UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Escuela de Ingenierías Industriales y Civiles

Proyecto Fin de Carrera:

Modelo numérico para la simulación de sistemas electroquímicos de distinta naturaleza a partir de la determinación de las propiedades de circuitos eléctricos equivalentes.

Autor:

Francisco González Pérez

Tutores:

Dr. D. Juan José Santana Rodríguez

Dr. D. Juan José Aznárez González

Titulación:

Ingeniero Industrial

Enero de 2015

En memoria de mis abuelos:

Manolo y Jomo

Mi más sincero agradecimiento a mis tutores Juan José Santana y Juan José Aznárez, por su ayuda y entera predisposición. Me siento muy agradecido por sus consejos y sus numerosas explicaciones, pero sobre todo, por haber compartido su tiempo conmigo y dejarme aprender de su dilatada experiencia. De igual forma quiero agradecer al profesor David Greiner su labor en este proyecto, y por haberme mostrado ese fantástico mundo de los Algoritmos Evolutivos.

Especial mención merecen mis padres, Fran y Pili, quienes con su esfuerzo incansable me han puesto en bandeja llegar hasta aquí. Gracias también al resto de mi familia y amigos, a mis compañeros de carrera y en especial a mi pareja Cristina, que prácticamente hemos hecho este camino juntos desde el principio y que con su apoyo y ayuda todo ha sido mucho más fácil.

ÍNDICE GENERAL

1. Introducción.

La Espectroscopía de Impedancia Electroquímica (EIS por sus siglas en inglés) es una técnica de medición de laboratorio ampliamente empleada en el modelado de sistemas electroquímicos de distinta naturaleza. Abarca tanto el análisis y caracterización de recubrimientos orgánicos aplicados sobre metales, la determinación de parámetros cinéticos y la caracterización de superconductores o de pilas, entre muchos otros.

Con el desarrollo de dicha técnica, la adquisición de datos experimentales de impedancia se ha vuelto mucho más fácil que antes. Por el contrario, la forma de analizar e interpretar los datos sigue siendo una tarea complicada y ardua. Por lo general, la interpretación de los datos de EIS incluye tres pasos. Una vez adquiridos los datos experimentales, éstos se representan en forma de diagrama de Nyquist o Bode. A partir de la forma de los diagramas obtenidos, el investigador obtiene información cualitativa sobre el comportamiento del sistema, apoyado en su experiencia en el campo de trabajo en cuestión. Para obtener datos cuantitativos, estos diagramas se ajustan a un circuito eléctrico denominado circuito eléctrico equivalente, compuestos por resistencias y elementos capacitivos, fundamentalmente. Para modelar el diagrama, en primer lugar se escoge un modelo de circuito equivalente para el sistema examinado a partir de la experiencia y conocimiento que se tenga en ese campo. En segundo lugar se determinan los valores de los parámetros de todos los componentes contenidos en el circuito mediante el uso de algunos métodos de ajuste de curvas. Finalmente, se calculan los parámetros cinéticos de la reacción del electrodo que están en concordancia con la estructura del circuito y los valores de sus componentes. Entre todos los pasos enumerados, cómo construir un modelo adecuado es el paso más crucial. Entre el software de modelado disponible para circuitos equivalentes en la actualidad, el software desarrollado por EQUIVCRT Boukamp fue el más ampliamente usado inicialmente, existiendo en la actualidad otros softwares comerciales con ligeras modificaciones, como son el ZSImpWin o el Nova. El software EQUlVCRT requiere que los usuarios tengan conocimiento y abundante experiencia profesional electroquímica, lo que limita su campo de aplicación. En cuanto a la segunda etapa, los investigadores han hecho mucho trabajo en este campo. El enfoque más popular adoptado por los investigadores para ajustar los datos de impedancia es el método de mínimos cuadrados no lineal compleja. Pero existen algunos problemas en el uso de este método, como la sensibilidad a los valores iniciales, la dificultad para calcular la expresión parcial derivado de cada parámetro para algunos circuitos complejos y la gran probabilidad de estancamiento en óptimos locales.

A la vista de todo el análisis anterior, se considera que la forma más conveniente para analizar los datos de impedancia electroquímica es construir el modelo de circuito equivalente de forma automática, así como optimizar los parámetros de los componentes de forma simultánea mediante el uso de algún método inteligente. De ahí que en este trabajo fin de título se propone el empleo de Algoritmos Evolutivos para

descubrir y optimizar la estructura de un circuito, obteniendo las constantes de todos los componentes eléctricos contenidos en el circuito.

De entre las posibles estrategias numéricas a utilizar en la elaboración de este código, se opta por los Algoritmos Evolutivos aprovechando la amplia experiencia que posee el Grupo de Computación Evolutiva y Aplicaciones (CEANI) de la ULPGC en el desarrollo de aplicaciones basadas en estas técnicas. En este sentido, el profesor David Greiner Sánchez colaborará de forma activa en la dirección de este Proyecto con los Tutores Titulares propuestos al comienzo.

1.1 Alcance:

El proyecto planteado consistirá en el desarrollo de una aplicación informática que permita modelar medidas de impedancia electroquímica, obteniendo los valores de los parámetros incluidos en el circuito así como su optimización. Para el ensayo y puesta a punto de la aplicación se emplearán datos obtenidos de la bibliografía cuyos valores serán conocidos de antemano.

1.2 Organización del documento:

Constará de un primer apartado donde se expone la física relativa al problema, el proceso corrosivo y su evaluación mediante el método de Espectroscopia de Impedancia Electroquímica, *"Electrochemical Impedance Spectroscopy"* de sus siglas en ingles "EIS". Se desarrollaran las ecuaciones de impedancias por medio del circuito equivalente correspondiente a metales y aleaciones recubiertos por polímeros.

Una segunda parte donde se define el algoritmo optimizador utilizado, así como las variantes y modificaciones al mismo que se han llevado a cabo.

En tercer lugar se expondrá la implementación del código, explicando que archivos de entrada hacen falta proporcionarle, así como los parámetros que hay que ajustar en cada tipo de algoritmo; también se explicará la estructura de los archivos de salida.

La última parte del documento constará de los ensayos realizados para validar los códigos y sus resultados, comentándolos y comparándolos con las soluciones que proporciona el programa comercial ZSimpWin.

2. Física del problema.

Para el desarrollo de este apartado se han tomado como referencia [1], [6] y [14].

2.1 Medidas de impedancia.

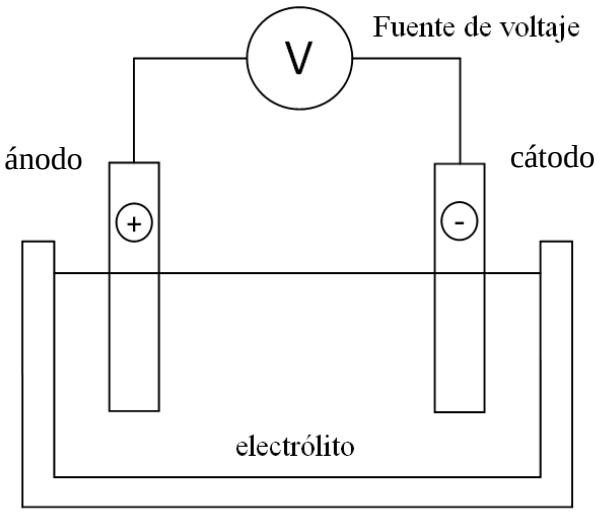

Figura 1. Celda electrelolítica.

La carga superficial de la superficie de contacto del electrodo con el electrolito queda equilibrada por una acumulación de iones en la superficie adyacente, convirtiéndose en un sistema neutro.

Debido a esta disposición de cargas, a la región próxima a la superficie del electrodo se le conoce como *capa doble*, y esta puede considerarse en su equivalente eléctrico como un condensador en paralelo con la resistencia del proceso farádico de transferencia de carga.

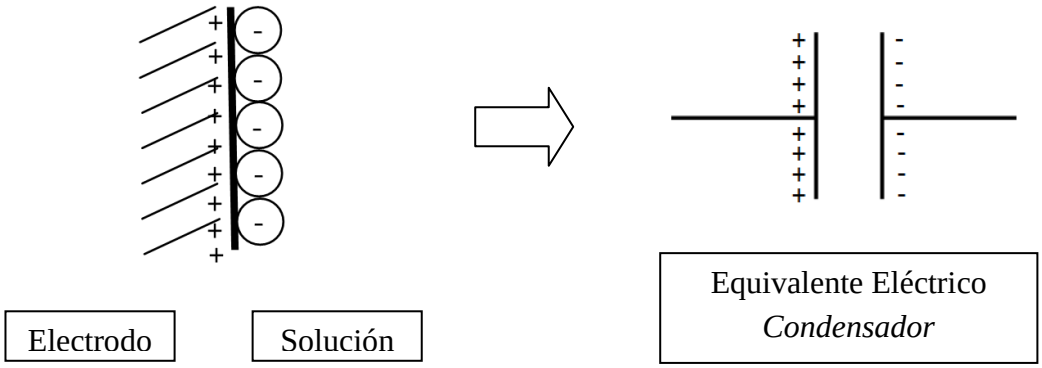

Figura 2.

El hecho de aplicar una señal senoidal a la probeta que sufre corrosión puede suministrar mucha información acerca del estado y comportamiento de la superficie de

la misma, que actúa con una determinada configuración de elementos resistentes y capacitivos.

La imposición de un potencial senoidal produce una corriente alterna resultante que tiene dos componentes; la corriente asociada al proceso de transferencia de carga, corriente farádica, que está en fase con el potencia aplicado, y la corriente que está relacionada con la carga y descarga de la *capa doble*, de naturaleza capacitiva. Por todo esto, en primera aproximación, la superficie del electrodo puede representarse como una red sencilla de resistencia y condensador en paralelo.

El vector impedancia en el campo complejo, viene dado por una combinación entre resistencias y reactancias en la que el eje horizontal de la impedancia, parte real, es la componente resistiva, y el eje vertical corresponde a la parte imaginaria, que es la componente capacitiva o inductiva (reactancia).

Las expresiones para la impedancia contienen como variable la frecuencia angular de la onda de excitación, es por ello, que tanto la magnitud como el ángulo de fase del vector impedancia variarán con dicha frecuencia angular.

En el diagrama que se muestra a continuación, cada punto del diagrama representa la magnitud y dirección del vector impedancia para una frecuencia dada.

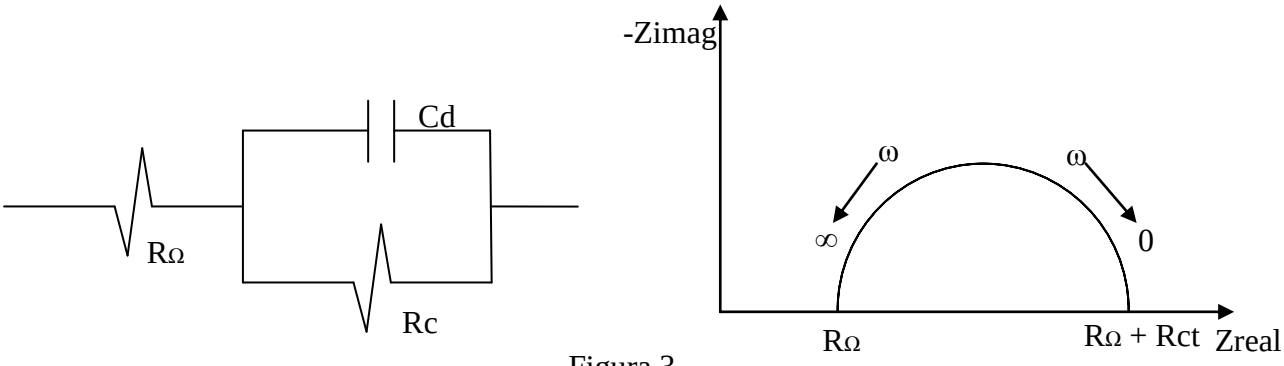

Figura 3.

En la figura se representa el circuito equivalente para una combinación de una resistencia y un condensador en paralelo, en la que además lleva incorporado una resistencia en serie.

Los condensadores dejan pasar la corriente fácilmente a altas frecuencias; cuando la frecuencia supere un determinado valor, el condensador podrá, prácticamente cortocircuitarse, dándose el caso de que la impedancia pasará a depender únicamente del efecto R_Ω, esta resistencia simula en el circuito equivalente los valores de la resistencia de la solución electrolítica. Esta circunstancia coincide con el punto de corte del lado izquierdo del semicírculo con el eje horizontal.

A medida que disminuye la frecuencia, el condensador se vuelve menos conductor, lo que motiva que la impedancia trace en función de la frecuencia angular el semicírculo de la figura. A muy bajas frecuencias, el condensador deja prácticamente de conducir, de manera que la impedancia toma el valor de la suma de las dos resistencias, que coincide con el punto de corte derecho del semicírculo con el eje horizontal.

El diámetro del semicírculo toma el valor de Rct. Cuando el sistema está bajo el control de activación, Rct coincide con el valor medio de la resistencia de polarización, y como ésta es inversamente proporcional a la velocidad de corrosión, nos permite una medida de esta velocidad.

2.2 Espectroscopía de Impedancia Electroquímica (EIS).

La técnica de Espectroscopía de Impedancia Electroquímica, es un ensayo no destructivo que permite desde el estudio de las reacciones químicas que ocurren en una celda electrolítica a la caracterización de recubrimientos orgánicos aplicados sobre metales a través de la estimulación con señales eléctricas sinusoidales, de amplitud constante y frecuencia variable. Entre sus aplicaciones se encuentra el estudio de procesos de corrosión.

La impedancia es obtenida a través de la medición de la señal de estímulo y de la respuesta del sistema. Es importante tener en cuenta el nivel de energía de la señal a aplicar, de modo que el sistema a evaluar, se mantenga en una región lineal. Es por ello que a menudo se utilizan tensiones y corrientes de estímulo de un valor muy reducido.

Para el caso de la estimulación a través de señales de tensión, la respuesta ante el estímulo será la corriente del sistema. En la figura se observa una señal de tensión aplicada a un sistema y la corriente de respuesta.

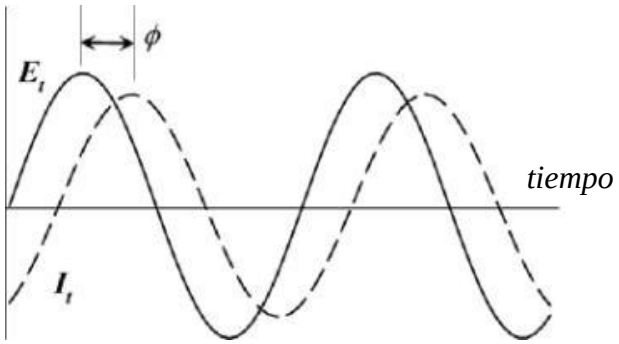

Figura 4.

La señal de excitación, expresada en función del tiempo tiene la forma:

$$V(t) = V_0 \cdot \text{sen}(\omega \cdot t) \qquad (1)$$

Donde V_0 es la amplitud de la señal y ω es la frecuencia angular de excitación expresada en rad/seg.

En un sistema lineal, la señal de respuesta I(t), tiene un cambio de fase (ϕ) y una amplitud diferente (I_0).

$$I(t) = I_0 \cdot \text{sen}(\omega \cdot t + \phi) \qquad (2)$$

Por tanto, la impedancia del sistema es calculada de la siguiente forma a partir de la ley de Ohm:

$$Z = \frac{V(t)}{I(t)} = \frac{V_0 \cdot \text{sen}(\omega \cdot t)}{I_0 \cdot \text{sen}(\omega \cdot t - \phi)} = Z_0 \cdot \frac{\text{sen}(\omega \cdot t)}{\text{sen}(\omega \cdot t - \phi)} \qquad (3)$$

Teniendo en cuenta la relación de Euler, podemos expresar la impedancia como una función compleja. La tensión se describe como:

$$V = V_0 \cdot e^{j\omega t} \qquad (4)$$

Y la respuesta de la corriente como:

$$I = I_0 \cdot e^{j\omega t - \phi} \qquad (5)$$

La impedancia entonces es representada como un número complejo:

$$Z(\omega) = \frac{V}{I} = Z_0 \cdot e^{j\phi} = Z_0 \cdot (cos\phi + j \cdot sen\phi) \qquad (6)$$

Para obtener el espectro de impedancia, en aplicaciones electroquímicas se varía la frecuencia de la señal sinusoidal dentro de un rango aproximado de 10 mHz a 10 kHz, obteniendo de esta forma el comportamiento que presenta la impedancia del sistema.

Las representaciones más comunes del espectro de impedancia de un circuito en el dominio de la frecuencia o en el plano complejo son:

Diagrama de Nyquist: Es una gráfica que consiste en presentar la parte real de la impedancia en el eje *X* y la parte imaginaria en el eje *Y* del plano complejo, en el caso electroquímico suele presentarse la parte imaginaria negativa en el eje *Y* del plano complejo. Lo anterior es debido a que, en estricto rigor matemático, en la mayoría de los sistemas electroquímicos la parte imaginaria de la impedancia tiene valores negativos.

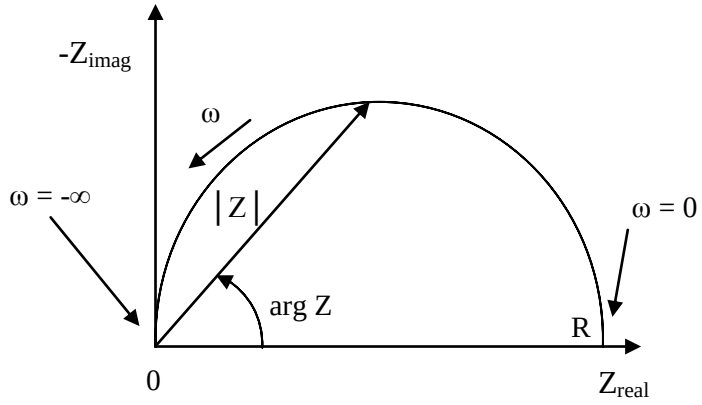

Figura 5. Diagrama de Nyquist.

Diagramas de Bode: Consiste en presentar la fase o magnitud de la impedancia en función de la frecuencia.

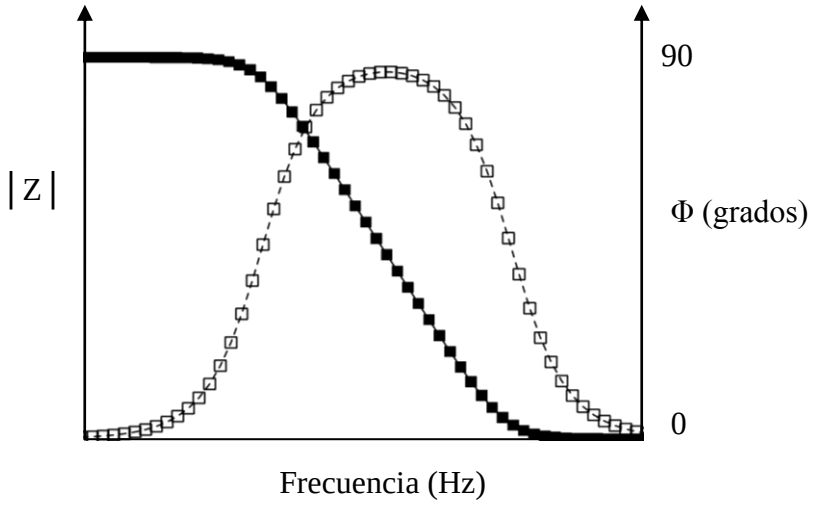

Figura 6. Diagramas de Bode.

Bajo ciertas circunstancias, es posible aplicar una señal pequeña de corriente y medir la respuesta en potencial del sistema. Así, el equipo electrónico usado procesa las mediciones de potencial - tiempo y corriente – tiempo.

En el caso de los estudios de corrosión que utilizan la técnica de EIS, los espectros de impedancia obtenidos suelen ser analizados mediante circuitos eléctricos equivalentes, compuestos por componentes tales como resistencias (R), capacitancias (C), inductancias (L), elementos de fase constante (CPE, "Constant Phase Element") entre otros.

El número de circuitos equivalentes que pueden ajustarse al comportamiento de una celda de corrosión es prácticamente infinito. No obstante, existe una condición esencial para la selección de un circuito equivalente; tanto los componentes del circuito, como el circuito eléctrico en sí mismo, deben tener explicación física. Esto es de particular importancia ya que usualmente pueden existir varios circuitos equivalentes que describan con la misma exactitud los datos experimentales.

2.3 Elemento de fase constante (CPE).

En sistemas reales los datos de EIS, representados en un diagrama de Nyquist, suelen mostrar una depresión del semicírculo que produce un achatamiento del mismo. Este comportamiento no se ha podido explicar totalmente y suele ser asociados a fenómenos tales como diseño de celda no adecuado, rugosidad superficial, porosidad superficial o reacciones que suceden en varios pasos.

A fin de ajustar espectros de EIS con depresión a un circuito eléctrico equivalente, suelen utilizarse "elementos de fase constante" (CPE, por sus siglas en inglés, "Constant Phase Element").

Un elemento de fase constante es, en realidad, una expresión matemática que representa varios elementos eléctricos. De manera formal, la impedancia de un CPE (Z_{CPE}) está dada por la siguiente ecuación:

$$Z_{CPE} = \frac{Z_0}{j\omega^n} \quad (7)$$

Cuando n=0, el CPE es una resistencia con $R=Z_0$. Si n=1 el CPE es un capacitor con $C=Z_0^{-1}$. La impedancia de Warburg (a altas frecuencias) es un caso especial y sucede cuando n=0,5. No obstante, ya que el origen físico de la depresión de los semicírculos de EIS no es claro, el significado del parámetro "n" tampoco se ha podido definir con certeza. Una consideración práctica es que, si el valor de n es mayor a 0,8, entonces el CPE puede ser considerado como un capacitor y por lo tanto la capacitancia puede ser estimada a partir de Z_0.

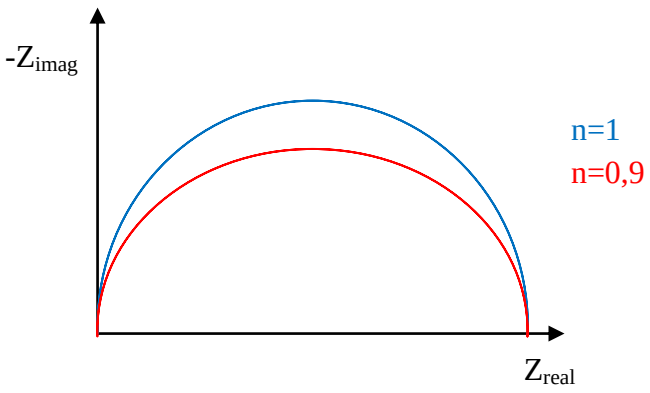

Figura 7.

