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

AJUSTE DE MODELO FENOMENOLÓGICO DE CELDAS DE BATERÍAS USANDO ALGORITMOS EVOLUTIVOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

FRANCISCO JAVIER VILLA SUÁREZ

PROFESOR GUÍA:
PABLO ESTÉVEZ VALENCIA

MIEMBROS DE LA COMISIÓN:
FELIPE GANA ORTEGA
WILLIAMS CALDERÓN MUÑOZ

SANTIAGO DE CHILE
2015

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO
POR: FRANCISCO VILLA
FECHA: ENERO 2015
PROF. GUÍA: PABLO ESTÉVEZ VALENCIA

AJUSTE DE MODELO FENOMENOLÓGICO DE CELDAS DE BATERÍAS USANDO ALGORITMOS EVOLUTIVOS

Los bancos de baterías de ion-litio son los más utilizados en aplicaciones de alta potencia como respaldos de sistemas eléctricos de potencia y vehículos eléctricos. Uno de los problemas de estos bancos es que pueden sufrir de envejecimiento prematuro por múltiples causas entre las que se destacan las altas temperaturas alcanzadas por las celdas en su operación. Una forma de minimizar el problema es utilizando algoritmos genéticos de optimización para optimizar el diseño de estos bancos. Estos algoritmos requieren evaluar muchas configuraciones, las que se pueden obtener construyéndolas o simulándolas, siendo estas últimas las más utilizadas debido a restricciones de tiempo. Dichas simulaciones se realizan generalmente en softwares denominados Computational Fluid Dynamics (CFD), estos requieren tiempos del orden de minutos o decenas de minutos por simulación. El tiempo requerido para completar una optimización suele ser prohibitivamente grande. Una forma de disminuir estos tiempos es utilizar modelos sustitutos. Con el fin de complementar el software CFD ANSYS, se creó un modelo sustituto basado en ecuaciones fenomenológicas sin embargo, éste difiere en algunos resultados obtenidos por el CFD por lo que es necesario ajustarlo.

El objetivo general del presente trabajo de título es ajustar un modelo fenomenológico de modo que éste entregue valores más cercanos a los del software CFD, esto permitiría emplear dicho modelo como un sustituto del software en un algoritmo de optimización.

Para lograr el objetivo anterior se hicieron modificaciones al modelo. Se escogieron los rangos de las variables de diseño para el modelo basados en el modelo fenomenológico original. Se seleccionaron tres expresiones del modelo a modificar: el factor de fricción, el coeficiente de arrastre y el número de Nusselt, en el modelo original estas se basadas en datos experimentales por lo que pueden tener un margen de error que mejorar. Se hicieron 1500 simulaciones en el software CFD los que separan en conjuntos de entrenamiento, validación y prueba. Con estos datos se ajustaron las tres expresiones del modelo antes indicadas. Para esto se usó como herramienta un algoritmo evolutivo llamado programación genética que permite generar expresiones matemáticas en forma de árbol.

Como resultado final se obtuvo un conjunto de expresiones que representan los coeficientes a ajustar. Los resultados se construyeron de tal manera de mantener las propiedades físicas del problema. Estos hacen que el modelo fenomenológico tenga errores promedio de presión, velocidad y temperatura de 1[Pa], 1[m/s] y 1[°C] respectivamente.

Se concluye que el modelo evolucionado se ajusta mejor a ANSYS, con expresiones que respetan las normas físicas del problema. Por lo tanto, la herramienta programación genética demostró ser útil en este tipo de problemas. Se recomienda finalmente corregir el modelo fenomenológico con las expresiones encontradas.

Tabla de Contenido

Tabla de figuras

Capítulo 1

Introducción

Las baterías de ion-litio son muy utilizadas en una amplia gama de aplicaciones, desde electrónica por ejemplo celulares o computadores portátiles hasta vehículos eléctricos. La presente memoria se centra en las aplicaciones de alta potencia como lo son los respaldos de sistemas eléctricos de potencia y vehículos eléctricos. Aunque este tipo de baterías son las más usadas ya que su desempeño es superior a otras en base a otros compuestos como plomo-acido o níquel-cadmio, su densidad energética es baja en comparación a su límite teórico. Otro problema que tienen estas baterías es que su vida útil se ve afectada por un gran número de factores como la cantidad de ciclos, estado de carga, sobrecargas, entre otros, siendo las altas temperaturas el factor que afecta en mayor medida (Vetter, et al., 2005).

