                if titulo==False:
                    self.Indice[0].append(row[0])
                    self.Indice[1].append(float(row[1]))
                    self.Indice[2].append(row[2])
                    i=i+1


        i=0
        for valor in self.Indice[2]:
            print valor

            if '0' in valor or '1' in valor:
                print valor
                i=i+1
                continue

            else:
                self.disparo=i
                self.dip=self.Indice[2][i]
                self.filename=self.Indice[0][self.disparo]
                self.numdisp.setText(str(self.disparo))
                self.archivo.setText(self.Indice[0][self.disparo])
                self.estado.setText(str(self.dip))
                self.Plot()
                break


    def Guardar(self):

        Tk().withdraw()
        self.file = tkFileDialog.asksaveasfilename( defaultextension=".csv")

        with open(self.file, 'w') as f:

            f.write('Archivo, Disparo, Dip\n')
            print'guardando'
            for i in range(0,len(self.Indice[0])):
                f.write(str(self.Indice[0][i])+', ' +str(self.Indice[1][i])+', ' +str(self.Indice[2][i])+'
                self.progressBar.setValue(int(100*float(i)/float(len(self.Indice[0]))))
                print i
        self.progressBar.setValue(0)

if __name__ == "__main__":
```

```
app = QtWidgets.QApplication(sys.argv)
MainWindow = QtWidgets.QMainWindow()
ui = Ui_MainWindow()
ui.setupUi(MainWindow)
MainWindow.show()
app.exec_()
sys.exit(app.exec_())
```

D. Código Python para Feature Extraction de Perfiles Característicos

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 27 16:23:15 2016

@author: Daniel Zanelli
"""

import csv
import scipy.fftpack
from Tkinter import Tk
import tkFileDialog
import matplotlib.pyplot as plt
import os
import math
import numpy

def MaxMin(t,V,dI):

    maximo=0                #Máximos y mínimos de dI
    minimo=0
    i=0
    for valor in dI:
        if float(valor)>float(maximo):
            maximo=valor
        if float(valor)<float(minimo):
            minimo=valor
        i=i+1
    dImax=maximo
    dImin=minimo

    maximo=0            #Máximos y mínimos de V
    minimo=0
    i=0
    for valor in V:
        if float(valor)>float(maximo):
            maximo=valor
        if float(valor)<float(minimo):
            minimo=valor
        i=i+1
    Vmax=maximo
    Vmin=minimo
    return dImax,'Maximo dI/dt',dImin,'Minimo dI/dt',Vmax,'Maximo Voltaje',Vmin,'Minimo Voltaje'



def KurtCrest(t,V,dI):


    suma=0              #Calculo de coeficiente de Crest
    for x in V:
        suma+=float(x)**2
    CrestV=float(max(V))/(numpy.sqrt(suma/len(V)))
```

```
    suma=0
    for x in dI:
        suma+=float(x)**2
    CrestdI=float(max(dI))/(numpy.sqrt(suma/len(dI)))


    suma=0              #Calculo de Kurtosis
    promedio=numpy.mean(V)
    for x in V:
        suma+=(float(x)-promedio)**4
    KurtosisV=suma/((len(V)-1)*(numpy.std(V)**4))


    suma=0
    promedio=numpy.mean(dI)
    for x in dI:
        suma+=(float(x)-promedio)**4
    KurtosisdI=suma/((len(dI)-1)*(numpy.std(dI)**4))


    return CrestdI,'Crest Indicator dI/dt',KurtosisdI,'Kurtosis dI/dt',CrestV,'Crest Indicator Voltaje',Ku



def ShapeClear(t,V,dI):

    suma1=0              #Calculo de Shape Indicator
    suma2=0
    for x in V:
        suma1+=x**2
        suma2+=abs(x)
    ShapeV=numpy.sqrt(suma1/len(V))/(suma2/len(V))

    suma1=0
    suma2=0
    for x in dI:
        suma1+=x**2
        suma2+=abs(x)
    ShapedI=numpy.sqrt(suma1/len(dI))/(suma2/len(dI))


    suma=0              #Calculo de Clearance Indicator

    for x in V:
        suma+=numpy.sqrt(abs(x))
    ClearanceV=max(V)/((suma/len(V))**2)


    suma=0
    for x in dI:
        suma+=numpy.sqrt(abs(x))
    ClearancedI=max(dI)/((suma/len(dI))**2)
```

```
    return ShapedI,'Shape Indicator dI/dt',ClearancedI,'Clearance Indicator dI/dt',ShapeV,'Shape Indicator


def Fourier(t,V,dI):



    dt=(t[len(t)-1]-t[0])/len(t)

    N = len(V)
    Vf = scipy.fftpack.fft(V)/N
    Vf=Vf[:(len(Vf)/2)]
    Vfabs=[]
    for v in Vf:
        Vfabs.append(abs(v))


    N = len(dI)
    dIf = scipy.fftpack.fft(dI)/N
    dIf=dIf[:(len(dIf)/2)]
    dIfabs=[]
    for di in dIf:
        dIfabs.append(abs(di))

    peak=False
    for i in range(0,len(dIfabs)):
        if dIfabs[i]>=max(dIfabs)/numpy.sqrt(2)and peak==False:
            wb=(max(dIfabs)-dIfabs[i-1])/(dIfabs[i]-dIfabs[i-1])*dt
            peak=True