2.4 Estudios de EIS en metales y aleaciones recubiertos por polímeros.

Una de las aplicaciones más útiles de la Espectrosocopía de Impedancia Electroquímica (EIS) ha sido la evaluación de las propiedades de metales recubiertos por polímeros y sus cambios durante la exposición a ambientes corrosivos. Mientras que en el pasado no era posible obtener información significante sobre el mecanismo con las técnicas tradicionales, EIS se emplea actualmente en la evaluación de una gran variedad de recubrimientos poliméricos sobre metales y aleaciones.

Se discutirán los modelos empleados para la simulación y análisis de los datos de EIS para recubrimientos poliméricos sobre metales y aleaciones, y se presentarán los datos experimentales junto con ejemplos.

Modelos para la simulación y análisis de los datos de EIS para metales recubiertos con polímeros.

Muchos de los datos de impedancia recogidos en la literatura para metales recubiertos de polímeros los cuáles han sido expuestos a medios agresivos se ajustan al modelo simple mostrado a continuación.

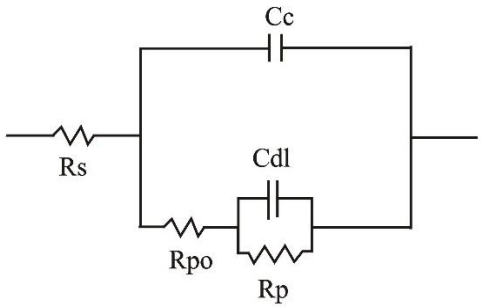

Figura 8.

R_S corresponde a la resistencia no compensada entre el electrodo de referencia y el electrodo de trabajo y C_C es la capacidad del recubrimiento polimérico el cuál viene dado por la siguiente ecuación:

$$C_C = \varepsilon \cdot \varepsilon_0 \cdot A/d \qquad (8)$$

donde ε es la constante dieléctrica del polímero, ε_0 es la constante dieléctrica del vacío ($8{,}85 \cdot 10^{-14}$ F/cm), A es el área expuesta del electrodo de trabajo y d es el espesor del recubrimiento. R_{po} es la resistencia de poro la cuál es debida a la formación de caminos iónicos conductores a través del recubrimiento. R_p es la resistencia de polarización del área en la interfase polímero/recubrimiento en la cuál tiene lugar la corrosión y C_{dl} es la capacitancia correspondiente. Ocasionalmente se ha sugerido adicionar otros elementos impedantes al circuito equivalente de la figura 8.

Una evaluación de los datos experimentales de EIS publicados hasta ahora hace frecuentemente su interpretación por la tendencia a presentar esos datos en el plano complejo lineal. Ya que la impedancia para metales recubiertos cambia en varios órdenes de magnitud, entre el valor de R_S para frecuencias muy altas y el valor de $R_S+R_{po}+R_p$ a frecuencias muy bajas, los datos de EIS se representan mejor como gráficas de Bode donde la magnitud del logaritmo del módulo de la impedancia $|Z|$ y el ángulo de fase P son representados vs. el logaritmo de la frecuencia aplicada f.

Adquisición de datos experimentales de EIS para metales y aleaciones recubiertas con polímeros.

En adición a los principios generales discutidos anteriormente, los siguientes puntos específicos deben ser tenidos en cuenta cuando realizamos medidas de EIS en metales y aleaciones con recubrimientos poliméricos.

- *Tamaño del electrodo:* Con motivo de reducir los valores experimentales de la resistencia e incrementar los valores de capacidad, el área expuesta del electrodo debe de incrementarse con el incremento del espesor del recubrimiento. Una relación de área/espesor que exceda de 10^4 cm proporciona buenos resultados.

- *Electrodo de referencia:* Es esencial utilizar un pseudo electrodo de referencia para evitar el cambio de fase que se introduce a frecuencias altas cuando se emplean electrodos de referencia tales como el de calomelano saturado (SCE). Este pesudo electrodo de referencia consiste en un SCE acoplado capacitivamente a un alambre metálico tal como el platino el cuál se extiende hasta el extremo del capilar Luggin.

- *Potencial aplicado:* Los datos de EIS para metales recubiertos con polímeros se obtienen usualmente al potencial de corrosión E_{corr}. Dado que para recubrimientos muy protectores no podemos medir un valor de E_{corr} verdadero, un potencial que esté cercano al E_{corr} para el metal desnudo se emplea frecuentemente. Para el acero en disolución de NaCl sin desairear un valor de –600 mV vs. SCE es razonable.

- *Rango de frecuencia:* El rango de frecuencias en el cuál se han de tomar los datos ha de extenderse desde el límite superior de frecuencia posible hasta unos 10 mHz. Este límite inferior depende del espesor del recubrimiento, del área expuesta y de la extensión del daño del recubrimiento. En muchos casos se toman 10 puntos espaciados igualmente en escala logarítmica por década de frecuencia. El rango superior de frecuencia está limitado generalmente al ancho de banda del potenciostato.

- *Resistencia del medidor de corriente:* Ya que la impedancia del recubrimiento polimérico cambia varios órdenes de magnitud, es extremadamente importante ajustar la resistencia del medidor de corriente R_m durante el transcurso de una medida. Este ajuste asegura que los datos de impedancia son medidos con igual exactitud a altas frecuencias, donde la impedancia es baja, y a muy bajas frecuencias, donde la impedancia es muy alta. En general, R_m debe estar en torno a un factor de 10 respecto a la impedancia a ser medida.

- *Señal AC:* Dado que un metal con recubrimiento polimérico puede ser considerado un sistema lineal (al menos si no contiene daños por corrosión considerables) es posible usar una señal AC tan grande como para metales desnudos y por eso disminuye la dispersión de los datos experimentales. Una aproximación satisfactoria involucra la aplicación de señales AC de unos 100 mV en el rango de altas frecuencias mientras que éstas han de disminuir en el rango de bajas frecuencias. Hay que tener cuidado para no tener sobrecarga de corriente.

- *Periodo de exposición:* Con el objeto de determinar la resistencia frente a la corrosión de un metal con recubrimiento polimérico frecuentemente es necesario exponerlo a un medio corrosivo tal como NaCl 0,5 N durante largos periodos de tiempo. Durante este periodo de tiempo el espectro de impedancia cambiará si se deterioran las propiedades del recubrimiento y tiene lugar la corrosión en la interfase metal/recubrimiento. Los cambios observados en los datos de EIS pueden ser utilizados para interpretaciones acerca del mecanismo. Algunas veces se efectúa un defecto pequeño en el recubrimiento lo cual nos permite un análisis de la dependencia del tiempo y la extensión de la delaminación del recubrimiento y la evaluación de la cinética de la reacción en el defecto.

<u>Análisis de los datos experimentales de EIS para metales y aleaciones recubiertas con polímeros</u>.

Los datos experimentales de EIS para metales recubiertos por polímeros pueden ser analizados con diversos programas de análisis existentes en el mercado con tal de que esos datos estén de acuerdo con el modelo de la figura 8. Los resultados de los análisis nos suministran valores numéricos para R_S, C_C, R_{po} y C_{dl}. Además, algunos parámetros experimentales los cuáles se han mostrado relacionados con daños en el recubrimiento y corrosión en la interfase metal/recubrimiento son mostrados. Estos parámetros incluyen el mínimo del ángulo de fase P_{min} y la frecuencia del punto de ruptura f_b la cuál corresponde a un ángulo de fase de $\alpha \cdot 45^o$ con $0 \leq \alpha \leq 1$.

La relación de delaminación D se calcula en base a la relación del valor experimental de C_{dl} y el valor teórico $C^0_{dl} = 20\ \mu F/cm^2$ para la capacidad de la doble

capa. Basado en el valor de D, el área delaminada en la cual se asume que tiene lugar la corrosión se calcula como

$$A_{corr} = A \times D = A \times C_{dl} / C^{0}_{dl} \qquad (9)$$

donde A es el área expuesta del electrodo de trabajo con el recubrimiento.

Estos se muestran en una tabla donde se resumen los resultados del análisis de los datos. Al final del análisis el espectro de impedancia experimental y el ajuste se superponen y se puede sacar una copia impresa.

Ajuste de los datos experimentales.

La figura 9 nos muestra un ejemplo del análisis de los datos de EIS para un metal recubierto con un polímero. Las gráficas de Bode para los datos experimentales y el ajuste de los datos pueden verse superpuestos. Los resultados de los análisis se muestran en una tabla al lado de las gráficas. Estos resultados incluyen los elementos del circuito equivalente de la figura 8 y otros parámetros los cuáles han sido calculados tal y como se describió anteriormente.

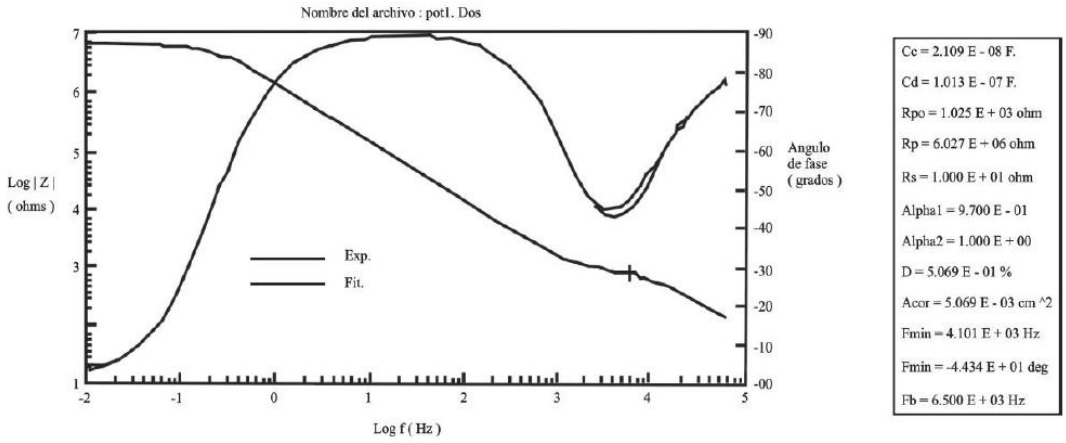

Figura 9. Análisis de los datos de EIS para un metal recubierto con un polímero.

Datos simulados vs. datos experimentales.

El software existente en el mercado permite comparar datos teóricos y experimentales. Después de introducir los valores de R_S, C_C, C_{dl}, R_{po}, R_p, α_1, α_2 y el rango de frecuencia para los datos simulados y el fichero de datos experimentales, los datos teóricos y experimentales se muestran superpuestos en gráficas Bode. El rango de frecuencia para los datos de impedancia teóricos puede ser ampliado para mostrar la dependencia con la frecuencia de la impedancia a frecuencias mucho más altas o bajas que el rango de frecuencia usado para obtener los datos experimentales.

2.5 Circuito equivalente utilizado en los códigos desarrollados y desacople de la parte real de la imaginaria.

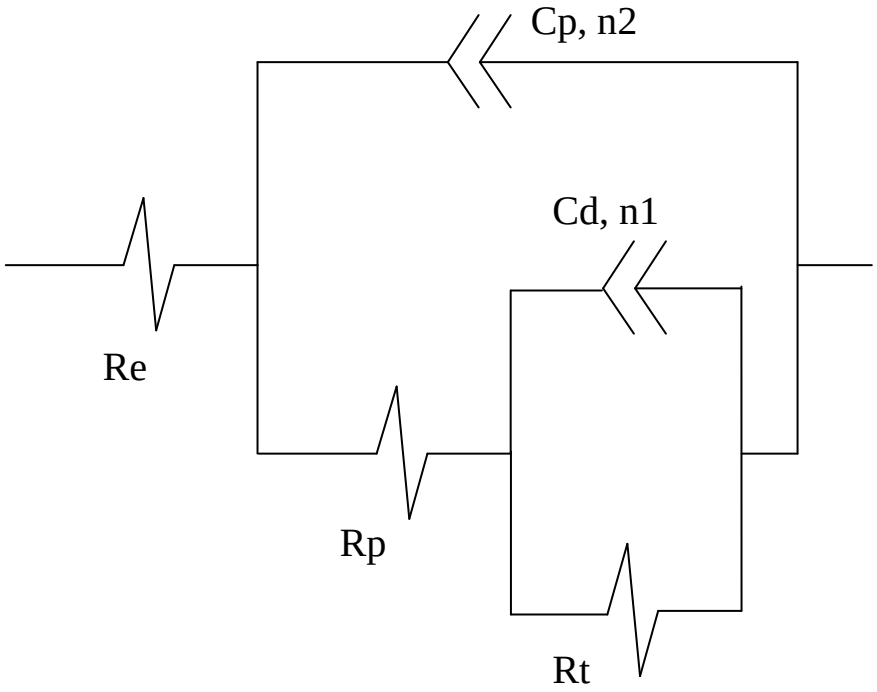

Figura 10. Circuito eléctrico equivalente.

Desarrollo de ecuaciones para desacoplar Zr y Zi en función de la frecuencia y poder crear la nube de puntos.

$$Z_{real} = f(\omega) \;;\; Z_{imag} = f(\omega) \qquad (10)$$

Según el circuito equivalente mostrado en la figura se puede calcular su impedancia mediante el cálculo de elementos en serie y paralelo:

$$Z_{eq} = R_e + \frac{1}{\frac{1}{\frac{1}{C_p \cdot (j \cdot \omega)^{n2}}} + \frac{1}{\frac{1}{\frac{1}{C_D \cdot (j \cdot \omega)^{n1}} + \frac{1}{R_T}} + R_p}}$$

$$Z_{eq} = R_e + \frac{1}{C_p \cdot (j \cdot \omega)^{n2} + \frac{1}{\frac{1}{C_D \cdot (j \cdot \omega)^{n1} + \frac{1}{R_T}} + R_p}} \qquad (11)$$

La mejor forma de abordar el desarrollo es haciéndolo por partes y aplicando la fórmula de Moivre:

Fórmula de Moivre: $(\cos(x)+j\cdot sen(x))^n=\cos(n\cdot x)+j\ sen(n\cdot x)$

$$Z_{eq} = R_e + \frac{1}{C_p\cdot(j\cdot\omega)^{n2}+\frac{1}{\frac{1}{C_D\cdot(j\cdot\omega)^{n1}+\frac{1}{R_T}}+R_p}} \qquad (12)$$

Primero abordaremos la parte de la ecuación marcada en verde, luego la azul, después la roja y por último la ecuación entera:

$$\frac{1}{C_D\cdot(j\cdot\omega)^{n1}+\frac{1}{R_T}} = \frac{1}{C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+j\cdot C_D\cdot\omega^{n1}\cdot\text{sen}(90\cdot n1)+\frac{1}{R_T}} =$$

$$= \frac{1}{C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+j\cdot[C_D\cdot\omega^{n1}\cdot\text{sen}(90\cdot n1)]+\frac{1}{R_T}} =$$

$$= \frac{1}{\left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]+j\cdot[C_D\cdot\omega^{n1}\cdot\text{sen}(90\cdot n1)]} =$$

$$= \frac{\left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]-j\cdot[C_D\cdot\omega^{n1}\cdot\text{sen}(90\cdot n1)]}{\left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]^2+[C_D\cdot\omega^{n1}\cdot\text{sen}(90\cdot n1)]^2} =$$

Tomando como K1 el denominador:

$$K1 = \left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]^2+[C_D\cdot\omega^{n1}\cdot\text{sen}(90\cdot n1)]^2$$

$$\frac{1}{C_D\cdot(j\cdot\omega)^{n1}+\frac{1}{R_T}} = \frac{\left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]}{K1}+j\cdot\frac{[C_D\cdot\omega^{n1}\cdot\text{sen}(90\cdot n1)]}{K1} =$$

Una vez obtenido esto, pasamos a descomponer en parte real y parte imaginaria lo siguiente:

$$\frac{1}{\frac{1}{C_D \cdot (j \cdot \omega)^{n1} + \frac{1}{R_T}} + R_p} = \frac{1}{\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]}{K1} + j \cdot \frac{[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1)]}{K1} + R_p} =$$

$$= \frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]}{K1} + R_p\right]^2 + j \cdot \left[\frac{\left[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1)\right]}{K1}\right]^2}{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]}{K1} + R_p\right]^2 + \left[\frac{\left[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1)\right]}{K1}\right]^2}$$

Tomando como K2 el denominador:

$$K2 = \left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]}{K1} + R_p\right]^2 + \left[\frac{\left[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1)\right]}{K1}\right]^2$$

$$\frac{1}{\frac{1}{C_D \cdot (j \cdot \omega)^{n1} + \frac{1}{R_T}} + R_p} = \frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]}{K1} + R_p\right]^2}{K2} + j \cdot \frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1)\right]}{K1}\right]^2}{K2}$$

Una vez obtenido esto, pasamos a descomponer en parte real y parte imaginaria lo siguiente:

$$C_p \cdot (j \cdot \omega)^{n2} + \frac{1}{\frac{1}{C_D \cdot (j \cdot \omega)^{n1} + \frac{1}{R_T}} + R_p} =$$

$$= C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) + j \cdot C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) + \frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]}{K1} + R_p\right]^2}{K2} + j \cdot \frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1)\right]}{K1}\right]^2}{K2} =$$

$$= \left[\frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T} \right]}{K1} + R_p \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right] + j \cdot \left[\frac{\left[\frac{[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1)]}{K1} \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right] =$$

Finalmente la impedancia total resulta:

$$Z_{eq} = R_e + \frac{1}{C_p \cdot (j \cdot \omega)^{n2} + \frac{1}{\frac{1}{C_D \cdot (j \cdot \omega)^{n1} + \frac{1}{R_T}} + R_p}} =$$

$$= R_e + \frac{1}{\left[\frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T} \right]}{K1} + R_p \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right] + j \cdot \left[\frac{\left[\frac{[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1)]}{K1} \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right]} =$$

$$= R_e + \frac{\left[\frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T} \right]}{K1} + R_p \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right] - j \cdot \left[\frac{\left[\frac{[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1)]}{K1} \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right]}{\left[\frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T} \right]}{K1} + R_p \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right]^2 + \left[\frac{\left[\frac{[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1)]}{K1} \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right]^2}$$

Tomando como K3 el denominador:

$$K3 = \left[\frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T} \right]}{K1} + R_p \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right]^2 + \left[\frac{\left[\frac{[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1)]}{K1} \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right]^2$$

Finalmente:

$$Z_{eq} = \left[R_e + \frac{\left[\frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T} \right]}{K1} + R_p \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right]}{K3} \right] - j \cdot \left[\frac{\left[\frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1) \right]}{K1} \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right]}{K3} \right] \quad (13)$$

Ahora ya tenemos la parte la parte real de la impedancia desacoplada de la parte imaginaria y ambas solo dependientes de la frecuencia angular.

Desarrollando cada parte por separado:

$$Z_{eq_{real}} = R_e + \frac{\left[\frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T} \right]}{K1} + R_p \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right]}{K3}$$

$$-Z_{eq_{imag}} = \frac{\left[\frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1) \right]}{K1} \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right]}{K3}$$

Sustituyendo K3:

$$Z_{eq_{real}} = R_e + \frac{\left[\frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T} \right]}{K1} + R_p \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right]}{\left[\frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T} \right]}{K1} + R_p \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right]^2 + \left[\frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1) \right]}{K1} \right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2) \right]^2}$$

$$-Z_{eq_{imag}} = \frac{\left[\frac{\left[\left[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1)\right]\right]^2}{K1} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2)\right]}{\left[\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]}{K1} + R_p\right]^2}{K2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2)\right]^2 + \left[\frac{\left[\left[C_D \cdot \omega^{n1} \cdot \text{sen}(90 \cdot n1)\right]\right]^2}{K1} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2)\right]^2}$$

Sustituyendo K2:

$$Z_{eq_{real}} = R_e + \frac{\left[\frac{\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]}{K1}+R_p\right]^2}{\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]}{K1}+R_p\right]^2+\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\mathrm{sen}(90\cdot n1)\right]}{K1}\right]^2}+C_p\cdot\omega^{n2}\cdot\cos(90\cdot n2)\right]}{\left[\frac{\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]}{K1}+R_p\right]^2}{\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]}{K1}+R_p\right]^2+\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\mathrm{sen}(90\cdot n1)\right]}{K1}\right]^2}+C_p\cdot\omega^{n2}\cdot\cos(90\cdot n2)\right]^2+\left[\frac{\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\mathrm{sen}(90\cdot n1)\right]}{K1}\right]^2}{\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]}{K1}+R_p\right]^2+\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\mathrm{sen}(90\cdot n1)\right]}{K1}\right]^2}+C_p\cdot\omega^{n2}\cdot\cos(90\cdot n2)\right]^2}$$

$$-Z_{eq_{imag}} = \frac{\left[\frac{\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\mathrm{sen}(90\cdot n1)\right]}{K1}\right]^2}{\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]}{K1}+R_p\right]^2+\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\mathrm{sen}(90\cdot n1)\right]}{K1}\right]^2}+C_p\cdot\omega^{n2}\cdot\cos(90\cdot n2)\right]}{\left[\frac{\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]}{K1}+R_p\right]^2}{\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]}{K1}+R_p\right]^2+\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\mathrm{sen}(90\cdot n1)\right]}{K1}\right]^2}+C_p\cdot\omega^{n2}\cdot\cos(90\cdot n2)\right]^2+\left[\frac{\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\mathrm{sen}(90\cdot n1)\right]}{K1}\right]^2}{\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\cos(90\cdot n1)+\frac{1}{R_T}\right]}{K1}+R_p\right]^2+\left[\frac{\left[C_D\cdot\omega^{n1}\cdot\mathrm{sen}(90\cdot n1)\right]}{K1}\right]^2}+C_p\cdot\omega^{n2}\cdot\cos(90\cdot n2)\right]^2}$$

Y finalmente sustituyendo K1:

$$Z_{eq_{real}} = R_e + \frac{\left[\frac{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]}{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]^2 + [C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]^2} + R_p\right]^2}{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]}{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]^2 + [C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]^2} + R_p\right]^2 + \left[\frac{[C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]}{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]^2 + [C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]^2}\right]^2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2)\right]^2 + \left[\frac{\left[\frac{[C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]}{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]^2 + [C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]^2}\right]^2}{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]}{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]^2 + [C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]^2} + R_p\right]^2 + \left[\frac{[C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]}{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]^2 + [C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]^2}\right]^2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2)\right]^2}$$

(14)

$$-Z_{eq_{imag}} = \frac{\left[\frac{\left[\frac{[C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]}{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]^2 + [C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]^2}\right]^2}{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]}{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]^2 + [C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]^2} + R_p\right]^2 + \left[\frac{[C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]}{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]^2 + [C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]^2}\right]^2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2)\right]^2 + \left[\frac{\left[\frac{[C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]}{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]^2 + [C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]^2}\right]^2}{\left[\frac{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]}{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]^2 + [C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]^2} + R_p\right]^2 + \left[\frac{[C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]}{\left[C_D \cdot \omega^{n1} \cdot \cos(90 \cdot n1) + \frac{1}{R_T}\right]^2 + [C_D \cdot \omega^{n1} \cdot \mathrm{sen}(90 \cdot n1)]^2}\right]^2} + C_p \cdot \omega^{n2} \cdot \cos(90 \cdot n2)\right]^2}$$

(15)

Al final se obtienen como resultado dos funciones, una de la parte real y otra de la parte imaginaria, totalmente desacopladas entre sí. Ahora, para unos valores dados de los parámetros del circuito equivalente se pueden calcular los puntos con las coordenadas de la parte real y la parte imaginaria para una misma frecuencia en el diagrama de Nyquist:

$$\left(Z_{eq_{real}} = f(\omega_i)\ ;\ -Z_{eq_{imag}} = f(\omega_i)\right) \qquad (16)$$

3. Algoritmo optimizador.

Para el desarrollo de este apartado se han tomado como referencia [5], [11] y [15].