Uno de los posibles enfoques para mejorar el rendimiento de los bancos de baterías es el diseño óptimo de éstos, tomando en cuenta las temperaturas de operación, dimensiones del banco, características eléctricas, etc. El trabajo que se presenta tiene como objetivo el ajuste de un modelo con el que se pueda obtener el comportamiento térmico de los bancos con el fin de ayudar en el diseño de bancos de baterías óptimos.

El diseño de baterías es muy complejo, ya que se deben tener en cuenta diversos criterios que muchas veces son contrarios, ósea, al mejorar uno se empeora el otro. Por eso, este problema se debe abordar como una optimización multi-objetivo. Dentro de la gama de herramientas para resolver este tipo de optimizaciones, los algoritmos genéticos han probado ser muy eficaces, pero tienen el problema que para llegar a una configuración de batería óptima, se deben probar miles de casos, ya sea haciendo los experimentos o simulándolas. Fabricar miles de configuraciones de baterías para probarlas no es factible por lo que se recurre a softwares de simulación de fluido dinámica (CFD: Computational Fluid Dynamics).

Los sistemas de fluido dinámica son descritos por las ecuaciones de Navier-Stokes que son un sistema de ecuaciones diferenciales no lineales. La solución de este tipo de sistemas requiere de algoritmos muy complejos por lo que es necesario el uso de softwares especializado para su evaluación. En la presente memoria se utiliza el CFD comercial ANSYS®.

Si bien los programas CFD reducen considerablemente el tiempo en comparación a construir cada configuración, el costo es muy alto al usar algoritmos evolutivos ya que una simulación puede durar desde un par de minutos hasta media hora o más, dependiendo de la complejidad de la configuración. Por este motivo se han desarrollado modelos sustitutos que buscan aproximar al software de simulación y cuya evaluación es muy rápida. El caso de estudio de esta memoria es un modelo fenomenológico que busca modelar el comportamiento térmico de un banco de baterías para una configuración establecida y donde sólo se pueden cambiar algunos parámetros de ésta. El modelo fenomenológico tiene la ventaja que demora solo una fracción de segundo en obtener el resultado, por lo que disminuye considerablemente el tiempo de la simulación. El problema es que algunas variables de salida que entrega este modelo tienen un error considerable

respecto a las obtenidas por el software CFD, esto puede ser debido a la utilización de datos empíricos que pueden no ajustarse perfectamente y simplificaciones que no necesariamente son ciertas en esta aplicación en particular. Para que el modelo fenomenológico sirva como sustituto debe asemejarse más al software CFD por lo que es necesario ajustarlo.

En este trabajo se propone ajustar el modelo fenomenológico para que pueda ser utilizado como modelo sustituto al software de simulación CFD para el diseño de bancos de batería. Existen múltiples herramientas que pueden servir para este fin. Dentro de la familia de algoritmos evolutivos, hay una que se caracteriza por tener muy buenos resultados en problemas de regresión simbólica, esta es la programación genética. Programación genética representa las expresiones matemáticas como árboles lo que hace que sea muy fácil operar con ellos. Como se requiere sustituir expresiones que representan cantidades físicas, es importante además tener en cuenta que las unidades sean correctas y que las expresiones dependan de variables que tengan sentido físico, por lo que no es exactamente igual a un problema de regresión simbólica ya que se le agregan estas restricciones.

Los modelos sustitutos están siendo muy utilizados en problemas de optimización (Lian, et al., 2010), (Jin, et al., 2000), (Pilat & Neruda, 2013) donde se tiene que simular con softwares de simulación muy costosos, pero en la mayoría de los casos se recurre a herramientas que no tienen representación física. Lo que se propone en esta memoria ofrece una alternativa distinta a los típicos modelos sustitutos ya que se basa en fenómenos físicos.

1.1 Objetivos

El objetivo general de la presente memoria es aplicar algoritmos evolutivos para ajustar un modelo fenomenológico que representa el comportamiento térmico de baterías de ion-litio para que pueda ser utilizado como modelo sustituto del software CFD de simulación ANSYS®.

Los objetivos específicos son los siguientes:

- Modelar el problema, definir variables de entradas y salidas y sus respectivos rangos.
- Encontrar las expresiones del modelo idóneas para ser modificadas
- Hacer un diseño de experimentos adecuado para el problema.
- Realizar las simulaciones necesarias en el software CFD ANSYS.
- Aplicar el algoritmo de programación genética para obtener las expresiones requeridas.