        elif dIfabs[i]<max(dIfabs)/numpy.sqrt(2) and peak==True:
            wa=(max(dIfabs)-dIfabs[i-1])/(dIfabs[i]-dIfabs[i-1])*dt
            peak=False
        if dIfabs[i]==max(dIfabs):
            wr=i

    chidI=(wb*wb-wa*wa)/(4*wr*wr)


    peak=False
    for i in range(0,len(Vfabs)):
        if Vfabs[i]>=max(Vfabs)/numpy.sqrt(2)and peak==False:
            wb=(max(Vfabs)-Vfabs[i-1])/(Vfabs[i]-Vfabs[i-1])*dt
            peak=True
        elif Vfabs[i]<max(Vfabs)/numpy.sqrt(2) and peak==True:
            wa=(max(Vfabs)-Vfabs[i-1])/(Vfabs[i]-Vfabs[i-1])*dt
            peak=False
        if Vfabs[i]==max(Vfabs):
            wr=i

    chiV=(wb*wb-wa*wa)/(4*wr*wr)
```

```
    entropy=0.0
    total=sum(Vfabs)
    for x in Vfabs:
        p_x=x/total
        if p_x>0:
            entropy += - p_x*math.log(p_x, 2)
    EntropiaV=entropy


    entropy=0.0
    total=sum(dIfabs)
    for x in dIfabs:
        p_x=x/total
        if p_x>0:
            entropy += - p_x*math.log(p_x, 2)
    EntropiadI=entropy


    return chidI,'Amortiguamiento dI/dt', EntropiadI,'Entropia dI/dt', chiV,'Amortiguamiento Voltaje', Ent
```

```
def FeatureExtraction(Archivo):

    t=[]
    V=[]
    dI=[]
    titulo=True
    i=-1

    with open(Archivo,'rb') as csvfile:
        spamreader = csv.reader(csvfile, delimiter=',', quotechar='.')

        for row in spamreader:
            if titulo==True:
                titulo=False
                continue
            if titulo==False:
                t.append(float(row[0])*10**9)
                V.append(row[1])
                dI.append(row[3])
                i=i+1

    i=0
    for x in V:
        V[i]=float(x)
        i+=1
```

```
    i=0
    for x in dI:
        dI[i]=float(x)
        i+=1


    Vprom=numpy.mean(V)
    i=0
    for valor in V:
        V[i]=float(valor)-Vprom
        i=i+1

    dIprom=numpy.mean(dI)
    i=0
    for valor in dI:
        dI[i]=float(valor)-dIprom
        i=i+1


    v1,tv1,v2,tv2,v3,tv3,v4,tv4=KurtCrest(t,V,dI)

    v5,tv5,v6,tv6,v7,tv7,v8,tv8=Fourier(t,V,dI)

    v9,tv9,v10,tv10,v11,tv11,v12,tv12=MaxMin(t,V,dI)

    v13,tv13,v14,tv14,v15,tv15,v16,tv16=ShapeClear(t,V,dI)

    return v1,tv1,v2,tv2,v3,tv3,v4,tv4,v5,tv5,v6,tv6,v7,tv7,v8,tv8,v9,tv9,v10,tv10,v11,tv11,v12,tv12,v13,t

print'Abriendo Archivos'

Tk().withdraw()
foldername = tkFileDialog.askdirectory()

print'Analizando'

folders=[]
i=0
Resultados=[]

for root, dirs, files in os.walk(foldername, topdown=True):
    for name in dirs:
        folders.append(name)
    for filename in files:
        if filename[(len(filename)-3):]=='csv':
            i=i+1

total_archivos=i

i=-1

for carpeta in folders:
    for root, dirs, files in os.walk(foldername+'/'+carpeta, topdown=True):
```

```
        for filename in files:
            if filename[(len(filename)-3):]=='csv':

                i=i+1

                print 'Archivo numero ', i
                print 'Avance: ', int(100*float(i)/float(total_archivos)), '%'
                Resultados.append(FeatureExtraction(foldername+'/'+carpeta+'/'+filename))

v1=[]
v2=[]
v3=[]
v4=[]
v5=[]
v6=[]
v7=[]
v8=[]
v9=[]
v10=[]
v11=[]
v12=[]
v13=[]
v14=[]
v15=[]
v16=[]

tv1=Resultados[1][1]
tv2=Resultados[1][3]
tv3=Resultados[1][5]
tv4=Resultados[1][7]
tv5=Resultados[1][9]
tv6=Resultados[1][11]
tv7=Resultados[1][13]
tv8=Resultados[1][15]
tv9=Resultados[1][17]
tv10=Resultados[1][19]
tv11=Resultados[1][21]
tv12=Resultados[1][23]
tv13=Resultados[1][25]
tv14=Resultados[1][27]
tv15=Resultados[1][29]
tv16=Resultados[1][31]


for j in range(0,len(Resultados)):

    v1.append(Resultados[j][0])
    v2.append(Resultados[j][2])
    v3.append(Resultados[j][4])
    v4.append(Resultados[j][6])
    v5.append(Resultados[j][8])
    v6.append(Resultados[j][10])
    v7.append(Resultados[j][12])
```

```
    v8.append(Resultados[j][4])
    v9.append(Resultados[j][16])
    v10.append(Resultados[j][18])
    v11.append(Resultados[j][20])
    v12.append(Resultados[j][22])
    v13.append(Resultados[j][24])
    v14.append(Resultados[j][26])
    v15.append(Resultados[j][28])
    v16.append(Resultados[j][30])
print'Resultados Calculados'

print'Guardando Resultados'


Tk().withdraw()
file = tkFileDialog.asksaveasfilename( defaultextension=".csv")

with open(file, 'w') as f:
    f.write(str(tv1)+', ' +str(tv2)+', ' +str(tv3)+', ' +str(tv4)+', '+ str(tv5)+', ' +str(tv6)+', ' +str(

    print'guardando'
    for i in range(0,len(v1)):
        f.write(str(v1[i])+', ' +str(v2[i])+', ' +str(v3[i])+', ' +str(v4[i])+', '+ str(v5[i])+', ' +str(v
```