3.1 Introducción.

Muchos problemas de optimización que aparecen en los ámbitos de las ingenierías son muy difíciles de solucionar por medio de técnicas tradicionales, por lo que a menudo se aplican algoritmos evolutivos, inspirados en la naturaleza, que recogen un conjunto de modelos basados en la evolución de los seres vivos.

El inicio de la utilización de las estrategias evolutivas en la solución de este tipo de problemas data del año 1960 cuando John Holland planteó la posibilidad de incorporar los mecanismos naturales de selección y supervivencia a la resolución de problemas de Inteligencia Artificial ("Adaptation in Natural and Artificial Systems"). Esta investigación fue fundamentalmente académica, siendo su realización práctica en aquella época muy difícil. La simulación de procesos de evolución natural de las especies da como resultado una técnica de optimización estocástica que posteriormente fue llamada algoritmos evolutivos, y que fueron enmarcados dentro de las técnicas no convencionales de optimización para problemas del mundo real. A partir de la creación de estas estrategias evolutivas aparecieron otras vías de investigación como son: Algoritmos Genéticos (Goldberg), Programación Genética (Koza), Programación Evolutiva (Fogel), y Estrategias de Evolución (Rechenberg/Schwefel), en donde la estrategia de evolución más conocida hoy en día son los algoritmos genéticos.

Los algoritmos genéticos constituyen una técnica poderosa de búsqueda y optimización con un comportamiento altamente paralelo, inspirado en el principio darwiniano de selección natural y reproducción genética. En este principio de selección de los individuos más aptos, tienen mayor longevidad y por tanto mayor probabilidad de reproducción. Los individuos descendientes de estos individuos tienen una mayor posibilidad de transmitir sus códigos genéticos a las próximas generaciones.

La aparición de computadores de grandes prestaciones y bajo coste a mediados de los 80 permite aplicar los algoritmos evolutivos a la resolución de ciertos problemas de ingeniería que antes eran inabordables, y a partir de entonces el desarrollo de estas técnicas ha sido continuo. Actualmente los algoritmos genéticos han cobrado gran importancia por su potencial como una técnica importante para la solución de problemas complejos, siendo aplicados constantemente en la ingeniería. Las aplicaciones de los algoritmos genéticos han sido muy conocidas en áreas como diseño de circuitos, cálculo de estrategias de mercado, reconocimiento de patrones, acústica, ingeniería aeroespacial, astronomía y astrofísica, química, juegos, programación y secuenciación de operaciones, contabilidad lineal de procesos, programación de rutas, interpolación de superficies, tecnología de grupos, facilidad en diseño y localización, transporte de materiales y muchos otros problemas que involucran de alguna manera procesos de optimización.

3.2 Algoritmo de Evolución Diferencial básico (DE).

El algoritmo de Evolución Diferencial (DE, de sus siglas en inglés *"Differential Evolution"*) fue desarrollado por Rainer Storn y Kenneth V. Price sobre 1995. Como muchos de los nuevos algoritmos optimizadores, DE fue motivado por problemas del mundo real. Hubo una rápida y fuerte entrada de los DE en el mundo de los Algoritmos Evolutivos terminando como una de las mejores entradas en el primer y segundo Certamen Internacional sobre Evolución Computacional. Las primeras publicaciones sobre DE fueron en actas de conferencias de 1996, y la primera publicación en una revista fue un año después. Sin embargo, la primera publicación que fue muy leída estaba en una revista de poco impacto. DE es un algoritmo evolutivo único porque no está motivado biológicamente.

DE es un algoritmo basado en la población, que está diseñado para optimizar funciones en un dominio continuo n-dimensional. Cada individuo de la población es un vector n-dimensional que representa una solución candidata al problema. DE se basa en la idea de tomar el vector diferencia entre dos individuos, y se añade una versión escalada del vector diferencia a un tercer individuo para crear una nueva solución candidata; este proceso es mostrado en la siguiente figura:

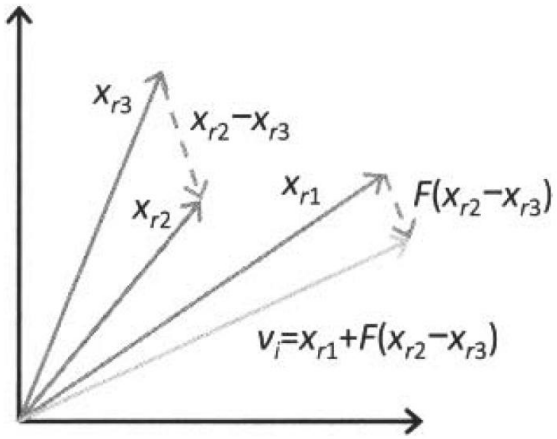

Figura 11.

La figura representa DE en un espacio de búsqueda bidimensional. Dos individuos, x_{r2} y x_{r3}, son aleatoriamente elegidos con la condición de que $r2 \neq r3$. Una versión escalada de la diferencia entre esos dos individuos se añade a un tercero también elegido aleatoriamente x_{r1} donde $r1 \notin \{r2, r3\}$. El resultado de hacer esto es un vector mutante v_i que podría ser aceptado dentro de la población como un nuevo candidato para la solución.

Después de que este vector mutante v_i se haya creado, se combina (es decir, es cruzado) con otro individuo de DE "x_i", donde $i \notin \{r1, r2, r3\}$, para crear el vector de prueba u_i. El cruce es implementado de la siguiente forma:

$$u_{ij}\begin{cases} v_{ij} & if\ (r_{cj} < CR)\ o\ (j = J_r) \\ x_{ij} & para\ lo\ demás \end{cases} \quad (17)$$

para j ∈ [1, n], donde n es la dimensión del problema y también la dimensión de u_i, v_i, y x_i; u_{ij} es el j-ésimo componente de u_i, v_{ij} es el j-ésimo componente de v_i, x_{ij} es el j-ésimo componente de x_i; r_{cj} es un número aleatorio tomado de una distribución uniforme [0, 1]; CR es el ratio de cruce (del inglés "*crossover rate*") ∈ [0, 1]; y J_r es un entero aleatorio tomado de una distribución uniforme [1, n]. Podemos ver que el vector de prueba u_i es una combinación componente a componente de un individuo de DE ya existente "x_i" y el vector mutante "v_i". El propósito de J_r es garantizar que u_i no sea un clon de x_i. El ratio de cruce CR controla cuanta información de u_i proviene de v_i y cuanta de x_i.

Después de que se hayan creado NP vectores de prueba u_i como se describió antes, donde NP es la dimensión de la población, los vectores u_i y x_i son comparados. El vector con mejor ajuste de cada pareja (u_i,x_i) se mantiene en la población para la siguiente generación de DE, y el otro es descartado. El algoritmo básico de DE (DE/rand/1/bin) para un problema n-dimensional es el que se muestra a continuación:

F = factor de escala ∈[0,4, 0,9]
CR = ratio de cruce ∈[0,1, 1]
Inicializar la población de soluciones candidatas {x_i} para i ∈[1, NP]
Mientras no (condición de parada)
    Para cada individuo x_i, i ∈[1, NP]
        r1 ← entero aleatorio ∈[1, NP] : r1 ≠ i
        r2 ← entero aleatorio ∈[1, NP] : r2 ∉{i,r1}
        r3 ← entero aleatorio ∈[1, NP] : r3 ∉{i,r1.r2}
        v_i ← x_{r1} + F·(x_{r2} – x_{r3}) (vector mutante)
        J_r ← entero aleatorio ∈[1, n]
        Para cada dimensión j ∈[1, n]
            r_{cj} ← numero aleatorio ∈[0, 1]
            Si (r_{cj} < CR) o (j = Jr) entonces
                u_{ij} ← v_{ij}
            sino
                u_{ij} ← x_{ij}
            Fin si
        Siguiente dimensión
    Siguiente individuo
    Para cada índice de población i ∈[1, NP]
        Si f(u_i) < f(x_i) entonces
            x_i ← u_i
        Fin si
    Siguiente índice de población
Siguiente generación

Como podemos ver del código, se tienen que ajustar muchos parámetros de DE. Como en otros Algoritmos Evolutivos, se tiene que elegir el tamaño de la población. Los parámetros específicos de DE incluyen el factor de escala F y el ratio de cruce (crossover rate) CR. Estos parámetros son dependientes del problema en cuestión pero típicamente toman el siguiente rango: $F \in [0{,}4, 0{,}9]$ y $CR \in [0{,}1, 1]$. El valor óptimo de F generalmente decrece según la raíz cuadrada del tamaño de la población NP. El valor óptimo de CR generalmente decrece con la posibilidad de desacople de la función objetivo.

El algoritmo previamente expuesto es usualmente referenciado como un DE clásico. También es llamado como DE/rand/1/bin porque el vector de base, x_{ri}, es aleatoriamente elegido (rand), un vector diferencia ($F \cdot (x_{r2} - x_{r3})$) es añadido a x_{r1}; y el número de elementos del vector mutante que contribuyen al vector de prueba sigue estrechamente una distribución binomial (bin). Podría seguir exactamente una distribución binomial si no fuera por la condición de *"$j = J_r$"*.

Algunas ideas sobre el algoritmo DE clásico nos indican el motivo por el cual funciona. Primero, perturbaciones de la forma ($x_{r2} - x_{r3}$) disminuye a medida que la población se acerca a la solución del problema. Segundo, las magnitudes de la perturbación son diferentes de una dimensión a otra, dependiendo de la magnitud del problema; esto es, la magnitud del p-ésimo componente de ($x_{r2} - x_{r3}$) es proporcional a como de cerca está la población de la solución del problema en relación con la p-ésima dimensión. Tercero, las medidas de la perturbación están correlacionadas entre dimensiones, que hace que la búsqueda sea eficiente incluso para problemas altamente no desacoplables. El resultado de esas características de DE es *contour matchin,* lo que significa que la población de DE se distribuye a lo largo de los contornos de la función objetivo. La población de DE tiende a adaptarse a la forma de la función objetivo.

3.3 Algoritmo de Evolución Diferencial Auto adaptativa (SaDE).

Para lograr el mejor funcionamiento en la optimización aplicando un DE convencional a un determinado problema, es común realizar el método de búsqueda de ensayo y error para encontrar la estrategia que nos da el vector de prueba más adecuado y ajustar los valores de sus correspondientes parámetros CR, F y NP. Obviamente, esto conllevara unos costes computacionales bastante elevados. Además, durante distintas partes de la evolución, diferentes estrategias para generar el vector de prueba junto con los valores de sus parámetros específicos de control pueden ser más efectivos que otros. Motivado por estas observaciones, se ha implementado el SaDE (siglas en ingles *"*Self *adaptive Differential Evolution"*, Evolución Diferencial Auto adaptativa) por el cual las estrategias de generación del vector de prueba junto con los valores de sus parámetros asociados pueden ser gradualmente auto adaptados en concordancia con previas experiencias de generar soluciones prometedoras. La idea principal detrás del algoritmo SaDE propuesto es dilucidada como sigue.

Adaptación de la estrategia para la creación del vector de prueba.

Ejecuciones de DE empleando diferentes estrategias para generar vectores de prueba normalmente funcionan de forma distinta cuando solucionamos problemas de optimización distintos. En lugar de emplear el método de ensayo y error para encontrar la estrategia más adecuada y los valores a sus correspondientes parámetros dando lugar grandes costes computacionales, se mantiene una lista de estrategias candidatas constituida por varias estrategias efectivas de generación del vector de prueba con distintas características. Durante la evolución, respecto a cada vector en la población actual, una estrategia es elegida de la lista de candidatos en concordancia con la probabilidad aprendida de su experiencia previa en generar buenas soluciones, y aplicarla para que actúe en la operación de mutación. La estrategia más exitosa que tuvo lugar en generaciones previas para la creación de soluciones prometedoras, es la que tiene más probabilidad de ser elegida en la población actual. Se ha creado una lista de candidatos bastante diversificada con cuatro estrategias bastante diferentes entre sí.

- Estrategias apoyándose en la mejor solución hasta ahora, tales como "DE/rand-to-best/1/bin", "DE/best/1/bin", y "DE/best/2/ bin", normalmente tienen una rápida convergencia y se ejecutan bien cuando tratan de resolver problemas unimodales. Algunas veces, tienen la tendencia a quedarse atrapados en óptimos locales y de este modo tienden a una convergencia prematura en caso de problemas multimodales.

- La estrategia denominada como "DE/rand/1/bin" manifiesta normalmente una convergencia lenta pero posee una mayor capacidad de exploración. Por lo tanto, es más adecuado para resolver problemas multimodales que las estrategias basadas en la mejor solución encontrada hasta el momento.

- Las estrategias basadas en dos vectores diferencia dan como resultado una mejor perturbación que las estrategias basadas en un vector diferencia. La ventaja de usar estrategias basadas en la diferencia de dos vectores radica en que la distribución estadística de la suma de todos los vectores de una sola diferencia tiene una forma de triángulo, mientras que la distribución estadística de la suma de todos los vectores de dos diferencias tiene una forma de campana que se considera generalmente como un mejor modo de perturbación.

- DE/current-to-rand/1 es una estrategia de invariancia rotacional. Su efectividad radica a la hora de aplicarla a problemas de optimización multiobjetivo.

Las cuatro estrategias de generación del vector de pruebas constituyen la lista de estrategias candidata en el algoritmo SaDE propuesto. El tipo de operador de cruce

binomial es utilizado en las tres primeras estrategias debido a su popularidad en la literatura para los casos de DE.

1. DE/rand/1/bin

$$u_{i,j}\begin{cases} x_{r1,j} + F \cdot (x_{r2,j} - x_{r3,j}), & si\ (rand[0,1) < CR)\ o\ (j = j_{rand}) \\ x_{i,j,} & para\ lo\ demás \end{cases}$$

2. DE/rand-to-best/2/bin

$$u_{i,j}\begin{cases} x_{i,j} + F \cdot (x_{best,j} - x_{i,j}) + F \cdot (x_{r1,j} - x_{r2,j}) + F \cdot (x_{r3,j} - x_{r4,j}), & si\ (rand[0,1) < CR)\ o\ (j = j_{rand}) \\ x_{i,j,} & para\ lo\ demás \end{cases}$$

3. DE/rand/2/bin

$$u_{i,j}\begin{cases} x_{r1,j} + F \cdot (x_{r2,j} - x_{r3,j}) + F \cdot (x_{r4,j} - x_{r5,j}), & si\ (rand[0,1) < CR)\ o\ (j = j_{rand}) \\ x_{i,j,} & para\ lo\ demás \end{cases}$$

4. DE/current-to-rand/1

$$U_{i,G} = X_{r1,G} + K \cdot (X_{r1,G} - X_{i,G}) + F \cdot (X_{r2,G} - X_{r3,G}).$$

En términos generales, una buena lista de candidatos debe ser restrictiva de manera que las influencias desfavorables de estrategias menos eficientes puedan ser suprimidas. Por otra parte, un conjunto de estrategias eficaces contenidas en una buena lista de candidatos debe tener características diversas, es decir, las estrategias utilizadas deben demostrar capacidades diferentes cuando se trata de un problema específico en diferentes etapas de la evolución.

En el algoritmo SaDE, con respecto a cada vector objetivo en la población actual, la estrategia de generación de vectores de prueba es seleccionada entre la lista de candidatos de acuerdo a la probabilidad aprendida de su tasa de éxito en la generación de soluciones exitosas dentro de un cierto número de generaciones previas. La estrategia seleccionada es subsecuentemente aplicada al correspondiente vector objetivo para generar un vector de pruebas. Más específicamente, en cada generación, la probabilidad de elegir cada estrategia de la lista de candidatos suma 1. Estas probabilidades se adaptan poco a poco durante la evolución de la siguiente manera. Supongamos que la probabilidad de aplicar la estrategia k-ésima en la lista de candidatos a un vector objetivo en la población actual es p_k, k = 1,2,…, K, donde K es el número total de estrategias contenidas en la lista. Las probabilidades con respecto a cada estrategia se inicializan como 1/K, es decir, todas las estrategias tienen la misma probabilidad de ser elegidas. Se utiliza el método estocástico de selección universal para seleccionar la estrategia de generación para cada vector objetivo en la población actual. En la generación G, después de evaluar todos los vectores de prueba generados, el número de vectores de prueba generados por la k-ésima estrategia que con éxito entra en la siguiente generación es almacenado como $n_{sk,G}$, mientras que el número de vectores de prueba generados por la k-ésima estrategia que fueron descartados en la siguiente generación es almacenado como $n_{fk,G}$. Introducimos *Memorias de Éxito y de Fallo* para

almacenar esos números en un número fijo de generaciones previas por medio del llamado periodo de aprendizaje (LP de sus siglas en inglés, *"Learning Period"*). Como se ilustra en la siguiente figura, en la generación G, el número de vectores de prueba generados por las diferentes estrategias que pueden entrar o dejar de entrar en la siguiente generación a través de LP generaciones previas, son almacenados en diferentes columnas de las *Memorias de Éxito y Fallo*. Una vez que las memorias se desbordan después de LP generaciones, los primeros registros almacenados en las memorias, es decir, $ns_{G\text{-}LP}$ o $nf_{G\text{-}LP}$ se eliminarán, de modo que los números calculados en la actual generación se pueden almacenar en la memoria como se muestra en la figura.

Memoria de Éxito:

Índice	Estrategia 1	Estrategia 2	…	Estrategia K
1	$ns_{1,G-LP}$	$ns_{2,G-LP}$	…	$ns_{k,G-LP}$
2	$ns_{1,G-LP+1}$	$ns_{2,G-LP+1}$	…	$ns_{k,G-LP+1}$
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
LP	$ns_{1,G-1}$	$ns_{2,G-1}$	…	$ns_{k,G-1}$

Tabla 1. Memoria de Éxito.

Memoria de Fallo:

Índice	Estrategia 1	Estrategia 2	…	Estrategia K
1	$nf_{1,G-LP}$	$nf_{2,G-LP}$	…	$nf_{k,G-LP}$
2	$nf_{1,G-LP+1}$	$nf_{2,G-LP+1}$	…	$nf_{k,G-LP+1}$
…	…	…	…	…
…	…	…	…	…
…	…	….	…	…
LP	$nf_{1,G-1}$	$nf_{2,G-1}$	…	$nf_{k,G-1}$

Tabla 2. Memoria de Fallo.

Progreso de la memoria de éxito:

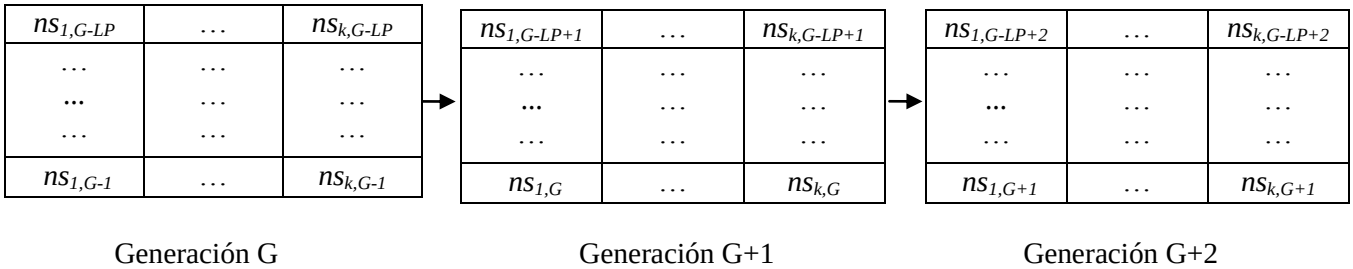

Figura 12. Progreso de la memoria de éxito.

Después de las primeras LP generaciones, las probabilidades de elegir de entre las diferentes estrategias serán actualizadas en cada generación posterior en base a las *Memorias de Éxito y de Fallo*. Por ejemplo, en la generación G, la probabilidad de elegir la k-ésima (k = 1,2,…,K) estrategia es actualizada por:

$$p_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^{K} S_{k,G}} \qquad (18)$$

donde

$$S_{k,G} = \frac{\sum_{g=G-LP}^{G-1} ns_{k,g}}{\sum_{g=G-LP}^{G-1} ns_{k,g} + \sum_{g=G-LP}^{G-1} nf_{k,g}} + \varepsilon\,, \quad (k = 1,2,\dots,K; G > LP) \qquad (19)$$

donde $S_{k,G}$ representa la tasa de éxito de los vectores de prueba generados por la estrategia k-ésima y entrar con éxito la próxima generación dentro de las generaciones anteriores de LP con respecto a la generación G. El pequeño valor constante ε = 0,01 se utiliza para evitar las posibles tasas de éxito nulas. Para asegurarse de que las probabilidades de elegir estrategias siempre suman 1, se divide $S_{k,G}$ por $\sum_{k=1}^{K} S_{k,G}$ para calcular $p_{k,G}$. Obviamente, cuanto mayor sea la tasa de éxito de la estrategia k-ésima dentro de las LP generaciones anteriores, mayor es la probabilidad de aplicarla para generar los vectores de prueba en la generación actual.

Adaptación de los parámetros de control.

En el algoritmo clásico de DE, la elección de valores numéricos para los tres parámetros de control F, CR y NP dependen en gran medida del problema en consideración. En el algoritmo SaDE, se deja NP como parámetro especificado por el usuario porque depende en gran medida de la complejidad del problema dado. De hecho, el tamaño de la población no necesita ser afinado y solo unos pocos valores típicos pueden ser evaluados de acuerdo a la complejidad estimada del problema dado. Entre los otros dos parámetros, CR normalmente es más sensible a aquellos problemas con diferentes características, por ejemplo, la unimodalidad y la multimodalidad, mientras que F está estrechamente relacionado con la velocidad de convergencia. Para este problema el parámetro F es aproximado por una distribución normal con valor medio de 0,5 y desviación típica de 0,3, N(0,5, 0,3). Un conjunto de valores de F se muestrean al azar desde tal distribución normal y se aplica a cada vector objetivo en la población actual. Es fácil comprobar que los valores de F deben caer en el rango [-0,4, 1,4] con una probabilidad de 0,997. De este modo, se intenta mantener tanto la explotación (con pequeños valores de F) y la exploración (con grandes valores de F) durante todo el proceso evolutivo. El parámetro de control K en la estrategia "DE/current-to-rand/1" es generado aleatoriamente dentro del rango [0, 1] con el fin de eliminar un parámetro adicional.