1.2 Estructura de la memoria

Para exponer el trabajo realizado se presenta este documento estructurado en los siguientes capítulos:

Introducción: En este primer capítulo se establece la motivación para trabajar en el tema, una descripción general del problema y los objetivos del trabajo.

Antecedentes y Revisión Bibliográfica: Se detallan los conceptos necesarios para la comprensión del presente trabajo. Aquí se describen temas como las baterías de ión-litio, modelos paramétricos, técnicas de inteligencia computacional en particular programación genética, modelos sustitutos y sus aplicaciones, simulaciones en software CFD y por último se explican los principios del modelo fenomenológico que será utilizado en la memoria.

Metodología e Implementación: En este capítulo se presentan los objetivos específicos del trabajo, las herramientas a utilizar, los algoritmos implementados junto con una descripción detallada de estos. Se describe la implementación del algoritmo de programación genética, los parámetros utilizados, las funciones adicionales y las alteraciones hechas al algoritmo con el fin de obtener mejores resultados.

Resultados: Se muestran los resultados obtenidos en todos los casos estudiados, estos son presentados con sus expresiones correspondientes así como también con gráficos. Los casos de estudio corresponden a las expresiones para el factor de fricción, coeficiente de arrastre y número de Nusselt.

Conclusiones: Aquí se presentan las conclusiones obtenidas a partir del trabajo realizado en la memoria junto con propuestas de trabajo futuro.

Capítulo 2

Antecedentes y Revisión Bibliográfica

2.1 Empaquetamiento de baterías de ion-litio

2.1.1 Celdas de baterías de ion-litio

Los bancos de baterías de ion-litio son dispositivos utilizados para el almacenamiento de energía en diversas aplicaciones destacando entre las más importantes los vehículos eléctricos y baterías de sistemas eléctricos de potencia. Una celda es la unidad fundamental de un módulo y éste a su vez es lo que compone un banco de baterías, en otras palabras, un banco está compuesto por módulos y un módulo por celdas. Las características del banco se obtienen a partir de las características de las celdas pues el voltaje, la corriente y la capacidad vienen determinadas principalmente por los balances de las reacciones electroquímicas que se producen dentro de ellas.

Las celdas de ion-litio están formadas por 4 componentes principales: el ánodo, el cátodo, el separador y el electrolito, lo que por medio de una reacción electroquímica son capaces de entregar corriente eléctrica. El ánodo por lo general está compuesto por carburo de litio (Li_4C) y el cátodo por óxido de litio metálico. El separador es una membrana porosa que evita el contacto físico entre los electrodos. Por último, el electrolito es una sal de litio cuya finalidad es actuar como medio conductor entregando los iones necesarios para la reacción química, pero no forma parte de ésta. En la figura 2.1 se ilustran las partes de una celda.

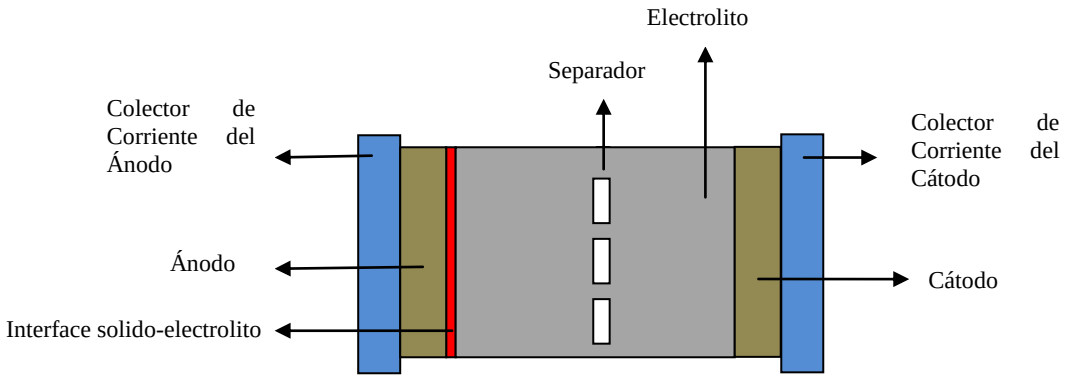

Figura 2.1 Partes de una celda de ion-litio

En el proceso de descarga los iones de litio se separan del ánodo y viajan a través del electrolito hacia el cátodo desde donde son enviados al circuito externo. En el proceso contrario, o sea, la carga, los iones circulan en la dirección opuesta.

Los dos tipos de geometría más comunes en las celdas de litio son las prismáticas o rectangulares, y las cilíndricas. Estos tipos determinan las características de la celda tales como la resistencia interna, la densidad energética y la densidad de potencia.