E. Código Python para Limpieza de Datos Repetidos

```
# -*- coding: utf-8 -*-
"""
Created on Mon Mar 27 12:32:51 2017

@author: Daniel Zanelli
"""
from shutil import copyfile

from Tkinter import Tk
import tkFileDialog
import os
import csv



def archivosiguales(archivo1,archivo2):

    t=[]
    V=[]
    dI=[]
    titulo=True
    i=0
    limite=10

    with open(archivo1,'rb') as csvfile:
        spamreader = csv.reader(csvfile, delimiter=',', quotechar='.')

        for row in spamreader:
            if i>=limite: break
            if titulo==True:
                titulo=False
                continue
            if titulo==False:
                t.append(float(row[0]))
                V.append(float(row[1]))
                dI.append(float(row[3]))
                i=i+1

    titulo=True
    i=0
    with open(archivo2,'rb') as csvfile:
        spamreader = csv.reader(csvfile, delimiter=',', quotechar='.')

        for row in spamreader:
            if i>=limite: break
            if titulo==True:
                titulo=False
                continue
            if titulo==False:

                if t[i]!=float(row[0]):
                    return False
```

```
                if V[i]!=float(row[1]):
                    return False
                if dI[i]!=float((row[3])):
                    return False
                i=i+1

    print 'Archivos iguales encontrados!\n',archivo2,'\n',archivo1,'\n'
    return True



print'Escoger Carpeta de Datos Repetidos'

Tk().withdraw()
folderdatos = tkFileDialog.askdirectory()

print folderdatos

print'Escoger Carpeta Donde Guardar'

Tk().withdraw()
folderfinal = tkFileDialog.askdirectory()

print folderfinal


print'Analizando'

folders=[]
i=0

for root, dirs, files in os.walk(folderdatos, topdown=True):
    for name in dirs:
        folders.append(name)
    for filename in files:
        if filename[(len(filename)-3):]=='csv':
            i=i+1


total_archivos=i


i=1
anterior=''
for root, dirs, files in os.walk(folderdatos, topdown=True):
    for filename in files:
        if filename[(len(filename)-3):]=='csv':

            print 'Archivo numero ', i
            print 'Avance: ', int(100*float(i)/float(total_archivos)), '%'

            if i==1:
                copyfile(folderdatos+'/'+filename, folderfinal+'/'+filename)
```

```
            else:
                respuesta=archivosiguales(folderdatos+'/'+filename,folderdatos+'/'+anterior)
#                print  respuesta,'\n',folderdatos+'/'+anterior,'\n',folderdatos+'/'+filename,'\n'

                if respuesta==False:
                    copyfile(folderdatos+'/'+filename, folderfinal+'/'+filename)

            anterior=filename
            i=i+1
```