La elección adecuada de CR puede provocar un exitoso rendimiento optimizado mientras que una elección equivocada podría deteriorar su rendimiento. De hecho,

buenos valores de CR caen generalmente en un rango pequeño para un problema dado, con la que el algoritmo puede ejecutarse consistentemente bien. Por lo tanto, se considera ajustar gradualmente el rango de valores de CR para un problema dado de acuerdo a los valores de CR anteriores que han generado los vectores de prueba que entran con éxito en la siguiente generación. Específicamente en este problema CR obedece a una distribución normal con media CRm y desviación típica de Std=0,1, denotado por N(CRm, Std) donde CRm es inicializado con 0,5. El Std se debe establecer con un valor pequeño para garantizar que la mayoría de los valores de CR generados por N(CRm, Std) estén entre [0, 1], incluso cuando CRm está cerca de 0 o 1. Por lo tanto, el valor de Std se establece como 0,1. Pequeños cambios en el parámetro Std de la distribución Gaussiana no influyen en el comportamiento de SaDE significativamente.

Es razonable adaptar el valor de CRm respecto a cada estrategia de generación de vectores de prueba. Sin pérdida de generalidad, con respecto a la estrategia de orden k, el valor de CRm_k se inicializa con el valor de 0,5. Un conjunto de valores de CR son generados al azar de acuerdo con $N(CRm_k, 0,1)$ y luego se aplica a los vectores objetivo al que se asigna la estrategia k-ésima. Para adaptar el ratio de cruce CR, se establecen memorias denominadas $CRmemory_k$ para almacenar aquellos valores de CR respecto a la estrategia k-ésima que han generado los vectores de prueba que han entrado con éxito en la próxima generación dentro de las LP generaciones anteriores. Específicamente, durante las primeras LP generaciones, los valores de CR con respecto a la k-ésima estrategia son generados por $N(CRm_k, 0,1)$. En cada generación tras estas LP generaciones el valor medio almacenado en $CRmemory_k$ se calculará para sobrescribir CRm_k. Entonces, los valores de CR pueden ser generados de acuerdo con $N(CRm_k, 0,1)$ al aplicar la estrategia k-ésima. Después de evaluar los vectores de prueba generados recientemente, los valores de CR en $CRmemory_k$ que corresponden a las generaciones anteriores serán reemplazados por lo valores prometedores de CR obtenidos en la generación actual respecto a la k-ésima estrategia.

Mediante la incorporación de la anteriormente mencionada estrategia para generar el vector de pruebas y los esquemas de adaptación de los parámetros de control en el marco de un DE convencional, se ha implementado un algoritmo SaDE. En SaDE, las estrategias de generación de los vectores de prueba así como los valores de los parámetros de control asociados son gradualmente auto adaptados por el aprendizaje de sus experiencias anteriores en la creación de soluciones exitosas. En consecuencia, se puede determinar de forma adaptativa una estrategia más adecuada junto con su ajuste de parámetros correspondiente para adaptarse a las diferentes fases del proceso de búsqueda. La descripción algorítmica de SaDE se presenta a continuación:

Paso 1. *Inicializar el número de generación G = 0, e inicializar aleatoriamente la población formada por NP individuos* $P_G = \{X_{1,G},\ldots,X_{NP,G}\}$ *con* $X_{i,G} = \{x_{i,G}^{1},\ldots, x_{i,G}^{D}\}$, $i = 1,\ldots,NP$ *uniformemente distribuida en el rango* $[X_{min}, X_{max}]$, *donde* $X_{min}=\{x_{min}^{1},\ldots,x_{min}^{D}\}$ *y* $X_{max}=\{x_{max}^{1},\ldots,x_{max}^{D}\}$. *Inicializar el valor medio de CR(*CRm_k*),la*

probabilidad de cada estrategia ($p_{k,G}$, $k = 1,\ldots,K$, K es el número de estrategias disponibles), el periodo de aprendizaje (LP)

Paso 2 *Evaluar la población*

Paso 3 *Mientras no (criterio de parada)*

> ***Paso 3.1*** *Calcular la probabilidad $p_{k,G}$ y actualizar las Memorias de Éxito y de Fallo*
> *Si G>LP*
> *Para k=1 hasta K*
> *Actualizar $p_{k,G}$ por la ecuación (18)*
> *Eliminar $ns_{k,G-LP}$ y $nf_{k,G-LP}$ de las Memorias de Éxito y de Fallo respectivamente*
> *Fin para*
> *Fin si*
>
> ***Paso 3.2*** *Asignar una estrategia de generación del vector de pruebas a cada vector objetivo $X_{i,G}$*
> */* Asignar estrategia de generación para el vector de pruebas */*
> *Usando "stochastic universal sampling" para seleccionar una estrategia k para cada vector objetivo $X_{i,G}$*
>
> */*Asignar el parámetro de control F */*
> *Para i=1 hasta NP*
> *F_i = Normrand(0,5, 0,3)*
> *Fin para*
>
> */* Asignar el parámetro de control CR */*
> *Si G >= LP*
> *Para k = 1 hasta K*
> *CRm_k = median($CRMemory_k$)*
> *Fin para*
> *Fin si*
> *Para k = 1 hasta K*
> *Para i = 1 hasta NP*
> *$CR_{k,i}$ = Normrand(CRm_k, 0,1)*
> *Mientras ($CR_{m,i} < 0$) o ($CR_{m,i} > 1$)*
> *$CR_{k,i}$ = Normrand(CRm_k, 0,1)*
> *Fin mientras*
> *Fin para*
> *Fin para*

Paso 3.3 *Generar una nueva población donde cada vector de prueba $U_{i,G}^{k}$ es generado en concordancia con su estrategia de generación k asociada del vector de prueba y los parámetros F_i y $CR_{k,i}$ en el paso 3.2.*

Paso 3.4 *Reinicializar aleatoriamente el vector de prueba $U_{i,G}^{k}$ dentro del espacio de búsqueda si alguna variable está fuera de los límites.*

Paso 3.5 *Selección:*

Para i = 1 hasta NP

Evaluar el vector de prueba $U_{i,G}^{k}$

Si $f(U_{i,G}^{k}) \leq f(X_{i,G}^{k})$

$X_{i,G+1} = U_{i,G}^{k}$, $f(X_{i,G+1}) = f(U_{i,G}^{k})$

$ns_{k,G} = ns_{k,G} + 1$

Almacenar $CR_{k,i}$ dentro de $CRMemory_k$

Si $f(U_{i,G}) \leq f(X_{best,G})$

$X_{best,G} = U_{i,G}$, $f(X_{best,G}) = f(U_{i,G})$

Fin si

Sino

$nf_{k,G} = nf_{k,G} + 1$

Fin si

Fin para

Almacenar $ns_{k,G}$ y $nf_{k,G}$ (k=1,…,K) dentro de las Memorias de Éxito y de Fallo respectivamente.

Paso 3.6 *Incrementar el contador de generación G=G+1*

Paso 4 *fin mientras*

3.4 Operador renacimiento.

El equilibrio entre exploración y explotación es el principal objetivo a alcanzar en la resolución de problemas de búsqueda. Una explotación excesiva de información de alta calidad obtenida implica una exploración peor del espacio de búsqueda, siendo la población finita. Análogamente, una exploración excesiva podría producir una pérdida de rendimiento en el proceso, acercándose a una búsqueda aleatoria. El comportamiento de la evolución de un algoritmo evolutivo puede ser tenido en cuenta desde el punto de vista de este equilibrio

Una forma de tratar con el equilibrio entre exploración y explotación en los algoritmos evolutivos es mediante dos factores principales que controlan la evolución: la presión de selección y la diversidad de la población. Ambos están inversamente relacionadas: una presión de selección alta implica una pérdida rápida de la diversidad de la población, debido a la excesiva concentración de la búsqueda evolutiva de los

mejores miembros de la población; por el contrario, el mantenimiento de la diversidad de la población puede neutralizar los efectos de una excesiva presión de selección. La mayoría de los parámetros que se utilizan para ajustar las estrategias de una búsqueda evolutiva son de hecho términos indirectos de la presión de selección de sintonización y la diversidad de la población.

Un operador que introduce diversidad en la población es la reinicialización. Este operador consiste en la creación de una nueva población de partida después del estancamiento del algoritmo evolutivo en el que se inserta el mejor individuo de la población anterior. Por lo tanto, este individuo proporciona los conocimientos obtenidos con la ejecución inicial y al mismo tiempo los nuevos individuos creados al azar contribuyen a la diversidad de la población; de esta manera, se permite una continuación más en la evolución.

Otro factor que influye en la diversidad de la población es el tamaño de la población. El tamaño de la población juega un papel importante en la diversidad de la población, y además en la tarea de exploración del algoritmo: grandes tamaños de las poblaciones están asociadas con una menor velocidad de convergencia del algoritmo, pero también con un menor estancamiento prematuro de la población. Pequeños tamaños de las poblaciones pueden dar lugar a un estancamiento prematuro, y su falta de diversidad de la población tiene que ser corregido con otros operadores que lo aumentan, como el aumento de las tasas de mutación o el operador de reinicialización.

El operador renacimiento consiste en una reinicialización y una reducción del intervalo de las variables (lo que implica una reducción del espacio de búsqueda). Cuando la población converge, se aplica y extermina todos los miembros de la población, excepto el mejor individuo. Se crea una nueva población al azar que se reduce y se centra en las variables del mejor individuo.

Los algoritmos evolutivos son herramientas globales de optimización gracias a tratar con una población de soluciones en lugar de un solo individuo, pero debido a su aleatoriedad, no se garantiza el hallazgo del óptimo global. Así, diferentes ejecuciones para resolver un problema conducirá a mejores soluciones diferentes (variación de los resultados cuando se ponen en marcha varias ejecuciones). Una alternativa para reducir esta variación, es realizar una búsqueda adicional después del estancamiento del algoritmo evolutivo. Hay autores que realizan una búsqueda local, teniendo en cuenta como punto de partida el mejor valor encontrado por el algoritmo evolutivo. Sin embargo, el operador renacimiento sustituye la búsqueda local por un evolutivo, manteniendo su característica global, en un espacio más pequeño del dominio de variables y centrado en los valores de las variables del mejor individuo.

4. Implementación.

En este capítulo se describe el programa de ordenador desarrollado y basado en los modelos algorítmicos, DE y SaDE, ya expuestos. Dicho programa, escrito en código C++, permite modelar medidas de impedancia electroquímica, obteniendo los valores de los parámetros incluidos en el circuito equivalente descrito en capítulos anteriores, así como su optimización.

Obviamente este programa puede optimizarse desde un punto de vista computacional, si bien el principal objetivo en su confección ha sido que el código resulte claro y fácil de seguir por el lector interesado. En este sentido, podrán realizarse modificaciones a la hora de incluir la posibilidad de otros circuitos equivalentes para procesos electroquímicos de distintas características.

El núcleo del código está consolidado, y contrastado su funcionamiento, en cambio, el pre-procesado y post-procesado no está trabajado, estando la interacción con el usuario de una forma poco amigable, necesitándose un procesador gráfico para la salida de datos. Todas estas mejoras en el pre-procesado y post-procesado se han dejado para trabajos futuros.

A continuación se expondrá al lector como hacer uso del programa, a través de ejemplos de ficheros de entrada y salida de datos. Primeramente se abordará el código para el DE básico y después para el SaDE. Por último, también se describirán los ficheros de entrada y salida que hacen falta para graficar la dispersión (Zr,Zi) a través de los valores de las variables resultantes de los anteriores algoritmos (valores de los parámetros del circuito equivalente que componen la solución del problema), así como el código para lograrlo.

4.1 Implementación de DE.

Este código, para su funcionamiento, requiere de un archivo de entrada con los datos experimentales denominado “datos_experimentales.txt” de la siguiente forma:

1.073742e+05 1.000022e+01 1.482240e-01

8.589935e+04 1.000034e+01 1.852796e-01

6.871948e+04 1.000054e+01 2.315987e-01

……... ……. ……

…….. ……. ……

1.119872e-05 3.093464e+02 7.022116e-03

0.0 0.0 0.0

Donde la primera columna son los valores de las frecuencias tomadas experimentalmente y la segunda y tercera columna son los valores de la parte real y parte imaginaria de las impedancias medidas a la frecuencia marcada por el método de la Espectroscopía de Impedancia Electroquímica. La última fila debe de estar compuesta por 0.0 en cada una de las tres columnas como se ha expuesto.

Por otro lado, si la ejecución del programa que se quiere realizar corresponde a un renacimiento, se requiere que exista un archivo denominado "renacimiento.txt", el cual contendrá los valores de los parámetros del mejor individuo en la anterior ejecución previa al renacimiento de la siguiente forma:

8.65061e+000 1.20693e+002 1.80884e+002 2.28655e-005 8.87901e-001 2.98804e-003 9.27072e-001

donde cada uno de los valores se corresponden con cada parámetro del circuito ordenados de la siguiente forma: Re, Rp, Rt, Cp, n2, Cd, n1.

Al final de cada ejecución, el algoritmo proporcionará dos archivos, uno de ellos, "ultima_generacion.txt", contiene todas las variables de todos los individuos en la última generación y su ajuste:

Re	Rp	Rt	Cp	n2	Cd	n1	Fit
5.17e-006	2.18e+006	3.21e+006	1.84e-009	9.00e-001	1.88e-007	4.43e-001	2.14e+005
3.54e-006	2.19e+006	3.19e+006	1.84e-009	9.00e-001	1.88e-007	4.46e-001	2.14e+005
4.87e-005	2.18e+006	3.20e+006	1.88e-009	8.98e-001	1.87e-007	4.43e-001	2.14e+005
……...	…….....	……...	……...	……...	…….	……...	……
2.83e-005	2.25e+006	3.15e+006	1.91e-009	8.96e-001	2.00e-007	4.49e-001	2.17e+005

y el otro, "mejores_individuos.txt", el número de generación y los valores de las variables con su ajuste para el mejor individuo en cada generación:

Iteración	Re	Rp	Rt	Cp	n2	Cd	n1	Fit
0	2.02338e+000	1.39486e+002	1.97534e+002	2.13935e-004	6.30573e-001	6.50044e-003	5.22538e-001	7.3806180e+004
1	2.02338e+000	1.39486e+002	1.97534e+002	2.13935e-004	6.30573e-001	6.50044e-003	5.22538e-001	7.3806180e+004
2	6.49129e+000	1.28188e+002	1.79237e+002	6.52944e-004	3.99014e-001	3.15958e-003	7.86287e-001	5.2228138e+004
…	……	……	……	……...	……	……	……	……
4999	8.66011e+000	1.20689e+002	1.80882e+002	2.29692e-005	8.87322e-001	2.98779e-003	9.27160e-001	9.0955060e+002
5000	8.66011e+000	1.20689e+002	1.80882e+002	2.29692e-005	8.87322e-001	2.98779e-003	9.27160e-001	9.0955060e+002

El código implementado en C++ para el DE básico es (los parámetros de entrada que hay que proporcionar al algoritmo se encuentran al principio del código marcados en **rojo**):

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

```
#include <string.h>
#include <time.h>

void inicializar_matriz(double X[][10], int N, double a1, double b1, double a2, double b2, double a3, double b3, double a4, double b4, double a5, double b5, double a6, double b6, double a7, double b7, int ren);    /*función que inicializa la matriz de la población*/

void mutacion(double F, double X[][10], double V[][10], int i, int N, double a1, double b1, double a2, double b2, double a3, double b3, double a4, double b4, double a5, double b5, double a6, double b6, double a7, double b7);
/* mutación de los parámetros del individuo "i" de la población */

void fitness (double X[][10], int i);                    /* calcula el fitness para un individuo de la población */

main(){

    srand (time (NULL));

    FILE *fich,*fich2,*fich3;
    fich=fopen("ultima_generacion.txt","w");
    fich2=fopen("mejores_individuos.txt","w");
    fich3=fopen("renacimiento.txt","r");
    fprintf(fich,"  Re        Rp        Rt        Cp        n2        Cd        n1        Fit\n\n");
    fprintf(fich2,"Iteración    Re        Rp        Rt        Cp        n2        Cd        n1
Fit\n\n");

    int N,i,j,Jr,cont,nit,ren,best;
    double X[1000][10],V[1000][10],U[1000][10],F,c,a,a1,b1,a2,b2,a3,b3,a4,b4,a5,b5,a6,b6,a7,b7;

    N=200;              /* población */
    F=0.5;              /* varía entre 0.4 y 0.9 */
    c=0.9;              /* varía entre 0.1 y 1 */
    nit=5000;           /* número de iteraciones */

    ren=0;              /* renacimiento sí = 1 ; renacimiento no = 0 */
    if (ren==1){
      fscanf(fich3, "%lf    %lf    %lf    %lf    %lf    %lf    %lf", &X[0][0], &X[0][1], &X[0][2], &X[0][3],
&X[0][4], &X[0][5], &X[0][6]);        /* primera fila es el mejor individuo anterior al renacimiento */
    }

    a1=0.0;             /* rango inferior de Re */
    b1=25.0;            /* rango superior de Re */
    a2=50.0;            /* rango inferior de Rp */
    b2=150.0;           /* rango superior de Rp */
    a3=150.0;           /* rango inferior de Rt */
    b3=200.0;           /* rango superior de Rt */
    a4=1e-13;           /* rango inferior de Cp */
    b4=1e-2;            /* rango superior de Cp */
    a5=0.0;             /* rango inferior de n2 */
    b5=1.0;             /* rango superior de n2 */
    a6=1e-13;           /* rango inferior de Cd */
    b6=1e-2;            /* rango superior de Cd */
    a7=0.0;             /* rango inferior de n1 */
    b7=1.0;             /* rango superior de n1 */

    inicializar_matriz(X,N,a1,b1,a2,b2,a3,b3,a4,b4,a5,b5,a6,b6,a7,b7,ren);
    cont=0;

    best=0;
    for (i=1; i<N; i++){
        if (X[i][7]<X[best][7]){
            best=i;
        }
    }

    fprintf(fich2,"  %d    %.5e    %.5e    %.5e    %.5e    %.5e    %.5e    %.7e\n\n", cont,
X[best][0], X[best][1], X[best][2], X[best][3], X[best][4], X[best][5], X[best][6], X[best][7]);
```

```
    while (cont<nit){

        for (i=0; i<N; i++){
           mutacion(F,X,V,i,N,a1,b1,a2,b2,a3,b3,a4,b4,a5,b5,a6,b6,a7,b7);          /* mutación de los parámetros del individuo "i" de la población */
           Jr=rand()%(7);
           for (j=0; j<7; j++){
              a=(double)rand()/RAND_MAX;
              if ((a<c)||(j==Jr)){
                 U[i][j]=V[i][j];
              }
              else {
                 U[i][j]=X[i][j];
              }
           }
        }

        for (i=0; i<N; i++){
           fitness (U,i);  /* calcula el fitness para un individuo de la población */
           if (U[i][7]<X[i][7]){
              for (j=0; j<8; j++){
                 X[i][j]=U[i][j];
              }
           }
        }

        cont=cont+1;

        best=0;
        for (i=1; i<N; i++){
           if (X[i][7]<X[best][7]){
              best=i;
           }
        }

        fprintf(fich2," %d %.5e %.5e %.5e %.5e %.5e %.5e %.7e\n\n", cont, X[best][0], X[best][1], X[best][2], X[best][3], X[best][4], X[best][5], X[best][6], X[best][7]);

    }

    for(i=0; i<N; i++){      /* imprime en el fichero de resultados la matriz X */
        for(j=0; j<8; j++){
           if (j<7){
              fprintf(fich,"%.2e      ", X[i][j]);
           }
           else{
              fprintf(fich,"%.2e\n\n", X[i][j]);
           }
        }
    }

    fprintf(fich,"\n\nNúmero de iteraciones: %d", cont);

    fclose(fich);
    fclose(fich2);
    fclose(fich3);

}
```

```
void inicializar_matriz(double X[][10], int N, double a1, double b1, double a2, double b2, double a3, double b3, double a4, double b4, double a5, double b5, double a6, double b6, double a7, double b7, int ren){     /*función que inicializa la matriz de la población*/
```

```
int i;

if (ren==0){

   for (i=0; i<N; i++){
      X[i][0]=a1+(b1-a1)*(double)rand()/RAND_MAX;
   }

   for (i=0; i<N; i++){
      X[i][1]=a2+(b2-a2)*(double)rand()/RAND_MAX;
   }

   for (i=0; i<N; i++){
      X[i][2]=a3+(b3-a3)*(double)rand()/RAND_MAX;
   }

   for (i=0; i<N; i++){
      X[i][3]=a4+(b4-a4)*(double)rand()/RAND_MAX;
   }

   for (i=0; i<N; i++){
      X[i][4]=a5+(b5-a5)*(double)rand()/RAND_MAX;
   }

   for (i=0; i<N; i++){
      X[i][5]=a6+(b6-a6)*(double)rand()/RAND_MAX;
   }

   for (i=0; i<N; i++){
      X[i][6]=a7+(b7-a7)*(double)rand()/RAND_MAX;
   }

}

else{

   for (i=1; i<N; i++){
      X[i][0]=a1+(b1-a1)*(double)rand()/RAND_MAX;
   }

   for (i=1; i<N; i++){
      X[i][1]=a2+(b2-a2)*(double)rand()/RAND_MAX;
   }

   for (i=1; i<N; i++){
      X[i][2]=a3+(b3-a3)*(double)rand()/RAND_MAX;
   }

   for (i=1; i<N; i++){
      X[i][3]=a4+(b4-a4)*(double)rand()/RAND_MAX;
   }

   for (i=1; i<N; i++){
      X[i][4]=a5+(b5-a5)*(double)rand()/RAND_MAX;
   }

   for (i=1; i<N; i++){
      X[i][5]=a6+(b6-a6)*(double)rand()/RAND_MAX;
   }

   for (i=1; i<N; i++){
      X[i][6]=a7+(b7-a7)*(double)rand()/RAND_MAX;
   }

}

for (i=0; i<N; i++){
```

```
        fitness (X,i);   /* calcula el fitness para un individuo de la población */
    }

}




void mutacion(double F, double X[][10], double V[][10], int i, int N, double a1, double b1, double a2, double b2, double a3, double b3, double a4, double b4, double a5, double b5, double a6, double b6, double a7, double b7){
/* mutación de los parámetros del individuo "i" de la población */

    int r1,r2,r3,j;

    for (j=0; j<7; j++){

        r1=i;       /* garantiza que entre la primera vez en el bucle while que le sigue */
        while ((r1==i)||(r2==i)||(r2==r1)||(r3==i)||(r3==r1)||(r3==r2)){
           r1=rand()%(N);
           r2=rand()%(N);
           r3=rand()%(N);
        }
        V[i][j]=X[r1][j]+F*(X[r2][j]-X[r3][j]);
    }

    for (j=0; j<7; j++){        /* recalcula algún valor si este se ha salido de los márgenes */

        if ((j==0)&&((V[i][j]<a1)||(V[i][j]>b1))){
           V[i][j]=a1+(b1-a1)*(double)rand()/RAND_MAX;
        }

        if ((j==1)&&((V[i][j]<a2)||(V[i][j]>b2))){
           V[i][j]=a2+(b2-a2)*(double)rand()/RAND_MAX;
        }

        if ((j==2)&&((V[i][j]<a3)||(V[i][j]>b3))){
           V[i][j]=a3+(b3-a3)*(double)rand()/RAND_MAX;
        }

        if ((j==3)&&((V[i][j]<a4)||(V[i][j]>b4))){
           V[i][j]=a4+(b4-a4)*(double)rand()/RAND_MAX;
        }

        if ((j==4)&&((V[i][j]<a5)||(V[i][j]>b5))){
           V[i][j]=a5+(b5-a5)*(double)rand()/RAND_MAX;
        }

        if ((j==5)&&((V[i][j]<a6)||(V[i][j]>b6))){
           V[i][j]=a6+(b6-a6)*(double)rand()/RAND_MAX;
        }

        if ((j==6)&&((V[i][j]<a7)||(V[i][j]>b7))){
           V[i][j]=a7+(b7-a7)*(double)rand()/RAND_MAX;
        }

    }

}




void fitness (double X[][10], int i){  /* calcula el fitness para un individuo de la población */

    double H1,H2,H3,H4,H5,H6,H7,H8,Zr,Zi,eZr,eZi,f,w,fit;
```

```
    FILE *fich;
    fich=fopen("datos_experimentales.txt","r");

    fit=0.0;
    fscanf(fich, "%lf\t%lf\t%lf", &f,&eZr,&eZi);    /* lectura de los valores experimentales del fichero */

    while (f!=0.0){             /* última fila del fichero ponerle "0(tabulador)0(tabulador)0" */

        w=2*M_PI*f;

        H1=(X[i][5]*pow(w,X[i][6])*cos((M_PI*X[i][6])/2))+(1/X[i][2]);
        H2=X[i][5]*pow(w,X[i][6])*sin((M_PI*X[i][6])/2);

        H3=(H1/(pow(H1,2)+pow(H2,2)))+X[i][1];
        H4=H2/(pow(H1,2)+pow(H2,2));

        H5=X[i][3]*pow(w,X[i][4])*cos((M_PI*X[i][4])/2);
        H6=X[i][3]*pow(w,X[i][4])*sin((M_PI*X[i][4])/2);

        H7=pow(H3,2)+pow(H4,2);

        H8=pow(H5+(H3/H7),2)+pow(H6+(H4/H7),2);

        Zr=((H5+(H3/H7))/H8)+X[i][0];
        Zi=(H6+(H4/H7))/H8;

        fit=fit+pow(eZr-Zr,2)+pow(eZi-Zi,2);

        fscanf(fich, "%lf\t%lf\t%lf", &f,&eZr,&eZi);       /* lectura de los valores experimentales del fichero */

    }

    X[i][7]=fit;

    fclose(fich);

}
```