2.1.2 Medidas de condición de una batería

Para poder caracterizar una batería se necesita de parámetros que la representen, estos además sirven para clasificar, describir y comparar diferentes tipos de baterías.

Estado de Carga (*State of Charge,* SOC): es el porcentaje de la capacidad actual de la batería con respecto a su capacidad máxima. Un SOC de 100 % representa a una batería completamente cargada en tanto que 0% representa a una batería completamente descargada. El SOC es calculado a través de la integral de la corriente en el tiempo (ecuación 2.1), donde $C(t)$ representa la capacidad (en Ah), $I(\omega)$ a la corriente (en A) y $\eta(\omega)$ a la eficiencia de la descarga o carga.

(2.1) $$SOC(t) = SOC(0) + \frac{1}{C(t)} \int_0^t \eta(\omega)\, I(\omega) d\omega$$

Profundidad de Descarga (*Depth of Discharge*, DOD): representa cuanto se ha descargado la batería y se expresa como un porcentaje de la capacidad máxima.

Resistencia Interna: corresponde a la resistencia que se mide entre el ánodo y cátodo de la celda. Depende entre otras cosas de la temperatura y del estado de carga. A mayor resistencia interna, mayor es el deterioro ya que hay más energía disipada como calor. Las celdas vienen con esta característica dada de fábrica por lo que un aumento en la resistencia reflejará el envejecimiento del acumulador.

Por otra parte, las especificaciones de una celdas y que son entregadas por los fabricantes corresponden al voltaje nominal, el voltaje de corte (en Volt), la capacidad nominal (en Ah), energía nominal (en Wh), energía específica (energía nominal por unidad de masa, en Wh/kg), densidad de energía (energía por unidad de volumen en Wh/L), potencia específica (potencia máxima por unidad de masa en W/kg) y densidad de potencia (en W/L).

2.1.3 Envejecimiento de una celda de batería de ion-litio

Dadas las aplicaciones a las que se ven expuestas las baterías, como automóviles eléctricos o híbridos, sistemas de respaldo de sistemas eléctricos de potencia, etc., es de vital importancia maximizar su vida útil. Esto apunta también a evitar el envejecimiento prematuro de éstas, que puede ser causada por diferentes motivos como la descomposición del electrolito, formación de capas pasivas, disolución de material activo, etc. estos y otros fenómenos son revisados en detalle en (Arora, et al., 1998) y (Vetter, et al., 2005). En la tabla 2.1 se muestran los principales fenómenos que producen el deterioro y sus causas.

Como se puede apreciar en la tabla 2.1, las altas temperaturas intervienen en la mayoría de los fenómenos que contribuyen al envejecimiento prematuro, en (Vetter, et al., 2005) se muestra que un aumento en 15°C en la temperatura de operación (en torno a los 25°C) hace decaer los ciclos de vida útil a la mitad. Dado lo anterior, es de gran importancia controlar este problema para poder obtener baterías que tengan una vida útil lo más larga posible.

Tabla 2.1 Fenómenos que provocan el envejecimiento prematuro de las baterías y sus causas (Vetter, et al., 2005)

Fenómeno	Causa
Descomposición de Electrolito, crecimiento de interface con ánodo	Altas temperaturas y alto SOC
Intercalamiento de solvente en el ánodo	Sobrecarga
Disminución de superficie accesible en ánodo	Altas temperaturas y alto SOC
Cambios en la porosidad del ánodo	Alta tasa de ciclado y alto SOC
Pérdida de material activo por cambio de volumen	Alta tasa de ciclado y altas profundidades de descarga
Descomposición de uniones	Altas temperaturas y alto SOC
Corrosión del colector del ánodo	Sobredescarga y bajo SOC
Descomposición del electrolito por litio metálico	Baja temperatura, alta tasa de ciclado, desbalance de celdas o fallas constructivas
Disolución de material activo del cátodo en el electrolito	Altas temperaturas y bajo SOC
Descomposición de material activo del cátodo en el electrolito	Bajo SOC
Crecimiento de interface cátodo-electrolito	Altas temperaturas y alto SOC

2.1.4 Aspectos básicos del empaquetamiento de baterías

El empaquetamiento de baterías es el proceso de diseño de un banco de baterías. Este comienza con la selección del tipo de celda que es la unidad básica de los bancos, cuyas características eléctricas definen al banco. Estas celdas luego son puestas en arreglos o módulos. Finalmente, un pack es un conjunto de módulos. Este pack junto con componentes como ventiladores, contactos eléctricos, computadores de control, y sensores son los que componen un banco de baterías. En la figura 2.2 se puede ver un esquema del diseño de bancos de baterías.