F. Código Python para Lectrura y Pre-tratamiento de Datos

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 31 20:39:33 2017

@author: Daniel Zanelli
"""

import csv
import numpy as np
import matplotlib.pyplot as plt
from Tkinter import Tk
import tkFileDialog
from scipy import signal
import tkMessageBox
import os
from scipy.optimize import curve_fit

root = Tk()
root.withdraw()
root.overrideredirect(True)
root.geometry('0x0+0+0')
root.deiconify()
root.lift()
root.focus_force()
result = tkMessageBox.askquestion("Indice Dip", "Abrir Indice Dip?")
root.destroy()
archivos=[]

if result=="yes":
    root = Tk()
    root.withdraw()
    root.overrideredirect(True)
    root.geometry('0x0+0+0')
    root.deiconify()
    root.lift()
    root.focus_force()
    filename=tkFileDialog.askopenfilename(parent=root,title="Indice de Dip")
    root.destroy()

######################### Abriendo indice Dip ###################


    Dip=[]
    titulo=True
    total=0
    with open(filename,'rb') as csvfile:
        spamreader = csv.reader(csvfile, delimiter=',', quotechar='.')

        for row in spamreader:
            if titulo==True:
                titulo=False
                continue
```

```
            if titulo==False:
                try:
                    Dip.append(int(float(row[2])))
                    archivos.append(row[0])
                    total=total+1
                except:
                    break



######################## Abriendo Archivos de Indice ###################
else:

    root = Tk()
    root.withdraw()
    root.overrideredirect(True)
    root.geometry('0x0+0+0')
    root.deiconify()
    root.lift()
    root.focus_force()
    foldername = tkFileDialog.askdirectory(parent=root,title="Carpeta de Datos")
    root.destroy()

    folders=[]
    i=0
    for root, dirs, files in os.walk(foldername, topdown=True):
        for name in dirs:
            folders.append(name)
        for filename in files:
            if filename[(len(filename)-3):]=='csv':
                i=i+1


    total_archivos=i

    i=-1

    for carpeta in folders:
        for root, dirs, files in os.walk(foldername+'/'+carpeta, topdown=True):
            for filename in files:
                if filename[(len(filename)-3):]=='csv':

                    i=i+1
                    archivos.append(foldername+'/'+carpeta+'/'+filename)


def OscAmort(x,gamma,A,omega):
    return (np.e**(-gamma*x/1000))*A*np.cos(omega*x/100)

def Norm01(Y):
    return (Y-min(Y))/(max(Y)-min(Y))

def Norm0max(Y):
```

```
    return (Y/max(abs(Y))-np.mean(Y/max(abs(Y))))

def Filtrar(Y,n):
    b = [1.0 / n] * n
    a = 1
    Y = signal.filtfilt(b,a,Y)
    return Y
tdip=int(raw_input("t dip? "))

Datos=[]
Features=[]
i=0
for f in archivos:
    t=[]
    V=[]
    dI=[]
    titulo=True
    with open(f,'rb') as csvfile:
        spamreader = csv.reader(csvfile, delimiter=',', quotechar='.')

        for row in spamreader:
            if titulo==True:
                titulo=False
                continue
            if titulo==False:
                t.append(float(row[0])*10**9)
                V.append(float(row[1]))
                dI.append(float(row[3]))

    t=np.array(t)
    dI=np.array(dI)
    V=np.array(V)
    dI=dI-np.mean(dI)
    V=V-np.mean(V)

######################## Seleccion de Datos ###################

    limite=0.95*float(max(dI))
    i0=0
    for valor in dI:
        i0=i0+1
        if valor>limite:
            break


    if i0>=len(V):
        continue


    AV=np.mean(V[i0:i0+10])
    AdI=np.mean(dI[i0:i0+10])

    Vnorm=V[i0:]/AV
```

```
    dInorm=dI[i0:]/AdI
    tnorm=t[i0:]


    dInorm = Filtrar(dInorm,200)
    Vnorm = Filtrar(Vnorm,200)

    V=V[i0:]
    dI=dI[i0:]
    t=t[i0:]