4.2 <u>Implementación de SaDE.</u>

Este algoritmo necesita los mismos ficheros de entrada que en al caso anterior para el DE básico, la única diferencia radica en el fichero de salida "mejores_individuos.txt", que aparte de proporcionar los valores de cada variable para el mejor individuo de cada generación, dice el método de mutación con el que se ha creado ese individuo, su factor de escala "F" y su crossover rate "CR":

Iteración	Re	Rp	Rt	Cp	n2	Cd	n1	Fit	MetMut	F	CR
0	5.78097e+000	1.41903e+002	1.97356e+002	3.87585e-005	7.53014e-001	3.68480e-003	6.94876e-001	6.1185155e+004	0	0.00e+000	0.00e+000
1	5.78097e+000	1.41903e+002	1.97356e+002	3.87585e-005	7.53014e-001	3.68480e-003	6.94876e-001	6.1185155e+004	0	2.66e-001	4.60e-001
2	5.78097e+000	1.41903e+002	1.97356e+002	3.87585e-005	7.53014e-001	3.68480e-003	6.94876e-001	6.1185155e+004	0	8.29e-001	4.47e-001
3	1.29322e+001	1.21032e+002	1.93900e+002	3.98158e-004	5.55651e-001	4.65499e-003	5.59618e-001	5.9010055e+004	1	2.56e-001	4.55e-001
4	1.29322e+001	1.21032e+002	1.93900e+002	3.98158e-004	5.55651e-001	4.65499e-003	5.59618e-001	5.9010055e+004	1	1.06e-001	4.95e-001
..	.....	.....	.....	.....	.....	.....	.....	.....	...	.....	.....
..	.....	.....	.....	.....	.....	.....	.....	.....	...	.....	.....
5000	8.60283e+000	1.20952e+002	1.80668e+002	2.32125e-005	8.85744e-001	2.99297e-003	9.28137e-001	9.0987269e+002	1	4.39e-001	3.63e-002

El fichero de salida “ultima_generación.txt” también posee la misma estructura que para el caso anterior (DE básico).

El código implementado en C++ para el SaDE básico es (los parámetros de entrada que hay que proporcionar al algoritmo se encuentran al principio del código marcados en **rojo**):

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>

void inicializar_matriz(double X[][20], int N, double a1, double b1, double a2, double b2, double a3, double b3, double a4, double b4, double a5, double b5, double a6, double b6, double a7, double b7, int ren); /* función que inicializa la matriz de la población */

void mutacion(double X[][20], double V[][20], int i, int N, int best, int x[], double a1, double b1, double a2, double b2, double a3, double b3, double a4, double b4, double a5, double b5, double a6, double b6, double a7, double b7); /* mutación de los parámetros del individuo "i" de la población */

void fitness (double X[][20], int i);                    /* calcula el fitness para un individuo de la población */

double Normrnd (double mu, double sigma);                /* devuelve un valor aleatorio según una ley normal */

double Median(double CRM[][5], int LP, int k);           /* devuelve la mediana de una serie de valores de CR en el intervalo de LP valores previos a cont */

main(){

    srand (time (NULL));

    FILE *fich,*fich2,*fich3;
    fich=fopen("ultima_generacion.txt","w");
    fich2=fopen("mejores_individuos.txt","w");
    fich3=fopen("renacimiento.txt","r");
    fprintf(fich,"  Re        Rp        Rt        Cp        n2        Cd        n1        Fit\n\n");
    fprintf(fich2,"Iteración    Re        Rp        Rt        Cp        n2        Cd        n1        Fit    MetMut    F        CR\n\n");

    int N,i,j,Jr,cont,LP,SM[100][5],FM[100][5],k,best,n,x[1000],m1,ns,nf,SMpv[5],FMpv[5],nit,ren;
    double X[1000][20],V[1000][20],U[1000][20],a,CRmk[5],pk[5],Sk[5],sSk,facum,CRM[100][5],r,CRMp[1000][5],CRMpc, m2,iN,CRMpv[5],a1,b1,a2,b2,a3,b3,a4,b4,a5,b5,a6,b6,a7,b7;

    N=200;              /* población */
    iN=0.005;           /* 1/N */
    LP=30;              /* Learning Period */
    nit=5000;           /* número de iteraciones */

    ren=0;              /* renacimiento sí = 1 ; renacimiento no = 0 */
    if (ren==1){
      fscanf(fich3, "%lf  %lf  %lf  %lf  %lf  %lf  %lf", &X[0][0], &X[0][1], &X[0][2], &X[0][3], &X[0][4], &X[0][5], &X[0][6]);      /* primera fila es el mejor individuo anterior al renacimiento */
    }

    a1=0.0;             /* rango inferior de Re */
```

```
b1=25.0;                /* rango superior de Re */
a2=50.0;                /* rango inferior de Rp */
b2=150.0;               /* rango superior de Rp */
a3=150.0;               /* rango inferior de Rt */
b3=200.0;               /* rango superior de Rt */
a4=1e-13;               /* rango inferior de Cp */
b4=1e-2;                /* rango superior de Cp */
a5=0.0;                 /* rango inferior de n2 */
b5=1.0;                 /* rango superior de n2 */
a6=1e-13;               /* rango inferior de Cd */
b6=1e-2;                /* rango superior de Cd */
a7=0.0;                 /* rango inferior de n1 */
b7=1.0;                 /* rango superior de n1 */

for (i=0; i<4; i++){      /* inicializa valor de CRmk y de pk */
     if (i!=3){
       CRmk[i]=0.5;
     }
     else{
       CRmk[i]=0.0;
     }
     pk[i]=0.25;
}

for (i=0; i<100; i++){
   for (j=0; j<5; j++){
      SM[i][j]=0;
      FM[i][j]=0;
      CRM[i][j]=0.0;
   }
}

inicializar_matriz(X,N,a1,b1,a2,b2,a3,b3,a4,b4,a5,b5,a6,b6,a7,b7,ren);
cont=0;

best=0;
for (i=1; i<N; i++){
     if (X[i][7]<X[best][7]){
          best=i;
     }
}
fprintf(fich2," %d %.5e %.5e %.5e %.5e %.5e %.5e %.5e %.7e %d %.2e
%.2e\n\n", cont, X[best][0], X[best][1], X[best][2], X[best][3], X[best][4], X[best][5], X[best][6], X[best][7], x[best],
X[best][9], X[best][10]);

while (cont<nit){      /* criterio de parada */

     if(cont>LP-1){              /* recalculo de la probabilidad de cada estrategia cada generación despues de LP
generaciones */
          for(i=0; i<4; i++){
               ns=0;
               nf=0;
               for (j=0; j<LP; j++){
                  ns=ns+SM[j][i];
                  nf=nf+FM[j][i];
               }
               Sk[i]=(ns/(ns+nf))+0.01;
          }
```

```
    sSk=0.0;
    for(i=0; i<4; i++){
        sSk=sSk+Sk[i];
    }
    for(i=0; i<4; i++){
        pk[i]=Sk[i]/sSk;
    }
}

i=0;                    /* seleccionar una estategia para cada individuo (stochastic universal sampling) */
k=-1;
facum=0.0;
r=iN*(float)rand()/RAND_MAX;
while (i<N){
    k=k+1;
    facum=facum+pk[k];
    while (facum>r){
        x[i]=k;
        i=i+1;
        r=r+iN;
    }
}

for(i=0; i<N; i++){     /* cálculo del valor de F según una ley normal para cada individuo de la población */
    X[i][9]=Normrnd(0.5,0.3);
}

if (cont>LP-1){                 /* recalculo del valor de la mediana de CR de cada estrategia cada
generación despues de LP generaciones (a la última estrategia no le corresponde valor de CR) */
  for (k=0; k<3; k++){
    CRmk[k]=Median(CRM,LP,k);
  }
}

for (i=0; i<N; i++){            /* cálculo del valor de CR según una ley normal para cada individuo de la
población (a la última estrategia no le corresponde valor de CR) */
    if (x[i]!=3){
      m1=x[i];
      m2=CRmk[m1];
      X[i][10]=Normrnd(m2,0.1);
      while ((X[i][10]<0)||(X[i][10]>1)){
          X[i][10]=Normrnd(m2,0.1);
      }
    }
    else {
      X[i][10]=0.0;
    }
}

for (i=0; i<N; i++){            /* cálculo de los hijos */
  mutacion(X,V,i,N,best,x,a1,b1,a2,b2,a3,b3,a4,b4,a5,b5,a6,b6,a7,b7);     /* mutación de los parámetros
del individuo "i" de la población */
    if (x[i]!=3){
      Jr=rand()%(7);
      for (j=0; j<7; j++){
        a=(double)rand()/RAND_MAX;
        if ((a<X[i][10])||(j==Jr)){
            U[i][j]=V[i][j];
```

```
            }
            else {
                U[i][j]=X[i][j];
            }
          }
        }
        else{
          for (j=0; j<7; j++){
            U[i][j]=V[i][j];
          }
        }
      }

      for (i=0; i<1000; i++){        /* inicializar la matriz provisional CRMp */
        for (j=0; j<5; j++){
          CRMp[i][j]=9.9;
        }
      }

      for (i=0; i<5; i++){           /* inicializa los vectores provisionales SMpv y FMpv */
        SMpv[i]=0;
        FMpv[i]=0;
      }

      for (i=0; i<N; i++){
        fitness (U,i);             /* calcula el fitness para un individuo de la población */
        if (U[i][7]<X[i][7]){
          for (j=0; j<8; j++){
            X[i][j]=U[i][j];
          }
          m1=x[i];
          SMpv[m1]=SMpv[m1]+1;     /* suma uno a la columna de la estrategia correspondiente del vector provisional de éxito si el ajuste del hijo es mejor que el del padre */
          if (x[i]!=3){
            CRMp[i][m1]=X[i][10];  /* guarda en la columna de la estrategia correspondiente de la matriz provisional de CRMp el valor de CR si el ajuste del hijo es mejor que el del padre */
          }
        }
        else{
          m1=x[i];
          FMpv[m1]=FMpv[m1]+1;     /* suma uno a la columna de la estrategia correspondiente del vector provisional de fallo si el ajuste del hijo es peor que el del padre */
        }
      }
      for (k=0; k<3; k++){         /* calcula la media de los valores de CR entre los individuos que tuvieron exito en la evaluación del ajuste para almacenarlos en CRM */
        n=0;
        CRMpc=0.0;
        for (i=0; i<N; i++){
          if(CRMp[i][k]!=9.9){
            CRMpc=CRMpc+CRMp[i][k];
            n=n+1;
          }
        }
        if (n==0){
          CRMpv[k]=CRmk[k];        /* si ningún valor tuvo éxito, se introduce el valor anterior de la mediana para mantener el cálculo de generaciones anteriores */
        }
```

```
            else{
              CRMpv[k]=CRMpc/n;
            }
          }

          for (i=0; i<LP-1; i++){          /* rueda todos los valores una fila hacia arriba de las matrices para dejar libre la última y guardar los vectores provisionales */
            for (k=0; k<4; k++){
              SM[i][k]=SM[i+1][k];
              FM[i][k]=FM[i+1][k];
              CRM[i][k]=CRM[i+1][k];
            }
          }
          for (k=0; k<4; k++){            /* guarda los vectores provisionales en la última fila de las matrices definitivas */
              SM[LP-1][k]=SMpv[k];
              FM[LP-1][k]=FMpv[k];
              CRM[LP-1][k]=CRMpv[k];
          }

          cont=cont+1;

          best=0;
          for (i=1; i<N; i++){
            if (X[i][7]<X[best][7]){
              best=i;
            }
          }
          fprintf(fich2," %d %.5e %.5e %.5e %.5e %.5e %.5e %.5e %.7e %d %.2e %.2e\n\n", cont, X[best][0], X[best][1], X[best][2], X[best][3], X[best][4], X[best][5], X[best][6], X[best][7], x[best], X[best][9], X[best][10]);

      }

      for(i=0; i<N; i++){       /* imprime en el fichero de resultados la matriz X */
          for(j=0; j<8; j++){
            if (j<7){
              fprintf(fich,"%.2e      ", X[i][j]);
            }
            else{
              fprintf(fich,"%.2e\n\n", X[i][j]);
            }
          }
      }

      fprintf(fich,"\n\nNúmero de iteraciones: %d", cont);

      fclose(fich);
      fclose(fich2);
      fclose(fich3);

}
```

```
void inicializar_matriz(double X[][20], int N, double a1, double b1, double a2, double b2, double a3, double b3, double a4, double b4, double a5, double b5, double a6, double b6, double a7, double b7, int ren){   /*función que inicializa la matriz de la población*/

    int i;

    if (ren==0){

      for (i=0; i<N; i++){
        X[i][0]=a1+(b1-a1)*(double)rand()/RAND_MAX;
      }

      for (i=0; i<N; i++){
        X[i][1]=a2+(b2-a2)*(double)rand()/RAND_MAX;
      }

      for (i=0; i<N; i++){
        X[i][2]=a3+(b3-a3)*(double)rand()/RAND_MAX;
      }

      for (i=0; i<N; i++){
        X[i][3]=a4+(b4-a4)*(double)rand()/RAND_MAX;
      }

      for (i=0; i<N; i++){
        X[i][4]=a5+(b5-a5)*(double)rand()/RAND_MAX;
      }

      for (i=0; i<N; i++){
        X[i][5]=a6+(b6-a6)*(double)rand()/RAND_MAX;
      }

      for (i=0; i<N; i++){
        X[i][6]=a7+(b7-a7)*(double)rand()/RAND_MAX;
      }

    }

    else{

      for (i=1; i<N; i++){
        X[i][0]=a1+(b1-a1)*(double)rand()/RAND_MAX;
      }

      for (i=1; i<N; i++){
        X[i][1]=a2+(b2-a2)*(double)rand()/RAND_MAX;
      }

      for (i=1; i<N; i++){
        X[i][2]=a3+(b3-a3)*(double)rand()/RAND_MAX;
      }

      for (i=1; i<N; i++){
        X[i][3]=a4+(b4-a4)*(double)rand()/RAND_MAX;
      }

      for (i=1; i<N; i++){
        X[i][4]=a5+(b5-a5)*(double)rand()/RAND_MAX;
```

```
    }

    for (i=1; i<N; i++){
       X[i][5]=a6+(b6-a6)*(double)rand()/RAND_MAX;
    }

    for (i=1; i<N; i++){
       X[i][6]=a7+(b7-a7)*(double)rand()/RAND_MAX;
    }

  }

  for (i=0; i<N; i++){
      fitness (X,i);   /* calcula el fitness para un individuo de la población */
  }

}
```

```
void mutacion(double X[][20], double V[][20], int i, int N, int best, int x[], double a1, double b1, double a2, double b2, double a3, double b3, double a4, double b4, double a5, double b5, double a6, double b6, double a7, double b7){
/* mutación de los parámetros del individuo "i" de la población */

  int r1,r2,r3,r4,r5,j;
  double k;

  if (x[i]==0){
    for (j=0; j<7; j++){
      r1=i;      /* garantiza que entre la primera vez en el bucle while que le sigue */
      while ((r1==i)||(r2==i)||(r2==r1)||(r3==i)||(r3==r1)||(r3==r2)){
          r1=rand()%(N);
          r2=rand()%(N);
          r3=rand()%(N);
      }
      V[i][j]=X[r1][j]+(X[i][9]*(X[r2][j]-X[r3][j]));
    }
  }

  if (x[i]==1){
    for (j=0; j<7; j++){
      r1=i;      /* garantiza que entre la primera vez en el bucle while que le sigue */
      while ((r1==i)||(r2==i)||(r2==r1)||(r3==i)||(r3==r1)||(r3==r2)||(r4==i)||(r4==r1)||(r4==r2)||(r4==r3)){
          r1=rand()%(N);
          r2=rand()%(N);
          r3=rand()%(N);
          r4=rand()%(N);
      }
      V[i][j]=X[i][j]+(X[i][9]*(X[best][j]-X[i][j]))+(X[i][9]*(X[r1][j]-X[r2][j]))+(X[i][9]*(X[r3][j]-X[r4][j]));
    }
  }

  if (x[i]==2){
    for (j=0; j<7; j++){
      r1=i;      /* garantiza que entre la primera vez en el bucle while que le sigue */
```

```
        while
((r1==i)||(r2==i)||(r2==r1)||(r3==i)||(r3==r1)||(r3==r2)||(r4==i)||(r4==r1)||(r4==r2)||(r4==r3)||(r5==i)||(r5==r1)||(r5==r2
)||(r5==r3)||(r5==r4)){
              r1=rand()%(N);
              r2=rand()%(N);
              r3=rand()%(N);
              r4=rand()%(N);
              r5=rand()%(N);
          }
          V[i][j]=X[r1][j]+(X[i][9]*(X[r2][j]-X[r3][j]))+(X[i][9]*(X[r4][j]-X[r5][j]));
        }
      }

      if (x[i]==3){
        for (j=0; j<7; j++){
          r1=i;      /* garantiza que entre la primera vez en el bucle while que le sigue */
          while ((r1==i)||(r2==i)||(r2==r1)||(r3==i)||(r3==r1)||(r3==r2)){
              r1=rand()%(N);
              r2=rand()%(N);
              r3=rand()%(N);
          }
          k=(float)rand()/RAND_MAX;
          V[i][j]=X[i][j]+(k*(X[r1][j]-X[i][j]))+(X[i][9]*(X[r2][j]-X[r3][j]));
        }
      }

      for (j=0; j<7; j++){  /* recalcula algún valor si este se ha salido de los márgenes */

          if ((j==0)&&((V[i][j]<a1)||(V[i][j]>b1))){
            V[i][j]=a1+(b1-a1)*(double)rand()/RAND_MAX;
          }

          if ((j==1)&&((V[i][j]<a2)||(V[i][j]>b2))){
            V[i][j]=a2+(b2-a2)*(double)rand()/RAND_MAX;
          }

          if ((j==2)&&((V[i][j]<a3)||(V[i][j]>b3))){
            V[i][j]=a3+(b3-a3)*(double)rand()/RAND_MAX;
          }

          if ((j==3)&&((V[i][j]<a4)||(V[i][j]>b4))){
            V[i][j]=a4+(b4-a4)*(double)rand()/RAND_MAX;
          }

          if ((j==4)&&((V[i][j]<a5)||(V[i][j]>b5))){
            V[i][j]=a5+(b5-a5)*(double)rand()/RAND_MAX;
          }

          if ((j==5)&&((V[i][j]<a6)||(V[i][j]>b6))){
            V[i][j]=a6+(b6-a6)*(double)rand()/RAND_MAX;
          }

          if ((j==6)&&((V[i][j]<a7)||(V[i][j]>b7))){
            V[i][j]=a7+(b7-a7)*(double)rand()/RAND_MAX;
          }

      }
```

```
}



void fitness (double X[][20], int i){  /* calcula el fitness para un individuo de la población */

    double H1,H2,H3,H4,H5,H6,H7,H8,Zr,Zi,eZr,eZi,f,w,fit;

    FILE *fich;
    fich=fopen("datos_experimentales.txt","r");

    fit=0.0;
    fscanf(fich, "%lf\t%lf\t%lf", &f,&eZr,&eZi);    /* lectura de los valores experimentales del fichero */

    while (f!=0.0){             /* última fila del fichero ponerle "0(tabulador)0(tabulador)0" */

        w=2*M_PI*f;

        H1=(X[i][5]*pow(w,X[i][6])*cos((M_PI*X[i][6])/2))+(1/X[i][2]);
        H2=X[i][5]*pow(w,X[i][6])*sin((M_PI*X[i][6])/2);

        H3=(H1/(pow(H1,2)+pow(H2,2)))+X[i][1];
        H4=H2/(pow(H1,2)+pow(H2,2));

        H5=X[i][3]*pow(w,X[i][4])*cos((M_PI*X[i][4])/2);
        H6=X[i][3]*pow(w,X[i][4])*sin((M_PI*X[i][4])/2);

        H7=pow(H3,2)+pow(H4,2);

        H8=pow(H5+(H3/H7),2)+pow(H6+(H4/H7),2);

        Zr=((H5+(H3/H7))/H8)+X[i][0];
        Zi=(H6+(H4/H7))/H8;

        fit=fit+pow(eZr-Zr,2)+pow(eZi-Zi,2);

        fscanf(fich, "%lf\t%lf\t%lf", &f,&eZr,&eZi);       /* lectura de los valores experimentales del fichero */

    }

    X[i][7]=fit;

    fclose(fich);