Los ventiladores juegan un papel fundamental ya que una mala elección puede llevar a las celdas a temperaturas no deseadas. Sin embargo, existen muchos objetivos de diseño que afectan el rendimiento de un banco ya que requiere balancear seguridad, costos, durabilidad, etc. (Pesaran, et al., 2009). La mecánica es otro punto que no se puede dejar de lado ya que aquí se toma en cuenta el ensamblaje, condiciones estructurales, puntos de anclaje, interfaces mecánicas con otros dispositivos, entre otros. Dentro del aspecto eléctrico se encuentran las interfaces eléctricas, monitores de voltaje, temperaturas y corrientes, capacidad, energía mínima disponible, estado de

carga del banco, estado de salud, etc. Estos son solo algunos de los aspectos que se deben tomar en cuenta a la hora de enfrentar el problema de empaquetamiento de baterías de ion-litio por lo que se convierte en un problema de optimización multi-objetivo.

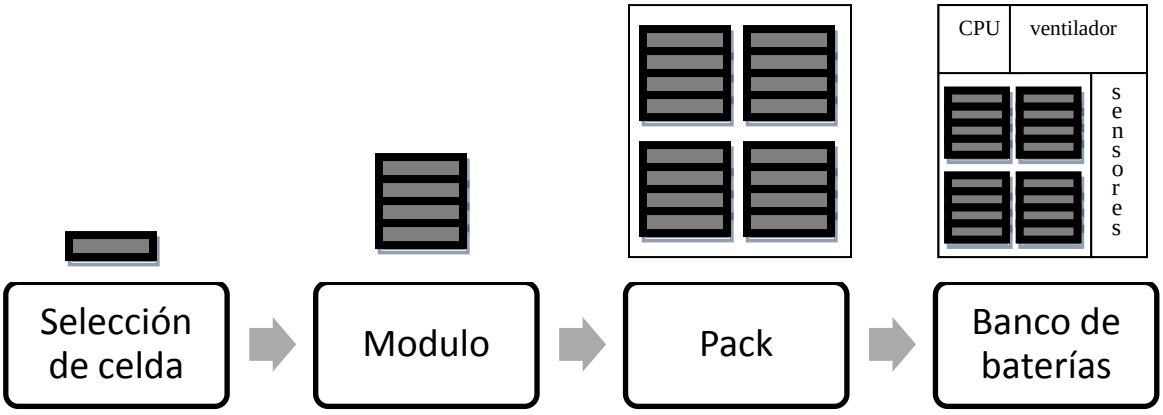

Figura 2.2 Etapas del empaquetamiento de baterías

Contar con herramientas computacionales adecuadas es fundamental para reducir el tiempo requerido para obtener un diseño que cumpla con las condiciones necesarias. Para eso existen softwares de simulación que si bien obtienen resultados muy precisos, toman demasiado tiempo en las simulaciones. Lo anterior ha llevado a desarrollar otro tipo de métodos como por ejemplo modelos sustitutos.

2.2 Modelos Paramétricos

Hay distintos modelos de regresión sobre como estimar una función dado un conjunto de puntos. Uno de ellos es utilizando modelos paramétricos. Estos parten de una función de distribución de probabilidad que se asume conocida y el problema se reduce a estimar los parámetros que mejor se ajusten a las observaciones con las que se cuentan. Dichos modelos resultan muy eficientes si los datos efectivamente tienen una forma similar a la distribución propuesta en un principio, pero pueden llegar a ser muy sensibles si la hipótesis de la distribución está equivocada, las muestras están en un espacio muy reducido o son muy pocas.

Supongamos el siguiente ejemplo: sea Y una variable dependiente que puede ser expresada como

(2.2) $$Y = f(x_1, x_2, \dots, x_n) + \varepsilon$$

Donde x_i son las variables independientes, f es la función que relaciona las variables utilizadas y ε es una perturbación aleatoria. Los modelos paramétricos asumen conocida la función f, por ejemplo lineal

(2.3) $$f(x) = ax + b$$

Donde x representa al vector de variables independientes. De esta forma, el problema se reduce a estimar los coeficientes a y b de la ecuación 2.3 que mejor se ajusten a los datos.

Por otro lado, los modelos no paramétricos son aquellos que no establecen una distribución conocida a priori si no que utilizan formas funcionales flexibles que aproximen la función objetivo.