############### Descomentar Para Usar Extrapolacion en vez de Filtro

#    gammaV=1000*0.001416
#    gammadI=1000*0.00733
#
#    omegaV=omegadI=100*0.0157
#    valsV, covarV = curve_fit(OscAmort, tnorm, Vnorm, p0=[gammaV,1,omegaV])
#    valsdI, covardI = curve_fit(OscAmort, tnorm, dInorm, p0=[gammadI,1,omegadI])
#
#    dInorm=np.concatenate((zeros(len(t)-len(dInorm)),dInorm))
#    Vnorm=np.concatenate((zeros(len(t)-len(Vnorm)),Vnorm))
#
#    dIfinal=dI-AdI*OscAmort(t,valsdI[0],valsdI[1],valsdI[2])
#    Vfinal=V-AV*OscAmort(t,valsV[0],valsV[1],valsV[2])
#
#
#    dIfinal-=np.mean(dIfinal)
#    Vfinal-=np.mean(Vfinal)
#    dIfinal=Norm11(dIfinal)
#    Vfinal=Norm11(Vfinal)

    dIfinal=Filtrar(dI-dInorm*AdI,10)
    Vfinal=Filtrar(V-Vnorm*AV,10)

    idip=0
    for valor in t:
        idip+=1
        if valor>tdip:
            break

    desde=idip-500
    hasta=idip+500


    if len(Vfinal)==0:
        continue

    n = 20
    b = [1.0 / n] * n
    a = 1
    dID = signal.lfilter(b,a,dIfinal)
```

```
    VD = signal.lfilter(b,a,Vfinal)

    tD=t[desde:hasta]
    dID=dID[desde:hasta]
    VD=VD[desde:hasta]


    if len(VD)==0:
        continue

    F2=np.multiply(dID,abs(1-VD))
    F2=Norm0max(F2)

################### Descomentar para Graficar Dato por Dato

#    fig, (ax1, ax2, ax3, ax4, ax5)= plt.subplots(5, sharex=True)
#    ax1.plot(t,dI/AdI,label="$\dot{I}$")
#    ax1.plot(t,dInorm,color='r',label="$\dot{I}$ filtrado")
#    ax1.legend()
##    ax1.plot(t,OscAmort(t,valsdI[0],valsdI[1],valsdI[2]),color='k')
#    ax2.plot(t,V/AV,label="$V$")
#    ax2.plot(t,Vnorm,color='r',label="$V$ filtrado")
#    ax2.legend()
##    ax2.plot(t,OscAmort(t,valsV[0],valsV[1],valsV[2]),color='k')
#    ax3.plot(t,dIfinal,color='k',label="$\dot{I}$ resultante")
#    ax3.legend()
#    ax4.plot(t,Vfinal,color='k',label="$V$ resultante ")
#    ax4.legend()
#    ax5.plot(tD,F2,color='g',label="$\dot{I}\cdot (1-V)$ entregado a NN")
#    ax5.legend()
#    plt.xlabel("tiempo [ns]")
#    plt.show()
#    kskjdgb=raw_input(":)")
#    if kskjdgb=="exit":
#        break



#################### Reduccion y Guardado de Datos ######################

    k=0
    DatosReducidos=[]
    for valor in F2:
        if k%10==0:
            DatosReducidos.append(valor)
        k+=1

    DatosReducidos=np.array(DatosReducidos)
    Datos.append(DatosReducidos)
    if result=="yes":
        print"Archivo ", i,"    Avance:   ", int(i*100.0/len(Dip)),"%"
    else:
        print"Archivo ", i,"    Avance:   ", int(i*100.0/total_archivos),"%"
```

```
    i=i+1

print "[COMPLETADO] Guardando a Archivos"

Datos=np.array(Datos)
if result=="yes":
    Dip=np.array(Dip)
Features=np.array(Features)

root = Tk()
root.withdraw()
root.overrideredirect(True)
root.geometry('0x0+0+0')
root.deiconify()
root.lift()
root.focus_force()
filedatos=tkFileDialog.asksaveasfilename(parent=root,title="Archivo de Datos")
if result=="yes":
        filedip=tkFileDialog.asksaveasfilename(parent=root,title="Archivo de Dip")
root.destroy()