}



double Normrnd (double mu, double sigma){                  /* devuelve un valor aleatorio según una ley normal */

  double U1, U2, W, mult, Y, h;

  W=0.0;
  while ((W>=1.0)||(W==0.0)){
```

```
    U1=-1+((double)rand()/RAND_MAX)*2;
    U2=-1+((double)rand()/RAND_MAX)*2;
    W=pow(U1,2)+pow(U2,2);
  }

  mult=sqrt((-2*log(W))/W);

  h=(double)rand()/RAND_MAX;
    if(h<0.5){
        Y=U1*mult;
    }
    else{
        Y=U2*mult;
    }

  return (mu+sigma*Y);
}
```

```
double Median(double CRM[][5], int LP, int k){     /* devuelve la mediana de una serie de valores de CR en el intervalo de LP valores previos a cont */

   double median, value[1000],temp;
   int i, j, min;

   for (i=0; i<LP; i++);{
      value[i]=CRM[i][k];
   }

   for (i=0; i<LP-1; ++i){
      min=i;
      for (j=i+1; j<LP; ++j){
         if (value[j]<value[min]){
            min=j;
         }
      }
      temp=value[i];
      value[i]=value[min];
      value[min]=temp;
   }

   if((LP%2)==1){
      median=value[((LP)+1)/2-1];
   }
   else{
      median=(value[(LP)/2]+value[(LP)/2-1])/2;
   }

   return median;
}
```

4.3 Implementación del algoritmo para crear la dispersión.

Lo que hace este algoritmo es tomar la solución dada por el algoritmo evolutivo de los valores de los parámetros del circuito equivalente y las frecuencias a las que se tomaron los datos experimentales, para que cada punto (Zr,Zi) calculado, se corresponda con cada punto de impedancia del ensayo electroquímico. Por tanto, este código necesita de dos ficheros de entrada, el primero denominado "frecuencias.txt" que posee las frecuencias a las cuales se tomaron datos experimentales de impedancia, de la siguiente forma:

1.073742e+05

8.589935e+04

6.871948e+04

5.497558e+04

4.398047e+04

……..

……..

1.399840e-05

1.119872e-05

0.0

La última fila debe de estar compuesta por 0.0. El segundo fichero que se necesita se denomina "resultado.txt" y posee la siguiente estructura:

8.66011e+000 1.20689e+002 1.80882e+002 2.29692e-005 8.87322e-001 2.98779e-003 9.27160e-001

donde cada uno de los valores son las soluciones dadas por el algoritmo evolutivo para cada uno de los parámetros en el siguiente orden: Re, Rp, Rt, Cp, n2, Cd, n1.

El código da un único fichero de salida denominado "ZrZi.txt" que muestra en dos columnas cada par de valores de impedancia real e impedancia imaginara que constituyen un punto en el diagrama de Nyquist para cada una de las frecuencias experimentales escritas en el archivo previamente descrito "frecuencias.txt". De esta forma, se puede comparar cada punto (Zr,Zi) medido experimentalmente con cada punto calculado por el algoritmo evolutivo porque ambos corresponden a una misma frecuencia. Necesitamos un postprocesador gráfico al que se le proporcione el archivo con los datos experimentales de impedancia y el archivo con los datos calculados de impedancia para poder comparar visualmente los resultados de la siguiente forma:

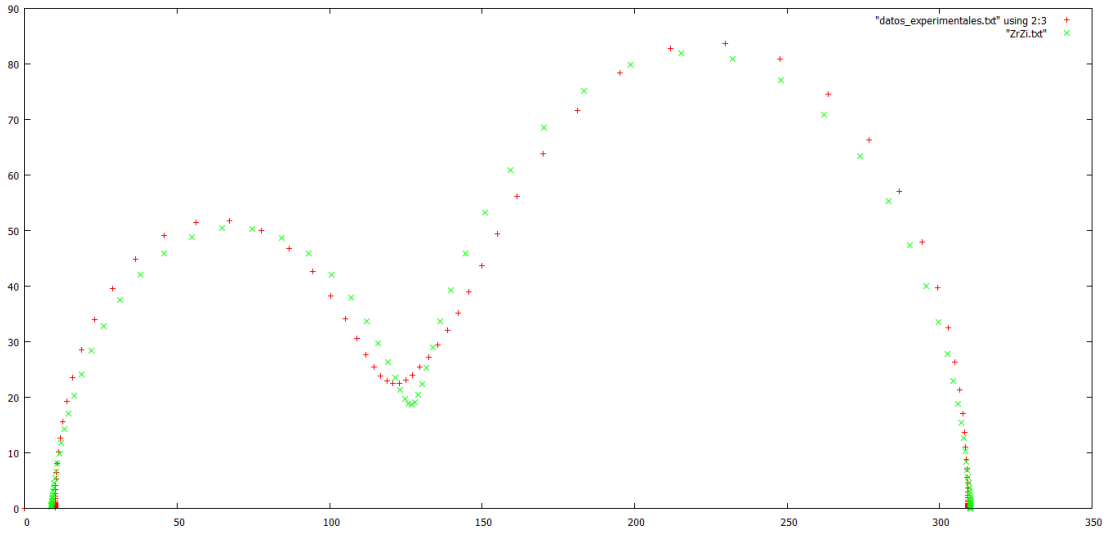

Figura 13.

El archivo “ZrZi.txt” de salida tiene el siguiente formato:

8.712331 0.287986

8.723942 0.350978

8.738183 0.427729

8.755669 0.521238

8.777173 0.635149

……. …….

……. …….

310.229078 0.017382

310.229457 0.014137

donde la primera columna es la parte real de la impedancia y la segunda es la parte imaginaria.

El código implementado en C++ es:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>

main(){

    double Re,Rp,Rt,Cd,n1,Cp,n2,H1,H2,H3,H4,H5,H6,H7,H8,Zr,Zi,f,w;

    FILE *fich;
```

```
    fich=fopen("resultado.txt","r");
    FILE *fichZrZi;
    fichZrZi=fopen("ZrZi.txt","w");
    FILE *fichfreq;
    fichfreq=fopen("frecuencias.txt","r");

    fscanf(fich, "%lf %lf %lf %lf %lf %lf %lf", &Re,&Rp,&Rt,&Cp,&n2,&Cd,&n1);
    fscanf(fichfreq, "%lf", &f);

    while (f!=0.0){

        w=2*M_PI*f;

        H1=(Cd*pow(w,n1)*cos((M_PI*n1)/2))+(1/Rt);
        H2=Cd*pow(w,n1)*sin((M_PI*n1)/2);

        H3=(H1/(pow(H1,2)+pow(H2,2)))+Rp;
        H4=H2/(pow(H1,2)+pow(H2,2));

        H5=Cp*pow(w,n2)*cos((M_PI*n2)/2);
        H6=Cp*pow(w,n2)*sin((M_PI*n2)/2);

        H7=pow(H3,2)+pow(H4,2);

        H8=pow(H5+(H3/H7),2)+pow(H6+(H4/H7),2);

        Zr=((H5+(H3/H7))/H8)+Re;
        Zi=(H6+(H4/H7))/H8;

        fprintf(fichZrZi,"%f        %f\n",Zr,Zi);

        fscanf(fichfreq, "%lf", &f);

    }

    fclose(fich);
    fclose(fichZrZi);
    fclose(fichfreq);

}
```

5. Resultados.

5.1 Estudio de sensibilidad de cada uno de los parámetros del circuito.

Antes de comenzar a realizar pruebas con los algoritmos es necesario saber cómo se manifiestan cada uno de los parámetros del circuito equivalente en el diagrama de Nyquist.

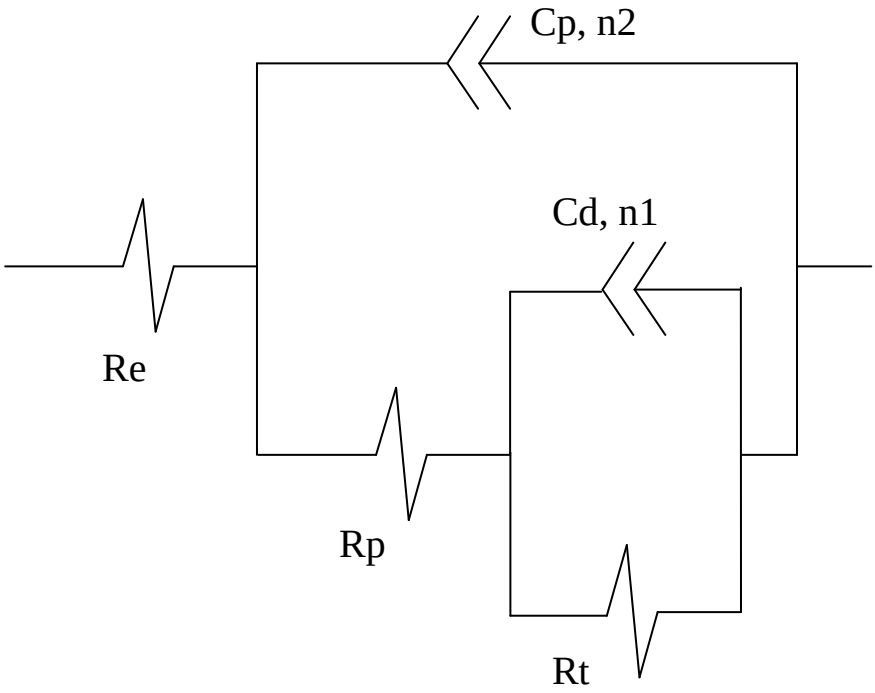

Figura 14. Circuito eléctrico equivalente.

Las resistencias Re, Rp y Rt:

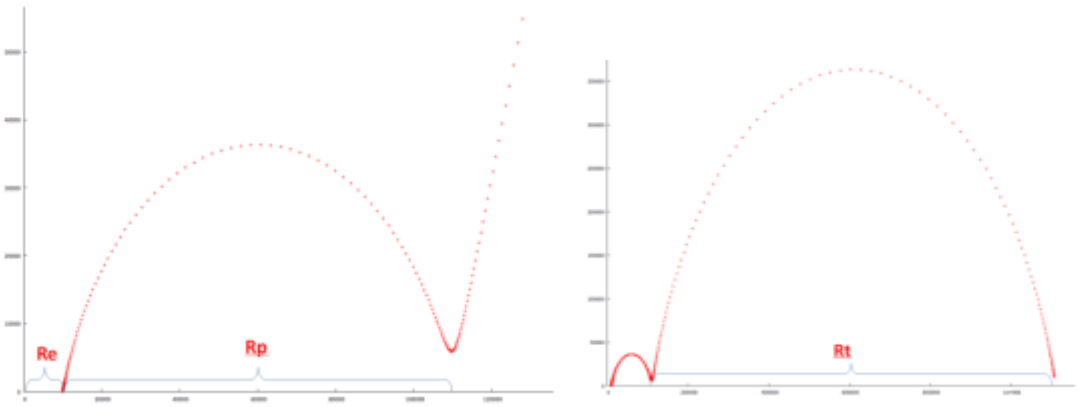

Figura 15.

Como se sabía de antemano, los dos semicírculos son debidos a los dos sistemas de resistencia y CPE en paralelo del circuito equivalente que se está tratando. Por otro

lado, los valores de las resistencias se manifiestan como los cortes de los semicírculos con el eje real de la impedancia en el diagrama de Nyquist. Por lo tanto, como se puede ver, estos valores son relativamente fáciles de sacar a priori, ya que visualmente se les puede dar una buena aproximación al algoritmo evolutivo de estas variables; no siendo tan sencillo para las restantes variables del sistema y otros casos que se mostrarán más adelante.

Los exponentes n1 y n2:

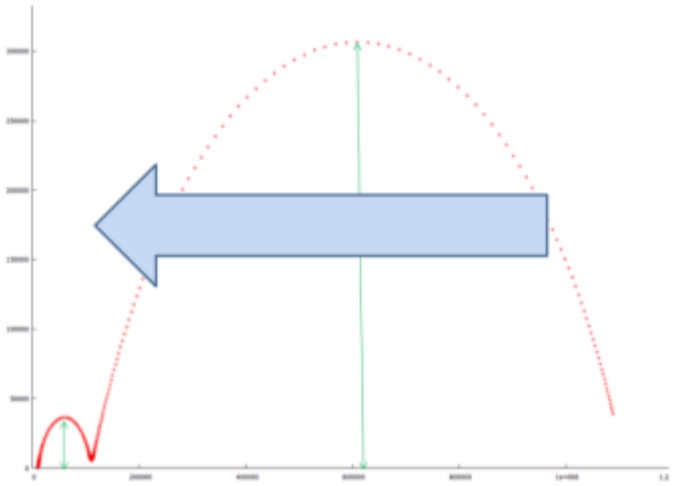

Figura 16.

El valor de estos parámetros se manifiesta en el sistema mediante el achatamiento de los semicírculos que se aprecian en la figura. Cada uno de los parámetros se corresponde con cada uno de los semicírculos. Por otro lado, estos parámetros también influyen en que la nube de puntos correspondientes al espectro de frecuencias, se desplace hacia valores más altos o más bajos de dichas frecuencias, como muestra la flecha azul.

Las capacitancias Cp y Cd:

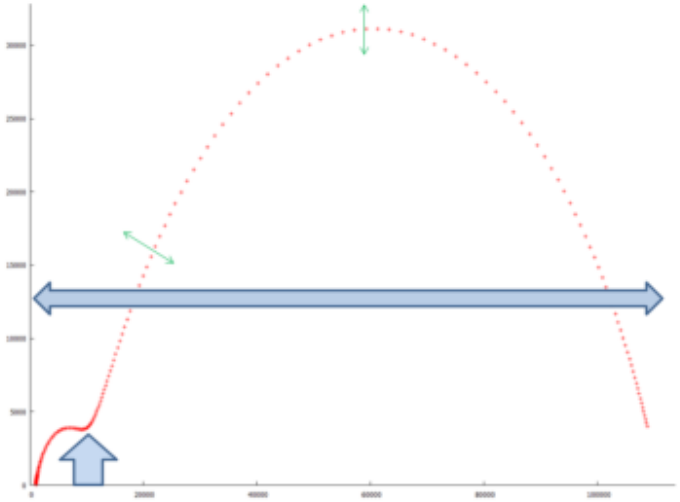

Figura 17.

El efecto que produce la variación de estas capacitancias en el diagrama de Nyquist se manifiesta en el paso del espectro desde un semicírculo a otro. También el efecto de variación de estos parámetros, al igual que en el caso anterior de los exponentes n1 y n2, hace que la nube de puntos correspondientes al espectro de frecuencias se desplace hacia valores más altos o más bajos de dichas frecuencias. El efecto que producen ambas capacitancias es el mismo, no siendo posible caracterizar ningún efecto específico para cada una de ellas.

5.2 Comparativa de DE con SaDE para la función de Ackley.

Lo que interesa saber es cuál de los dos códigos tiene un mejor funcionamiento y, por supuesto, si funcionan correctamente. Para ello, se han implementado estos códigos para comprobar que los mismos puedan llegar al mínimo de la función de Ackley y así comprobar su potencial en cuanto a velocidad de convergencia, precisión y robustez. Esta función de Ackley, se caracteriza por su infinidad de óptimos locales, lo que lo hace ampliamente aplicable a problemas de optimización para comprobar que el algoritmo sea capaz de alcanzar el óptimo absoluto.

La función de Ackley para dos dimensiones da como resultado el gráfico siguiente:

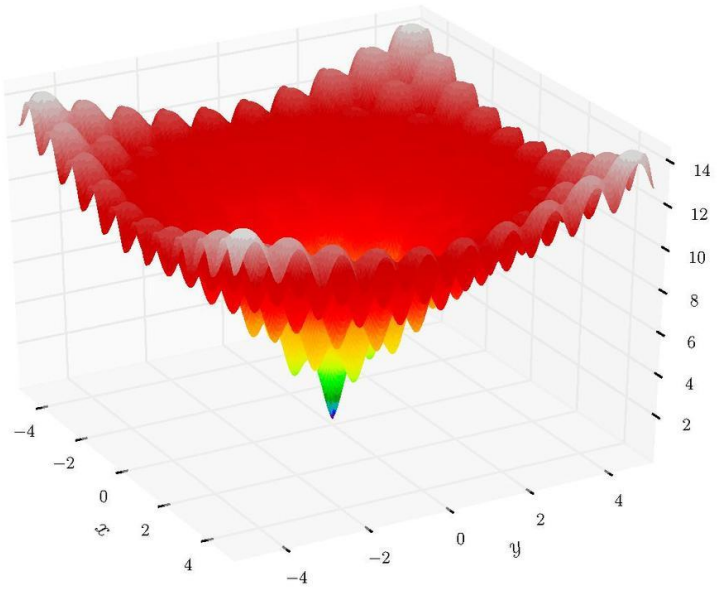

Figura 18. Función de Ackley en dos imensiones.

cuya fórmula es:

$$f(x,y) = -20 \cdot e^{-0{,}2 \cdot \sqrt{0{,}5 \cdot (x^2+y^2)}} - e^{0{,}5 \cdot (\cos(2 \cdot \pi \cdot x) + \cos(2 \cdot \pi \cdot y))} + e + 20 \qquad (20)$$

produciéndose el mínimo:

$$f(0,0) = 0 \qquad (21)$$

y un espacio de búsqueda recomendado:

$$-5 \leq x, y \leq 5 \qquad (22)$$

En este proyecto, para comprobar la potencia de los algoritmos, se va a ampliar la función de Ackley a 30 dimensiones en lugar de 2, y para un espacio de búsqueda de (-30, 30)

Para hacer la comparativa entre el DE clásico y el SaDE se utiliza para ambos el mismo número de individuos en la población inicial. Y ya que es un problema estocástico, se ha realizado la media entre 20 ejecuciones. El siguiente gráfico imprime la convergencia del ajuste del mejor individuo de la población para cada iteración de la media entre 20 ejecuciones junto con los valores de los parámetros asociados a cada método.

DE: (DE/rand/1/bin; F=0.5; CR=0.9, N=50) Sa-DE: (LP=25, N=50)

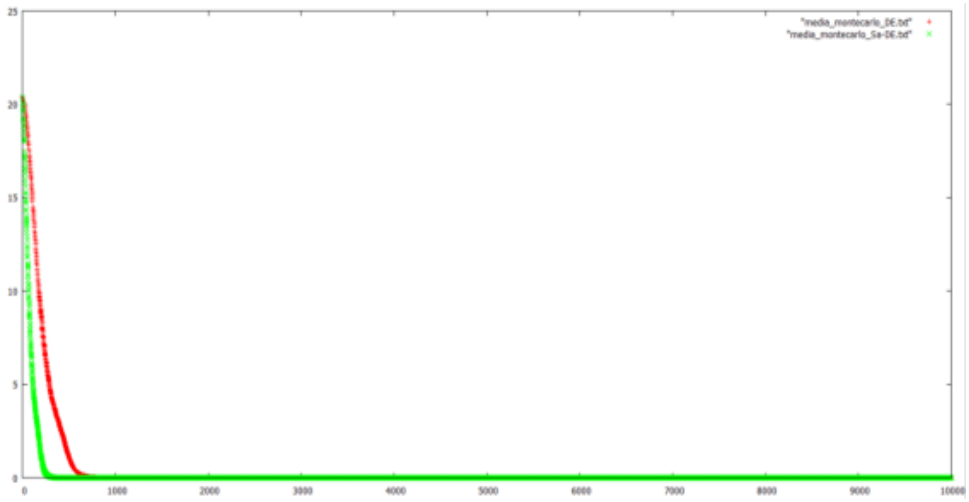

Figura 19.

Haciendo un zoom en las primeras 1000 iteraciones:

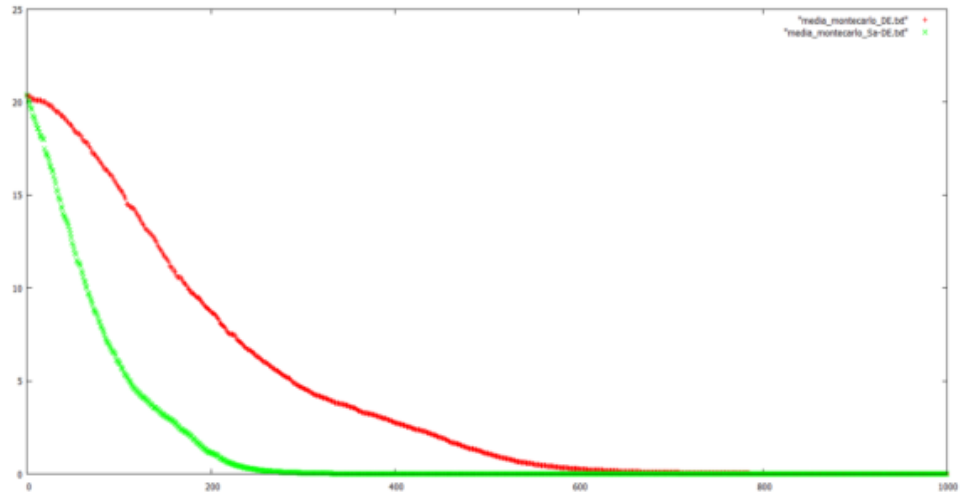

Figura 20.

De este gráfico se puede comprobar que el algoritmo SaDE (verde) converge más rápidamente hacia el óptimo correcto, lo que lo convierte en un código más potente que el DE básico (rojo).

Lo que se muestra a continuación es el mejor caso posible mediante el método de Montecarlo (20 ejecuciones):

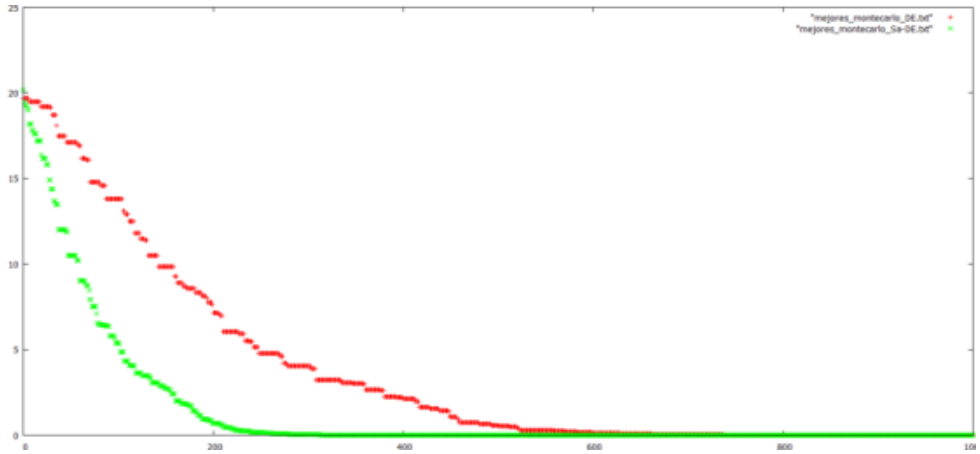

Figura 21.