En ambos casos es necesario estimar parámetros de la forma funcional escogida. La diferencia radica en que para el caso de los modelos paramétricos, la elección de la forma del funcional es hecha a priori, lo que puede llevar a un modelo que no ajuste bien a los datos. En cambio, los modelos no paramétricos se pueden ajustar bien a un conjunto de datos cualquiera ya que presenta muchas menos restricciones. Sin embargo, se ha comprobado que si la distribución de la muestra es normal, los métodos paramétricos resultan más eficientes que aquellos no paramétricos. En (Yatchew, 1998) se hace un estudio de las ventajas y desventajas de la utilización de las técnicas de regresión no paramétricas.

Los modelos no paramétricos ajustan cualquier función y por lo tanto tienen menor sesgo, pero fluctúan mucho ya que resultan ser funciones muy complejas. En cambio, los modelos paramétricos suelen tener un mayor sesgo si se intenta modelar una función que no tenga la forma que se supone a priori, pero sus fluctuaciones suelen ser menores. Lo anterior describe el llamado dilema sesgo-varianza, el cual es inherente al proceso de modelado en general. Cuando existe gran sesgo, el modelo sugerido está lejano a la solución, es decir, hay subajuste (*underfitting*). Por otro lado, cuando se presenta una varianza considerable, el modelo se ajusto incluso al ruido, lo que se le llama sobreajuste (*overfitting*) (Beltrán, 2008).

Por lo tanto, un problema se puede abordar desde cualquiera de los dos enfoques, aproximando la función por un modelo paramétrico o uno no paramétrico, la elección depende de muchos factores como la facilidad de implementación, información del modelo que se quiere aproximar, tiempo que demora, carga computacional, entre otros. Y si bien es posible comparar entre modelos paramétricos ya que como se expuso anteriormente, el desempeño de estos depende en gran medida de la forma de la función, comparar entre modelos paramétricos y no paramétricos puede ser muy complejo por lo que un factor importante es también la experiencia del diseñador.

2.3 Técnicas de inteligencia computacional

Inteligencia computacional es un término que comprende un gran y diverso campo de teorías y técnicas que están en permanente evolución. Estos buscan resolver problemas complejos de mucha incertidumbre que muchas veces los enfoques clásicos no pueden abordar. La característica común entre tan diversas técnicas es que todas ellas están inspiradas de alguna manera en la naturaleza. Entre ellas están las redes neuronales (Rumelhart & McClelland, 1986), la computación evolutiva (Fogel, et al., 1966) y los sistemas difusos (Zadeh, 1965).

La inteligencia computacional utiliza la computación para proveer a los sistemas con la habilidad de aprender y/o tratar situaciones nuevas, en otras palabras, los hace inteligentes. Entendiendo sistema inteligente como aquel que posee capacidad de generalización, descubrimiento, asociación y/o abstracción. Cualquier sistema que tenga por lo menos uno de estos atributos, se considera inteligente (Ruiz del Solar & Held, 2012).

2.3.1 Algoritmos genéticos

Los algoritmos genéticos son parte de las técnicas de inteligencia computacional, en particular de los algoritmos evolutivos. Estos son algoritmos de búsqueda heurística que se utilizan

generalmente en problemas de optimización. Están basados en el proceso de selección natural por lo que se utilizan herramientas como herencias, mutación, selección y *crossover* (reproducción).

En un algoritmo genético una población de posibles soluciones a un determinado problema evoluciona hasta llegar a una mejor solución. Esto se hace iniciando con una población creada aleatoriamente que evoluciona mediante un proceso iterativo donde cada iteración representa una generación. En cada iteración, una función de *fitness* evalúa a todos los individuos de una generación; el *fitness* es usualmente la función objetivo del problema de optimización que se desea resolver. Los individuos de mejor *fitness* son escogidos estocásticamente para ser modificados y crear una nueva generación y así continua el algoritmo hasta alcanzar un criterio de término que puede ser por ejemplo, un número máximo de iteraciones. En la figura 2.3 se muestra un esquema con las principales etapas de un algoritmo genético.

Estas etapas pueden variar según el algoritmo específico del que se esté hablando, pueden ser modificadas y/o agregar nuevas. Por ejemplo, muchos algoritmos utilizan lo que se denomina búsqueda local, que consiste en buscar un individuo más apto dentro de una vecindad a partir del mejor individuo de una generación. En otros casos pueden variar los operadores genéticos, como por ejemplo modificar el *crossover* para que involucre a tres o más individuos en vez de solo dos.