np.save(filedatos,Datos)
if result=="yes":
    np.save(filedip,Dip)
```

G. Código Python para Entrenamiento Red Neuronal

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 31 20:28:36 2017

@author: Daniel Zanelli
"""


import matplotlib.pyplot as plt
import numpy as np

from Tkinter import Tk
import tkFileDialog

def sigmoid(x):
    output = 1/(1+np.exp(-x))
    return output

def sigmoid_output_to_derivative(output):
    return output*(1-output)


def shuffle(x,y):
    xRand=[]
    yRand=[]
    index_shuf = range(len(x))
    np.random.shuffle(index_shuf)
    for i in index_shuf:
        xRand.append(x[i])
        yRand.append(y[i])

    return xRand,yRand


def NN(V,dip,n,A,B,alphas=[0.1]):
    C=2
    X=V
    y=range(len(dip))
    i=0
    for d in dip:
        if d==0:
            y[i]=np.array([0,1])
        elif d==1:
            y[i]=np.array([1,0])
        i+=1

    Xrand,yrand=shuffle(X,y)
    alfa=1

    for alpha in alphas:
        print "\n################# Iterando con Alfa: " + str(alpha)+" ################"
        np.random.seed(1)
```

```
Errores=[]
Iteraciones=[]

W0 = 2*np.random.random((int(len(V[0])+1),A)) - 1
W1 = 2*np.random.random((A+1,B)) -1
W2 = 2*np.random.random((B+1,C)) - 1

for j in xrange(n):

    error=0
    for i in xrange(len(V)):

        # Defincion de las Capas
        L0 = np.append(Xrand[i],[1])
        L1 = np.append(sigmoid(np.dot(L0,W0)),[1])
        L2 = np.append(sigmoid(np.dot(L1,W1)),[1])
        L3 = sigmoid(np.dot(L2,W2))


        # Calculo de errores y Backpropagation
        L3_error = L3- yrand[i]
        L3_delta=L3_error*sigmoid_output_to_derivative(L3)

        L2_error=np.delete(L3_delta.dot(W2.T),B,0)
        L2_delta=L2_error*sigmoid_output_to_derivative(np.delete(L2,B,0))

        L1_error=np.delete(L2_delta.dot(W1.T),A,0)
        L1_delta=L1_error*sigmoid_output_to_derivative(np.delete(L1,A,0))

        W2 -= alpha * (L2.reshape(B+1,1).dot(L3_delta.reshape(1,C)))
        W1 -= alpha * (L1.reshape(A+1,1).dot(L2_delta.reshape(1,B)))
        W0 -= alpha * (L0.reshape(int(round(len(V[0])+1)),1).dot(L1_delta.reshape(1,A)))

        error+=np.mean(np.abs(L3_error))/len(V)

    Xrand,yrand=shuffle(Xrand,yrand)
    if (j%100)==0:
        Errores.append(error)
        Iteraciones.append(j)
        print "-Iteracion [",j,"]","    Error promedio :" + str(error)


plt.figure(alfa)
plt.plot(Iteraciones,Errores)
plt.xlabel("Iteraciones")
plt.ylabel("Error de Capa Final")
plt.title("Alfa="+str(alpha)+" | Datos Entrenamiento="+str(len(V))+" | Iteraciones="+str(n-1))
plt.show()
alfa+=1
np.save("C:/Users/Dam/Documents/Tesis/Datos/Python/Data/w0-f"+str(len(V))+"-i"+str(n-1)+"-a"+str(a
np.save("C:/Users/Dam/Documents/Tesis/Datos/Python/Data/w1-f"+str(len(V))+"-i"+str(n-1)+"-a"+str(a
np.save("C:/Users/Dam/Documents/Tesis/Datos/Python/Data/w2-f"+str(len(V))+"-i"+str(n-1)+"-a"+str(a
```

```
    return W0,W1,W2


##############################  MAIN  ##############################



cantidad=int(float(raw_input("Cantidad de Datos de Entrenamiento? ")))
iteraciones=int(float(raw_input("Cantidad de Iteraciones? ")))
cantalfas= int(float(raw_input("Cantidad de Alfas? ")))

alfas=[]
for i in range(cantalfas):
    alfas.append(float(raw_input("Alfa Numero "+str(i+1)+"? ")))

root = Tk()
root.withdraw()
root.overrideredirect(True)
root.geometry('0x0+0+0')
root.deiconify()
root.lift()
root.focus_force()

print "Especificar Archivo '.npy' con Indice de Dip"
dipfile=tkFileDialog.askopenfilename(parent=root,title="Archivo de Dip")

print "Especificar Archivo '.npy' con Datos"
datafile=tkFileDialog.askopenfilename(parent=root,title="Archivo de Datos")

root.destroy()

Datos=np.load(datafile)
Dip=np.load(dipfile)

Datos,Dip=shuffle(Datos,Dip)
Datos=Datos[:cantidad]
Dip=Dip[:cantidad]

w0,w1,w2=NN(Datos,Dip,iteraciones+1,100,10,alfas)


root = Tk()
root.withdraw()
root.overrideredirect(True)
root.geometry('0x0+0+0')
root.deiconify()
root.lift()
root.focus_force()
fw0=tkFileDialog.asksaveasfilename(parent=root,title="Matriz W0")
fw1=tkFileDialog.asksaveasfilename(parent=root,title="Matriz W1")
fw2=tkFileDialog.asksaveasfilename(parent=root,title="Matriz W2")
root.destroy()
```

```
np.save(fw0,w0)
np.save(fw1,w1)
np.save(fw2,w2)
```