Un dato importante que se puede sacar de la comparativa entre la media y el mejor caso posible por el método de Montecarlo para ambos algoritmos, DE y SaDE, es que apenas existe diferencia entre el mejor y la media en cuanto al ajuste, esto es bastante importante ya que implica que los algoritmos son bastantes consistentes en sus ejecuciones, no siendo muy importante la aleatoriedad de la población inicial.

5.3 Comparativa de DE con SaDE para los datos experimentales 1.

Antes de comenzar a realizar ensayos con datos de Espectroscopía de Impedancia Electroquímica, se tenía que debatir cuál iba a ser la función objetivo. El resultado óptimo se presentó para la minimización de la siguiente función:

$$\sum(\overline{Z_{rei}} - Z_{rei})^2 + \sum(\overline{Z_{imi}} - Z_{imi})^2 \qquad (23)$$

La expresión anterior indica que la resta por separado de la componente real y la componente imaginaria, para cada punto de impedancia medida experimentalmente y para los valores calculados por el algoritmo; ha de ser mínima. El hecho de que la expresión esté elevada al cuadrado es para corregir el signo.

Por otro lado, otro parámetro muy importante y a tener en cuenta es la dimensión de la población, es decir, el número de individuos que la componen. Se tiene que mantener ese equilibrio entre exploración y explotación que se comentaba en apartados anteriores, ya que una población con un excesivo número de individuos podría entrar en una búsqueda casi aleatoria, y con una población con un número de individuos demasiado pequeño podría dar lugar a un estancamiento dentro de un óptimo local. En

los distintos ensayos realizados se ha variado este parámetro, y de los mismos se deduce que para este problema electroquímico en cuestión, los valores óptimos del número de individuos están comprendidos entre 160 y 200. Un número de individuos mayor a 200 ralentiza excesivamente la ejecución del algoritmo y ya no proporciona una notable mejoría en cuanto al resultado final, mereciendo más la pena hacer renacimientos si se desea profundizar más en el resultado.

Estos datos experimentales parten de unos ensayos hechos previamente a este PFC y se utilizarán para comprobar el comportamiento de los algoritmos, DE básico y SaDE. A continuación se muestran los datos experimentales de partida:

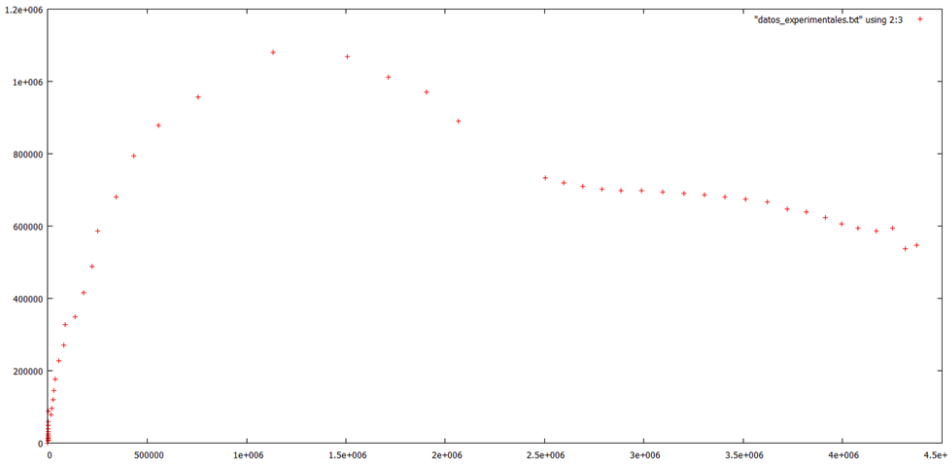

Figura 22. Datos experimentales 1.

Se pueden apreciar los dos semicírculos, y gracias a ello se le puede dar como punto de partida al algoritmo unos valores aproximados de las resistencias, ya que como previamente se comentó, estos valores coinciden con el corte en los ejes de los semicírculos.

Se realizaron 5 ejecuciones de cada uno de los algoritmos para garantizar, por el método de Montecarlo, que se solventa la problemática de la aleatoriedad de los valores iniciales, ya que, como se ha comentado en varias ocasiones, es un proceso estocástico.

La media por el método de Montecarlo:

DE: (DE/rand/1/bin; F=0.5; CR=0.9, N=200) Sa-DE: (LP=30, N=200)

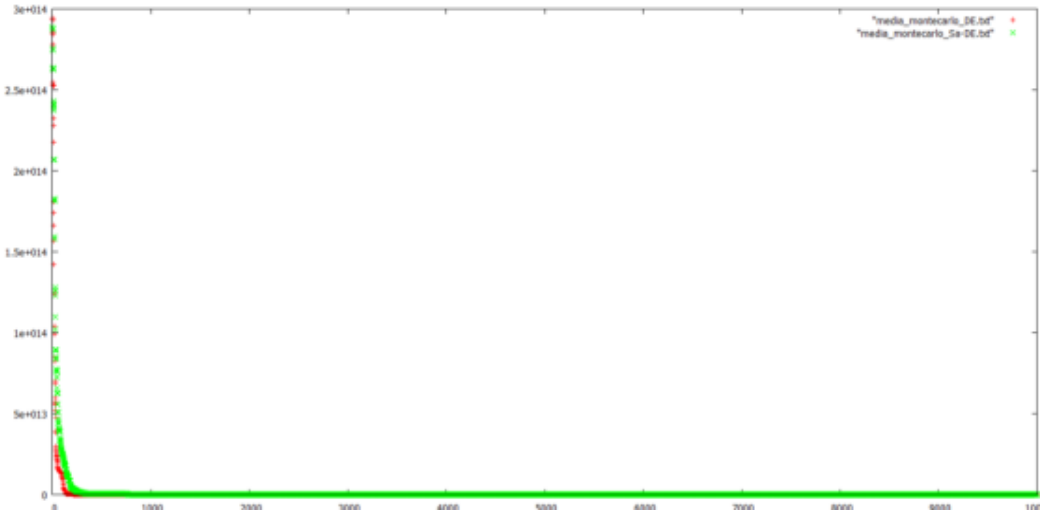

Figura 23.

Haciendo un zoom para las primeras 1000 iteraciones:

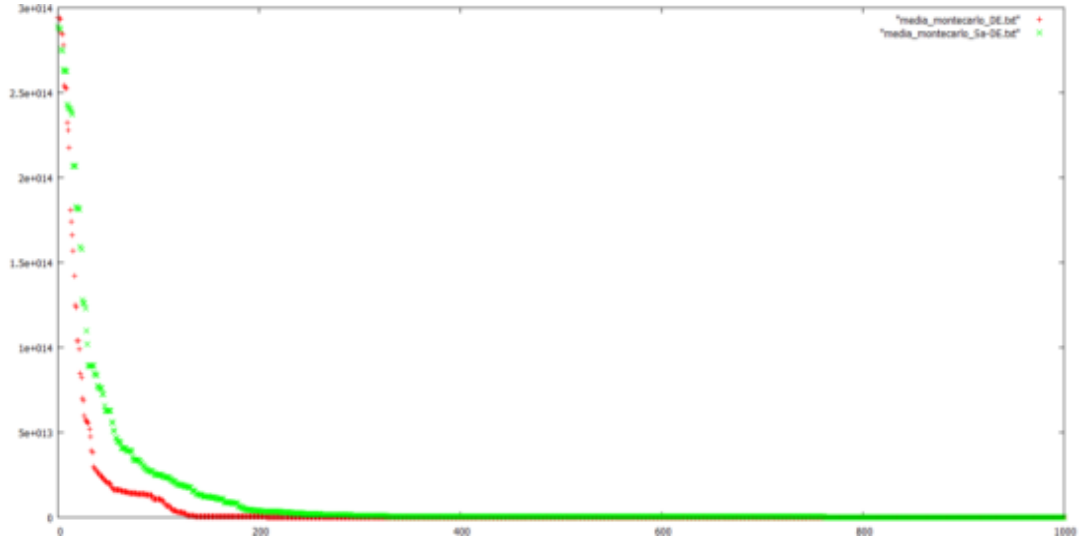

Figura 24.

Como se puede apreciar, los algoritmos convergen, y como se verá a continuación, lo hacen hacia el óptimo correcto. Por otro lado, se da una solución inesperada, y es que el DE básico (rojo) converge más rápidamente que el del SaDE (verde). Esto, lo que quiere decir, es que el método de mutación para crear el vector de prueba del DE básico (DE/rand/1/bin), es el que mejor se ajusta a las condiciones del problema en cuestión, así como sus parámetros asociados CR y F.

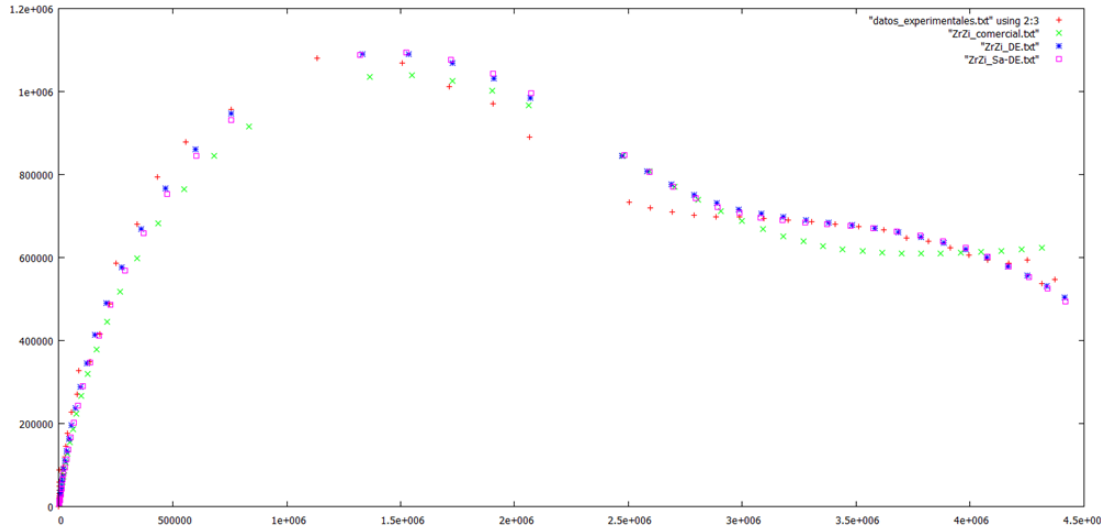

Figura 25.

En la última grafica, los datos experimentales son los puntos marcados en color rojo. El resultado del programa comercial ZSimpWin, da como solución los puntos marcados en verde donde se aprecia que no se obtiene una solución correcta al problema, con un ajuste de $\mathbf{3{,}1980 \cdot 10^{11}}$. Por otro lado, los puntos marcados en color azul y en color rosado, son la solución final correspondiente a los algoritmos DE y SaDE respectivamente; esta solución final es prácticamente la misma para ambos, solo que, como se comentó anteriormente, el DE converge un poco más rápido que el SaDE, siendo sus ajustes de $\mathbf{1{,}1389 \cdot 10^{11}}$ y $\mathbf{1{,}1936 \cdot 10^{11}}$ respectivamente. Se aprecia que estos valores del ajuste son notablemente mejores que los resultantes del ZSimpWin.

Renacimiento:

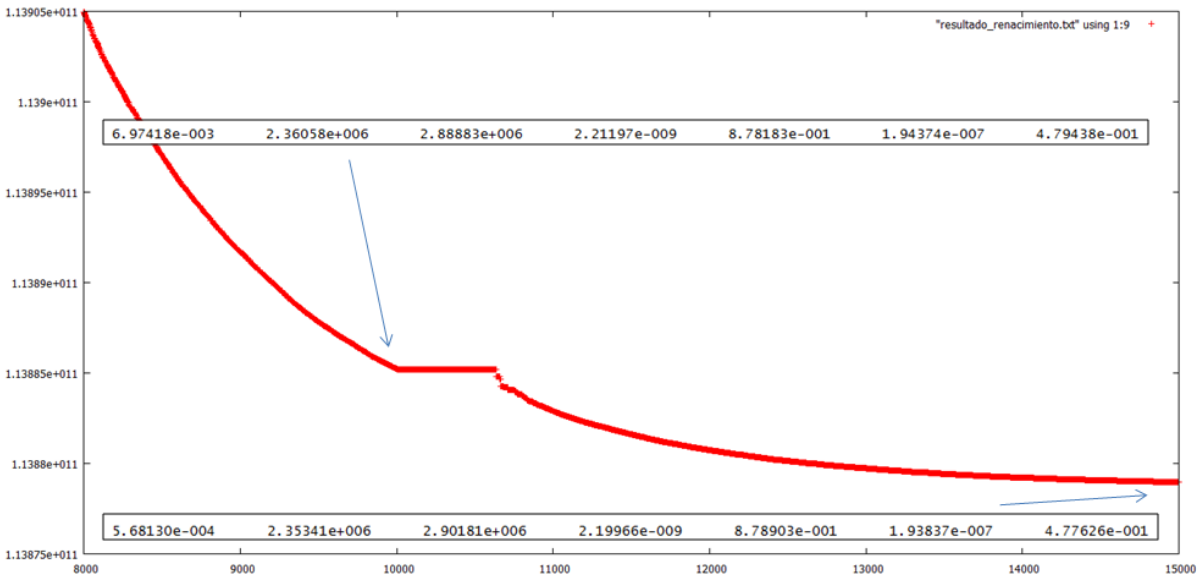

Figura 26.

Se ha aplicado un renacimiento, pero no proporciona una notable mejoría y ni si quiera cambia los valores finales de las variables. El cambio se produce en la tercera

cifra significativa, exceptuando la primera variable (Re), la resistencia del electrolito. Este cambio en Re se debe a que en el algoritmo se fuerza para que su valor sea positivo, en caso contrario, el algoritmo daría como solución un valor negativo de esta resistencia. El motivo es que dicha resistencia, al corresponderse con la resistencia del electrolito (que suele valer aproximadamente 2 o 3 Ω/cm^2), tiene un orden de magnitud mucho menor que el de las otras dos resistencias (10^6). Esto implica que el error en el ajuste de las otras resistencias sea mucho mayor que el valor de la del electrolito. A continuación se verán otros casos en el cual los órdenes de magnitudes coinciden o son muy próximos para los valores de estas resistencias.

El valor de la variable Re (resistencia del electrolito) puede ser medido experimentalmente a la hora de hacer el ensayo de Espectroscopía de Impedancia Electroquímica.

5.4 Comparativa de DE con SaDE para los datos experimentales 2.

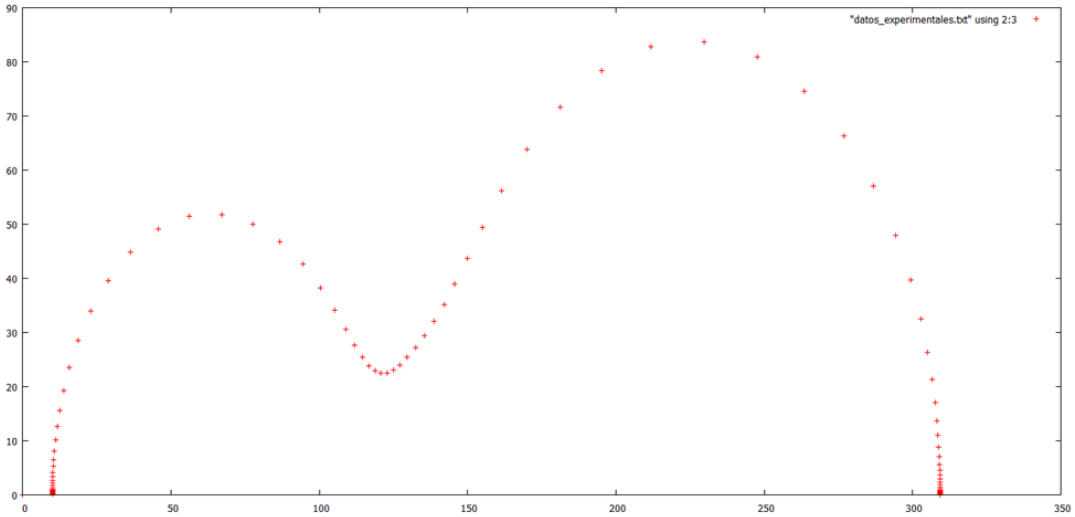

Figura 27. Datos experimentales 2.

La media por el método de Montecarlo (5 ejecuciones):

DE: (DE/rand/1/bin; F=0.5; CR=0.9, N=200) Sa-DE: (LP=30, N=200)

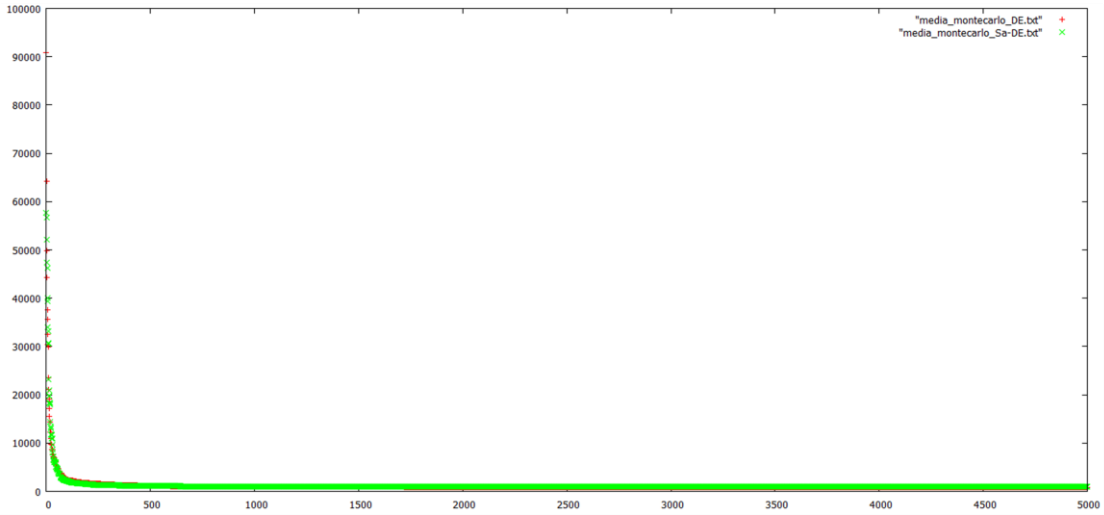

Figura 28.

Haciendo un zoom para las primeras 1000 iteraciones:

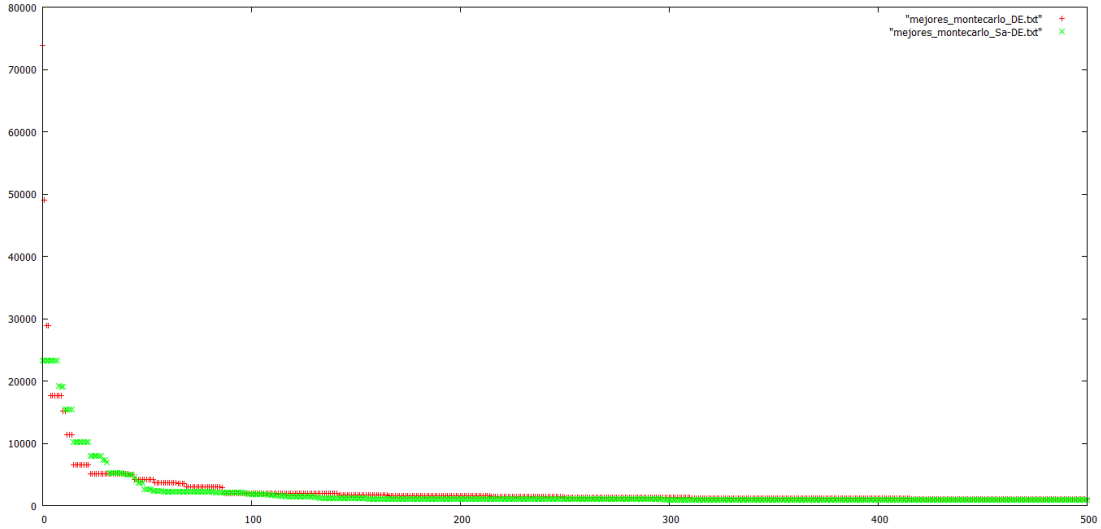

Figura 29.

En este ensayo se ve que los dos algoritmos convergen igual de bien y mucho más rápido que en el ensayo anterior.

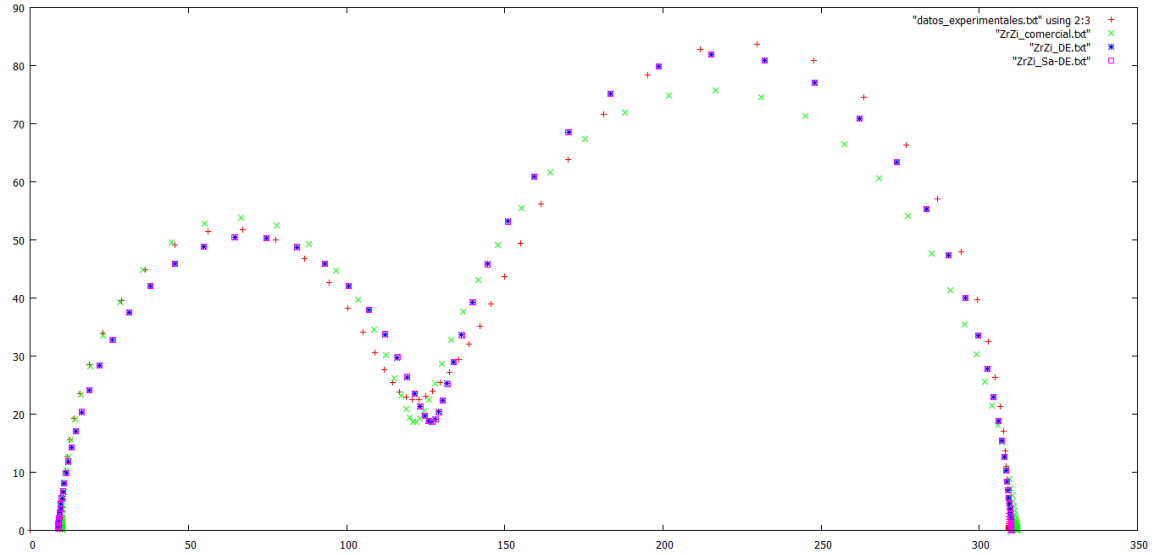

Figura 30.

En la última grafica, los datos experimentales son los puntos marcados en color rojo. El resultado del programa comercial ZSimpWin, da como solución los puntos marcados en verde, obteniendo una solución correcta al problema, con un ajuste de $\mathbf{1{,}8164 \cdot 10^{3}}$. Por otro lado, los puntos marcados en color azul y en color rosado, son la solución final correspondiente a los algoritmos DE y SaDE respectivamente; esta solución final es prácticamente la misma para ambos, solo que, como se comentó anteriormente, el DE converge un poco más rápido que el SaDE, siendo sus ajustes de $\mathbf{9{,}0955 \cdot 10^{2}}$ y $\mathbf{9{,}1664 \cdot 10^{2}}$ respectivamente. Se aprecia que estos valores del ajuste son mejores que los resultantes de la solución que proporciona el ZSimpWin. La dispersión de ambos algoritmos, DE y SaDE, prácticamente se superponen en la gráfica.