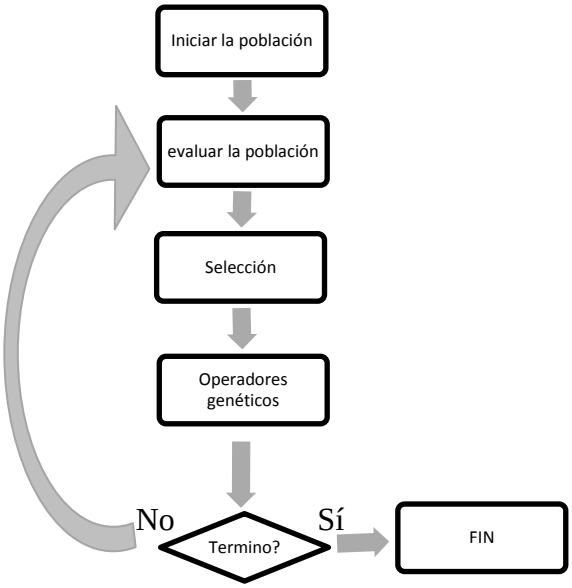

Figura 2.3 Principales etapas de un algoritmo genético

Inicialización del algoritmo: Inicialmente, un conjunto de posibles soluciones (individuos) es creado por lo general aleatoriamente. La cantidad de individuos dependerá del problema en particular pero típicamente son del orden de cientos o miles. Es importante generar aleatoriamente estos individuos para tener una diversidad de posibles soluciones.

Evaluación de la población: En esta etapa a todos los individuos se les calcula su función de fitness para luego poder discriminar los mejores.

Selección: En cada generación existe el proceso de selección donde los individuos son seleccionados para generar la siguiente generación. Mientras mayor sea el fitness hay mayor probabilidad de ser elegido.

Operadores genéticos: para generar nuevos individuos en la próxima generación se utilizan estos operadores, los más comunes son mutación y *crossover*. El primero es un cambio aleatorio en un individuo, esto tiene típicamente una probabilidad muy baja de ocurrencia. El segundo por otra parte es el más frecuente ya que la mayoría de los individuos de una generación provienen de este operador, el que consiste en tomar un par de soluciones de una generación, combinarlas produciendo "hijos" que son individuos de la nueva generación. Estos procesos se repiten hasta tener una cantidad de individuos adecuada para la nueva iteración. Si bien el proceso de *crossover* es intuitivamente entre dos individuos ya que está basado en la reproducción sexuada, (Ting, 2005) sugiere que más de dos padres pueden generar mejor descendencia.

Condición de término: existen muchos criterios de término del algoritmo, entre ellos están: encontrar la solución que satisface cierto criterio, alcanzar un máximo de iteraciones, inspección manual, por capacidades computacionales, llegar a un estancamiento en el valor máximo del fitness en sucesivas generaciones o una mezcla de todas la anteriores.

2.3.2 Optimización multi-objetivo con algoritmos evolutivos

Un problema de optimización multi-objetivo es aquel que tiene un conjunto de objetivos que debe cumplir y estos tienen, por lo general, una relación inversa entre ellos. Formalmente

$$\min_{x\in\Omega} F(x) = \left(f_1(x), \dots, f_m(x)\right)^T \tag{2.4}$$

Donde Ω es el espacio de decisión y $x \in \Omega$ es un vector de decisión, m es la cantidad de funciones objetivo que tiene el problema. Estas funciones típicamente están en conflicto por lo que si una mejora, otra puede empeorar. Es por esto que en este tipo de problemas no existe una solución única que minimice todas las funciones objetivo, en cambio, las soluciones son parte del conjunto de soluciones óptimas de Pareto. Una solución es óptima de Pareto si no hay otra solución factible que haga que uno de sus elementos baje más sin hacer que otro suba. El conjunto de estas soluciones es el conjunto óptimo de Pareto y a la imagen de este conjunto en el espacio de objetivos es denominado Frente de Pareto. Si se refiere a este frente como aquel obtenido tomando en cuenta todas las soluciones posibles es llamado Frente de Pareto verdadero, mientras se si obtiene un frente tomando en cuenta solo un subconjunto de posibles soluciones es un Frente de Pareto (asociado a dicho subconjunto).