H. Código Python para Comparar Predicciones con Datos Pre-clasificados

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 05 16:21:57 2017

@author: Daniel Zanelli
"""



import matplotlib.pyplot as plt
import numpy as np
from Tkinter import Tk
import tkFileDialog

def sigmoid(x):
    output = 1/(1+np.exp(-x))
    return output

def NNpredict(W0,W1,W2,X):
    L0 = np.append(X,[1])
    L1 = np.append(sigmoid(np.dot(L0,W0)),[1])
    L2 = np.append(sigmoid(np.dot(L1,W1)),[1])
    L3 = sigmoid(np.dot(L2,W2))
    return L3



root = Tk()
root.withdraw()
root.overrideredirect(True)
root.geometry('0x0+0+0')
root.deiconify()
root.lift()
root.focus_force()

print "Especificar Archivo '.npy' con Indice de Dip"
dipfile=tkFileDialog.askopenfilename(parent=root,title="Archivo de Dip")

print "Especificar Archivo '.npy' con Datos"
datafile=tkFileDialog.askopenfilename(parent=root,title="Archivo de Datos")

print "Especificar Archivo '.npy' con W0"
w0file=tkFileDialog.askopenfilename(parent=root,title="Archivo de W0")

print "Especificar Archivo '.npy' con W1"
w1file=tkFileDialog.askopenfilename(parent=root,title="Archivo de W1")

print "Especificar Archivo '.npy' con W2"
w2file=tkFileDialog.askopenfilename(parent=root,title="Archivo de W2")

root.destroy()
```

```
Datos=np.load(datafile)
Dip=np.load(dipfile)

w0=np.load(w0file)
w1=np.load(w1file)
w2=np.load(w2file)



x=[]
y=[]
for d in Datos:
    pred=NNpredict(w0,w1,w2,d)
    x.append(pred[0])
    y.append(pred[1])


predicciones=[]
for i in xrange(len(Datos)):
    valor=round((1+x[i]-y[i])/2)
    predicciones.append(valor)
i=0
error=0
fallasdip=0
fallasnodip=0
for p in predicciones:
    if p!=Dip[i]:
        if Dip[i]==1:
            fallasdip+=1
        else:
            fallasnodip+=1
    error+=abs(p-Dip[i])
    i+=1

print "Falsos Negativos: ",fallasdip,"   Falsos Positivos: ",fallasnodip
print "El Sistema Neuronal Puede Predecir con ",100*(1-error*1.0/len(Dip)),"% de Asertividad"

contReal=0
contPred=0
real=[]
pred=[]
i=0
paso=100
for p in predicciones:
    contPred+=p
    contReal+=Dip[i]
    if i%paso==0:
        real.append(contReal)
        pred.append(contPred)
        contReal=0
        contPred=0
    i+=1
plt.figure()
```