5.5 Comparativa de DE con SaDE para los datos experimentales 3.

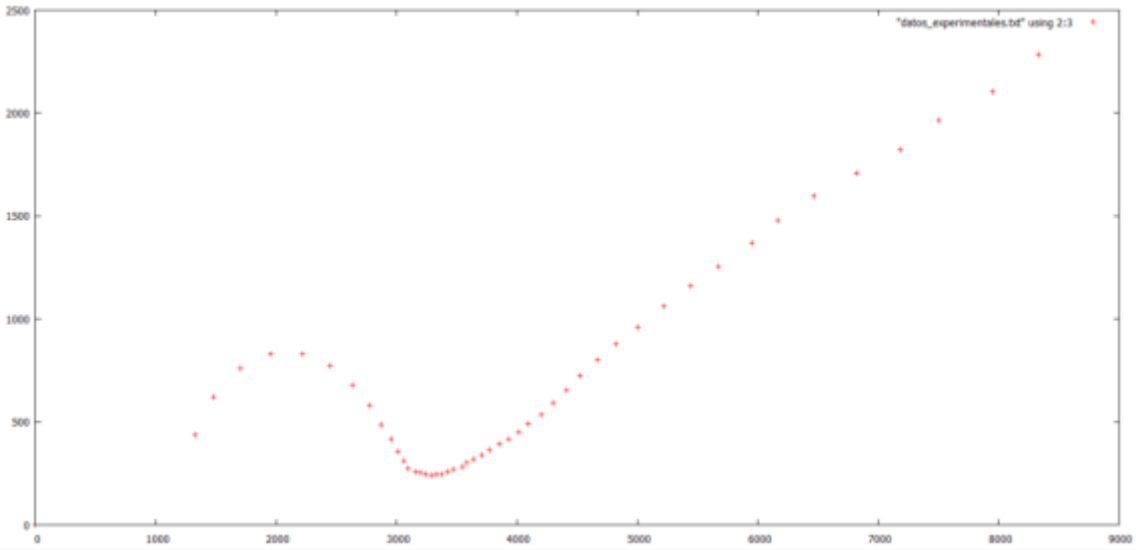

Figura 31. Datos experimentales 3.

La media por el método de Montecarlo (5 ejecuciones):

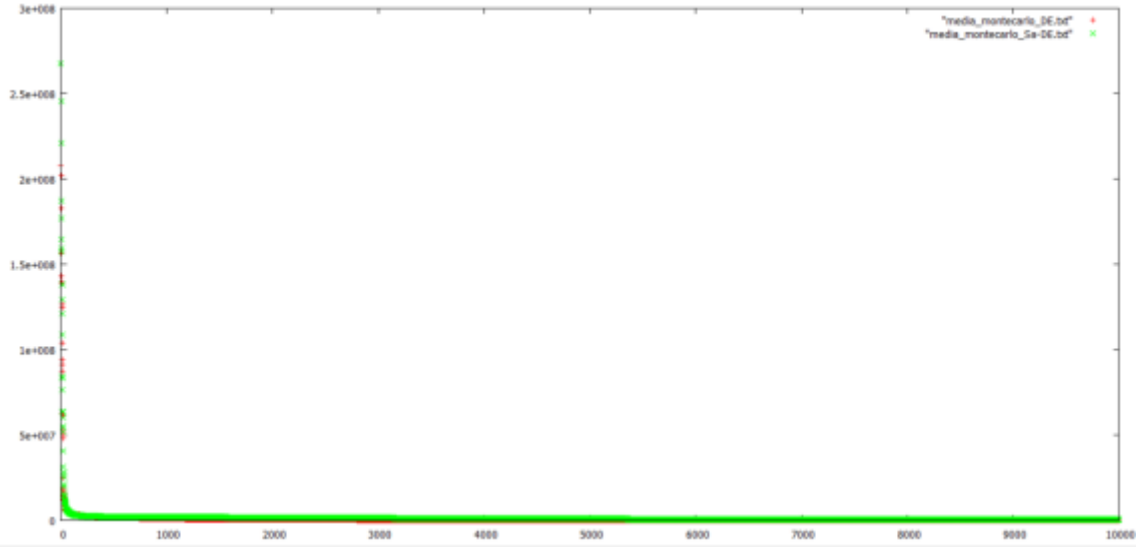

Figura 32.

Haciendo un zoom para las primeras 1000 iteraciones:

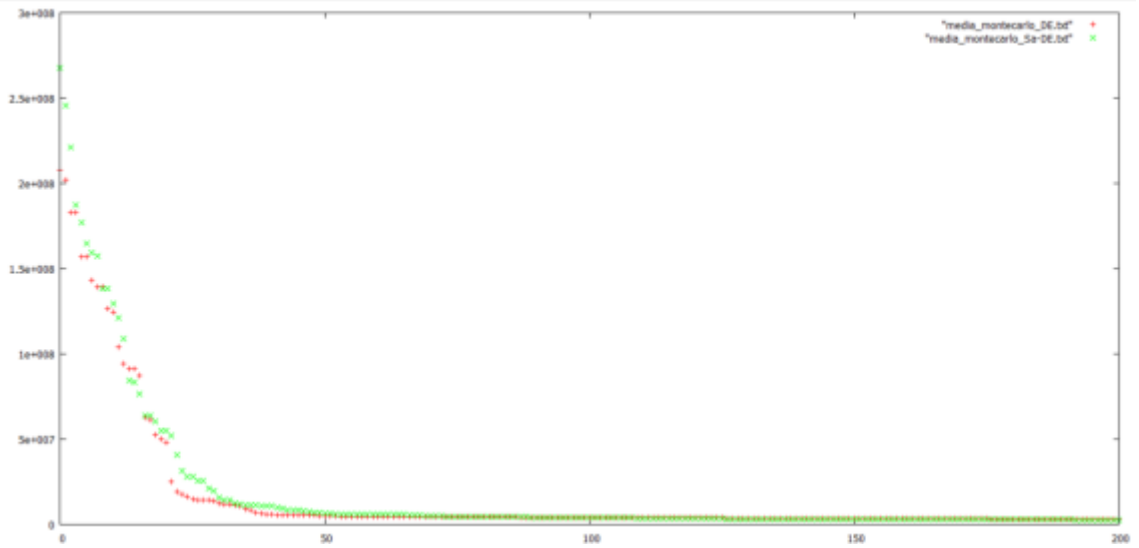

Figura 33.

Ambos algoritmos convergen bien, haciéndolo ligeramente mejor el DE (rojo). En la siguiente gráfica es donde se comprueba que existe una notable diferencia a favor del DE básico.

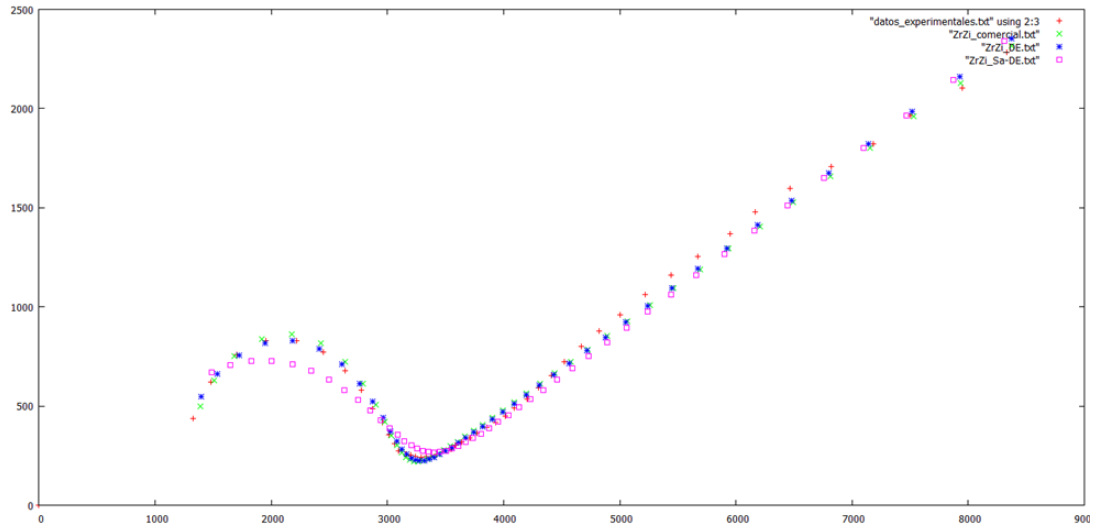

Figura 34.

En la última grafica, los datos experimentales son los puntos marcados en color rojo. El resultado del programa comercial ZSimpWin, da como solución los puntos marcados en verde, obteniendo una solución correcta al problema, con un ajuste de $\mathbf{1{,}0333 \cdot 10^5}$. Por otro lado, los puntos marcados en color azul y en color rosado, son la solución final correspondiente a los algoritmos DE y SaDE respectivamente; esta solución final es prácticamente la misma para ambos, solo que, como se comentó anteriormente, el DE converge un poco más rápido que el SaDE, siendo sus ajustes de $\mathbf{9{,}8082 \cdot 10^4}$ y $\mathbf{3{,}6564 \cdot 10^5}$ respectivamente. De la gráfica se deduce que en este caso merece la pena aplicar la técnica de renacimiento para afinar más el resultado:

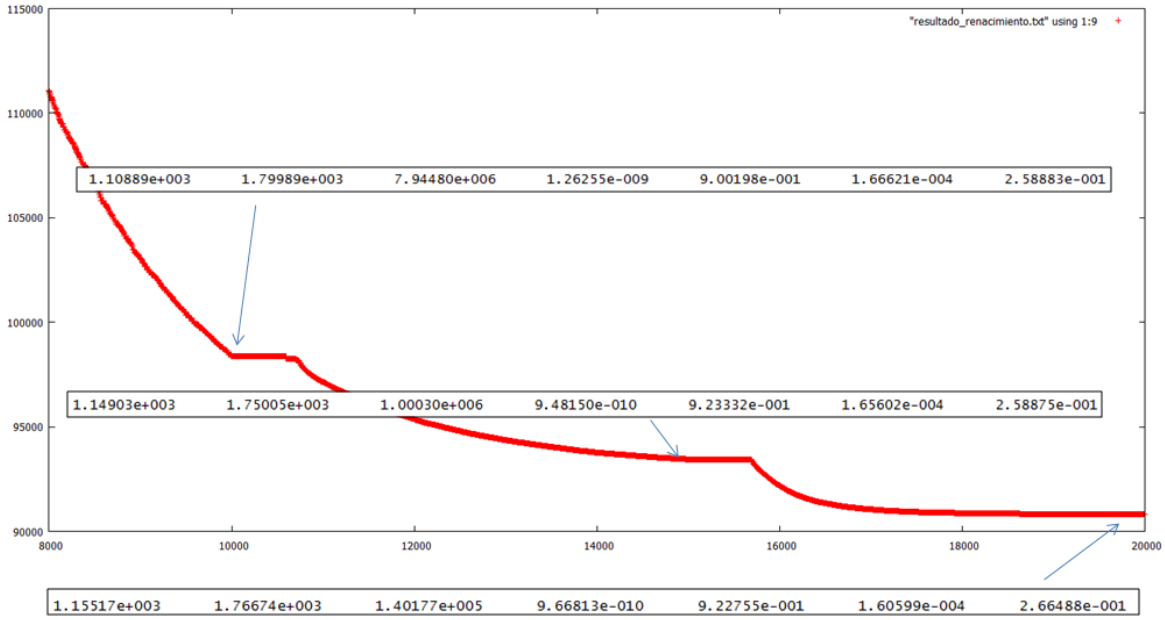

Figura 35.

La evolución de la dispersión antes y después de cada renacimiento:

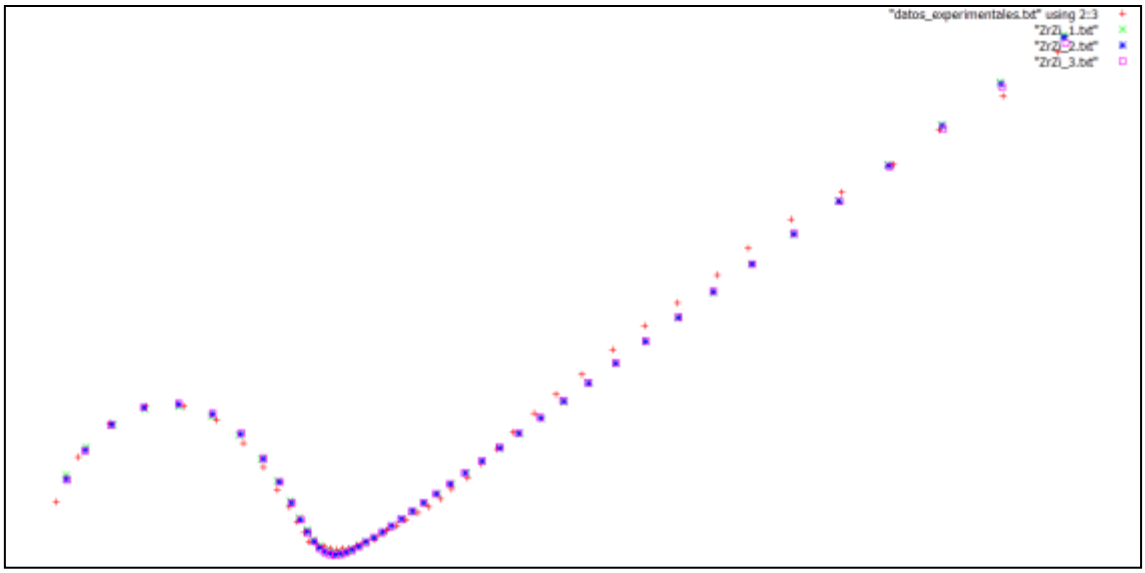

Figura 36.

De esta gráfica se concluye que ya no hay una mejoría significativa y no hay que seguir haciendo renacimientos.

5.6 Uso de coeficientes de ponderación en la función objetivo.

Se ha probado a modificar la función objetivo añadiéndole un coeficiente de ponderación asociado a cada punto de impedancia medido experimentalmente:

$$\sum \left[c_i \times \left(\overline{Z_{re(i)}} - Z_{re(i)} \right)^2 \right] + \sum \left[c_i \times \left(\overline{Z_{im(i)}} - Z_{im(i)} \right)^2 \right] \quad (24)$$

Se ha tomado como valor de este coeficiente la distancia a sus dos puntos adyacentes, de forma que se le dé más peso a los puntos que se encuentran más aislados de la siguiente forma:

$$d_i = \sqrt{(Z_{re(i)} - Z_{re(i-1)})^2 + (Z_{im(i)} - Z_{im(i-1)})^2} + \sqrt{(Z_{re(i)} - Z_{re(i+1)})^2 + (Z_{im(i)} - Z_{im(i+1)})^2} \quad (25)$$

$$K = \sum d_i \quad (26)$$

d_i es la suma de la distancia del punto “i” con su punto anterior “(i-1)”, más la distancia con su punto posterior “(i+1)”

Finalmente c_i toma el valor de:

$$c_i = \frac{d_i}{K} \quad (27)$$

Para los dos puntos extremos de la dispersión, lo que se hace es multiplicar por 2 la distancia a su único punto adyacente, para equipararlos a los demás.

Una vez implementada esta variante en el algoritmo, se comparó su resultado con el proporcionado por la función objetivo sin coeficientes, dando el siguiente resultado:

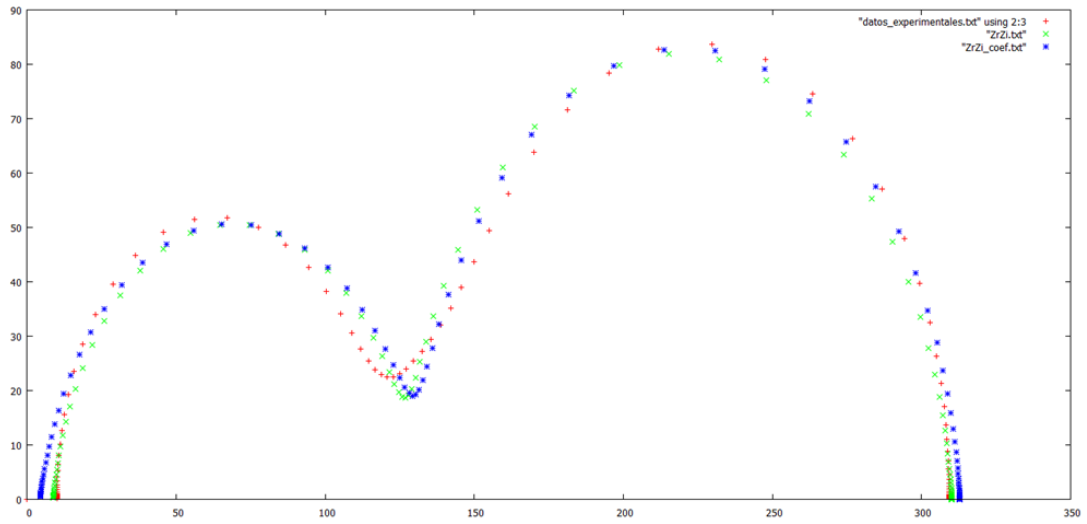

Figura 37. Comparativa del resultado con función objetivo con y sin coeficientes.

Los datos experimentales son los puntos marcados en rojo, los datos resultantes a través de la función objetivo sin coeficientes de ponderación son los marcados en verde, y los datos resultantes a través de la función objetivo con coeficientes están en azul. A la vista de los resultados se ve que se produce lo que ya se preveía, el algoritmo con coeficientes ajusta mejor en aquellos puntos que se encuentran más aislados, por ejemplo los puntos más altos de los semicírculos; pero pierde precisión en aquellas zonas donde se acumulan puntos, por ejemplo las zonas de corte de los semicírculos con el eje real.

Por otro lado, esta acumulación se produce en puntos clave de la dispersión: el paso de un semicírculo a otro y los puntos de corte de los semicírculos con el eje real. Esto nos beneficia, ya que, por ejemplo, los puntos de corte con el eje real marcan los valores de las resistencias del circuito equivalente y la transición de un semicírculo a otro determina los valores de las capacitancias. Por lo anterior, se plantea como mejor solución el uso de este algoritmo sin los coeficientes de ponderación, ya que de forma indirecta, por acumulación de puntos, se le está dando más peso a estas zonas clave de la dispersión.

6. Revisión y desarrollo futuro.

La aplicación informática aquí desarrollada cumple con creces los objetivos marcados, e incluso lo hace mejor que la aplicación comercial ZSimpWin, gracias a que el resultado da un mejor ajuste a los datos experimentales.

En cambio, este software comercial posee más estructuras para circuitos equivalentes distintos, lo que abre una vía para desarrollos futuros de la aplicación objeto de este trabajo. Se puede modificar la estructura del circuito eléctrico equivalente y mediante un desarrollo simple de las ecuaciones de impedancia semejante al expuesto en el apartado 2.5, adaptar el código a otros sistemas electroquímicos distintos al estudio de recubrimientos orgánicos; como por ejemplo las pilas de combustible o la caracterización de superconductores o baterías entre muchos otros.

Otra ventaja es la posibilidad de establecer límites a las variables, para que estas permanezcan siempre en un rango lógico de valores, ya que en muchos casos, el programa comercial falsea resultados cuando algunas de las variables toman valores muy grandes o muy pequeños, sin la posibilidad de ponerle límites durante la ejecución. En el código objeto de este proyecto se puede obligar a que el Algoritmo Evolutivo no permita salir a una variable fuera de sus límites previamente definidos, durante toda la ejecución del algoritmo.

Para culminar el desarrollo de este software, hace falta trabajar más el pre-procesado y el post-procesado de los datos de entrada y salida, ya que se requiere de un procesador gráfico para una visualización cómoda de los resultados.

Referencias bibliográficas.

[1] Antaño Q. R., Galicia L. y González I., "Aplicación de un algoritmo basado en un modelo de medición para la detección de errores en las medidas experimentales de impedancia", Sección de Publicaciones DCBI, UAMI (1997)

[2] Arbor A., " ZSimpWin Version 3.00 - Electrochemical Impedance Spectroscopy Data Analysis Software", EChem Software, Michigan, USA (2001)

[3] Byron S.G., "Programing with C", Second Edition, Schaum's Outlines Series, Mcgraw-Hill (1996)

[4] Boukamp B.A. y Yeum B., "Equivalent Circuit (EQUIVCRT.PAS) - The Computer Assisted Electrochemical Acimmittance Data Analysis System for IBM-PC Computers and Compatibles, Written in Turbo Pascal Version 3.0", University of Twente, Department of Chemical Technology (1989)

[5] Greiner D., Emperador J.M. y Winter G., "Single and multiobjective frame optimization by evolutionary algorithms and the auto-adaptive rebirth operator", *Computer Methods in Applied Mechanics and Engineering,* 194, pp. 3711-3743 (2004)

[6] Hernández J.A., Martínez J. y Amaya J.A., "Software para estimar velocidad de corrosión basado en la técnica de Espectroscopía de Impedancia Electroquímica (EIS), para la Corporación para la Investigación de la Corrosión (CIC)", Sección de Publicaciones EIEET, UIS (2009)

[7] Hongqing C., Jingxian Y. y Lishan K., *"*An evolutionary approach for modeling the equivalent circuit for electrochemical impedance spectroscopy", *The 2003 Congress on Evolutionary Computation,* 3, pp. 1819-1825 (2003)

[8] Maeso O. y Aznárez J.J., "Estrategias para la estimación del impacto acústico en el entorno de carreteras. Una aplicación del método de los elementos de contorno", Sección de Publicaciones IUSIANI, ULPGC (2004)

[9] Minli Y., Xuhong Z., Xiaohong L. y Xizun W., "A hybrid genetic algorithm for the fitting of models to electrochemical impedance data", *Journal of Electroanalytical Chemistry,* 519, pp. 1-8 (2002)

[10] No se especifican, "NOVA - Advanced Electrochemical Software. V 1.10", Metrohm Autolab B.V. (2013)

[11] Qin A.K., Huang V.L. y Suganthan P.N., "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization", *IEEE Transactions on Evolutionary Computation*, 13, pp. 398-417 (2008)

[12] Samin S.A., Matthew L.T., Zijie L., George R.E., Bruno K. y Digby D.M., "Modeling of the electrochemical impedance spectroscopic behavior of passive iron using a genetic algorithm approach", *Electrochimica Acta*, 102, pp. 161-173 (2013).

[13] Santana J.J., González J.E., Morales J., González S. y Souto R.M., "Evaluation of Ecological Organic Paint Coatings via Electrochemical Impedance Spectroscopy", *Journal of Electroanalytical Chemistry*, 7, pp. 6489-6500 (2012).

[14] Santana J.J., González J.E. y Santana F.J., "Fundamentos de la Corrosión Metálica", Sección de Publicaciones ETSII, ULPGC (2004)

[15] Simon D., "Evolutionary Optimization Algorithms", Wiley (2013)