```
plt.plot(np.array(range(len(real)))*paso, real,  color='b')
plt.plot(np.array(range(len(real)))*paso, pred,  color='r')
plt.show()
```

I. Código Python para Pronosticos de Disponibilidad y Predicciónes de Falla

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 31 22:43:53 2017

@author: Daniel Zanelli
"""


import matplotlib.pyplot as plt
import numpy as np
from Tkinter import Tk
import tkFileDialog
from scipy import signal
from scipy.optimize import curve_fit

def sigmoid(x):
    output = 1/(1+np.exp(-x))
    return output

def NNpredict(W0,W1,W2,X):
    L0 = np.append(X,[1])
    L1 = np.append(sigmoid(np.dot(L0,W0)),[1])
    L2 = np.append(sigmoid(np.dot(L1,W1)),[1])
    L3 = sigmoid(np.dot(L2,W2))
    return L3


def Filtrar(Y,n):
    b = [1.0 / n] * n
    a = 1
    Y = signal.filtfilt(b,a,Y)
    return Y

root = Tk()
root.withdraw()
root.overrideredirect(True)
root.geometry('0x0+0+0')
root.deiconify()
root.lift()
root.focus_force()

print "Especificar Archivo '.npy' con Datos"
datafile=tkFileDialog.askopenfilename(parent=root,title="Archivo de Datos")

print "Especificar Archivo '.npy' con W0"
w0file=tkFileDialog.askopenfilename(parent=root,title="Archivo de W0")

print "Especificar Archivo '.npy' con W1"
w1file=tkFileDialog.askopenfilename(parent=root,title="Archivo de W1")

print "Especificar Archivo '.npy' con W2"
w2file=tkFileDialog.askopenfilename(parent=root,title="Archivo de W2")
```

```
root.destroy()

Datos=np.load(datafile)
w0=np.load(w0file)
w1=np.load(w1file)
w2=np.load(w2file)



x=[]
y=[]
for d in Datos:
    pred=NNpredict(w0,w1,w2,d)
    x.append(pred[0])
    y.append(pred[1])



predicciones=[]
for i in xrange(len(Datos)):
    valor=round((1+x[i]-y[i])/2)
    predicciones.append(valor)
i=0
error=0
fallasdip=0
fallasnodip=0

contPred=0
pred=[]
i=0
paso=100
for p in predicciones:
    contPred+=p
    if i%paso==0:
        pred.append(contPred)
        contPred=0
    i+=1

pred=np.array(pred)
x=np.array(range(len(pred)))*paso

def Weibull(x,a,b):
    return (b/a)*((x/a)**(b-1)) *np.e**(-(1.0*x/a)**b)


offset=27
for i in xrange(3,len(pred)):
    datos_offset=pred[i-3:i]
    prom=np.mean(datos_offset)
    if prom>20:
        offset=i-10
        break
```

```
Diferencias=[]
Fallas=[]
Disparo_falla=0

for limite in x[offset+1000/paso:]:
    limite=limite/paso
    P=pred[:limite]
    xP=x[:limite]
    errormin=10**10
    for k1 in np.linspace(1,4,10):
        for k2 in np.linspace(10, 30, 10):

            xw= xP * k1/max(xP)
            yw= Filtrar(P,5) * np.mean(P)/(max(P)*k2)
            xwo=xw[offset:]
            ywo=yw[offset:]
            x0=xwo[0]
            xwo=xwo-x0

            try:
                best_vals, covar = curve_fit(Weibull, xwo, ywo, p0=[1.0,1.4])
                a,b=best_vals

                y=(Weibull(xwo,a,b)*max(P)*k2/np.mean(P))
                y=np.concatenate((zeros(offset),y))
                error=np.mean(abs(Filtrar(P,5)[len(P)/2:]-y[len(y)/2:]))

            except (RuntimeError, TypeError, NameError):
                error=10**10
                pass

            if errormin>error:
                errormin=error
                k1min=k1
                k2min=k2
                amin=a
                bmin=b
                xwomin=xwo

    largografico=100000
    d=xwomin[len(xwomin)-1]-xwomin[len(xwomin)-2]
    xw=np.linspace( 0, (largografico-paso*offset)*d/paso,(largografico-paso*offset)/paso)
    y=(Weibull(xw,amin,bmin)*max(P)*k2min/np.mean(P))
    xw=range(0,(len(xw)+offset)*paso,paso)
    y=np.concatenate((zeros(offset),y))
    i=0
    cruce=False
    fail=5
    for yi in y:
        if yi>fail:
            cruce=True
        if cruce and yi<fail:
```

```
            break
        i+=1
    if Disparo_falla!=0:
        Diferencias.append(abs(1.0*i*paso-Disparo_falla)/Disparo_falla)
    Disparo_falla=i*paso
    Fallas.append(Disparo_falla)

    print "Se predice que el Aislante fallará en el disparo ",Disparo_falla

    if limite%10==0:
        plt.figure()
        plt.bar(xP, P,label='Dip Identificado NN',width=100,color='k')
        plt.plot(xw[:(Disparo_falla+5000)/paso], y[:(Disparo_falla+5000)/paso],  color='#aa007f',lw=2.0, la
        plt.xlabel('Disparos')
        plt.ylabel('Cantidad de Dip [%]')
        plt.title('Predicciones con '+str(limite*paso)+' Disparos')
        plt.axvline(Disparo_falla, color='r',ls='--',lw=1.5, label='Corte 5% -'+str(Disparo_falla))
        plt.legend()
        plt.show()

plt.figure()
plt.plot(x[offset+1000/paso:],Fallas,color='#aa007f',marker='D',markerfacecolor='k',lw=2.0,ls='dashed')
plt.xlabel("Disparos")
plt.ylabel("Predicciones de Corte 5%")
```