Los algoritmos evolutivos utilizados para optimización multi-objetivo buscan un Frente de Pareto lo más cercano al frente verdadero. Una de las ventajas de estos algoritmos en este tipo de problemas es que trabajan con una población en vez de con un solo individuo a la vez, lo que es más conveniente ya que en este caso siempre se tendrá más de una solución. Además, debido a la naturaleza de los operadores evolutivos, no es tan importante la inicialización de la población ya que estos operadores aseguran una exploración continua de nuevas soluciones, esto representa un problema no menor en enfoques más clásicos ya que las soluciones están muy relacionadas con la inicialización de las variables. Otro punto importante es que este tipo de algoritmos no tiene la restricción de que la función objetivo sea derivable lo que lo hace una buena opción cuando la función objetivo no es derivable o de derivación muy compleja.

Otra ventaja que tienen los algoritmos evolutivos es que permiten hacer búsquedas locales lo que se hace por ejemplo luego de encontrar el mejor individuo en una generación. Se realiza una búsqueda local intentando encontrar en una vecindad cercana algún individuo que mejore su fitness. Luego de esto existen dos métodos muy utilizados para completar el proceso. El aprendizaje *Baldwiniano* y el *Lamarckiano*. El primero reemplaza el fitness del individuo original

por el fitness mejorado encontrado en la búsqueda local. En cambio, el segundo reemplaza el individuo completo por el recién encontrado. En la figura 2.4 se puede ver un esquema que muestra la diferencia entre estos dos tipos de aprendizaje.

Luego de aplicar algún tipo de aprendizaje, la generación quedará modificada en al menos un individuo. Esto hace que en cada iteración usualmente se alcancen mejores resultados por lo que acelera el proceso de optimización y se pueden llegar a buenos resultados con una menor cantidad de generaciones. Aplicaciones del aprendizaje *lamarckiano* y una descripción detallada del *baldwiniano* se pueden ver en (Lim, et al., 2010).

Otra técnica es utilizar modelos sustitutos, los que se detallan en la sección 2.4

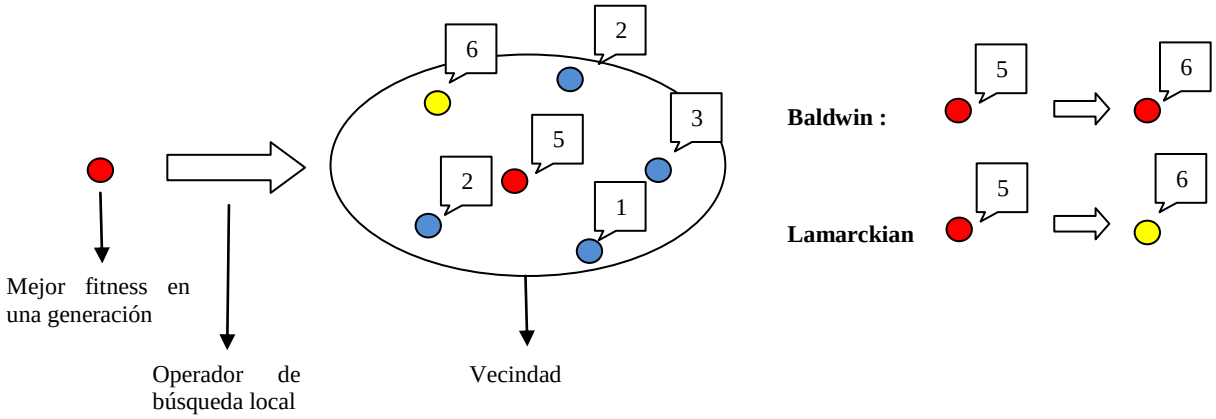

Figura 2.4 Aprendizaje Lamarckiano y Baldwiniano. Se muestran los individuos y sus valores de fitness

2.3.3 Programación Genética

La programación genética es un paradigma de los algoritmos evolutivos, que se utiliza generalmente para encontrar programas computacionales así como también encontrar expresiones matemáticas (Koza, 1992). Se puede considerar un caso especial de los algoritmos genéticos donde los individuos representan programas computacionales o expresiones matemáticas.

Para la representación de los programas o expresiones se utilizan estructuras de datos conocidas como árboles. Un árbol es un elemento compuesto por nodos que se conectan entre sí de forma jerárquica. Los nodos que no tienen nodos más abajo se denominan hojas mientras que los demás son nodos internos. En particular, los árboles utilizados en programación genética tienen en sus hojas las variables de diseño o valores constantes, mientras que en sus nodos internos tienen funciones matemáticas u operadores condicionales. En la figura 2.5 se puede ver la representación de una expresión matemática. Este tipo de estructura hace muy fácil su evaluación y modificación.

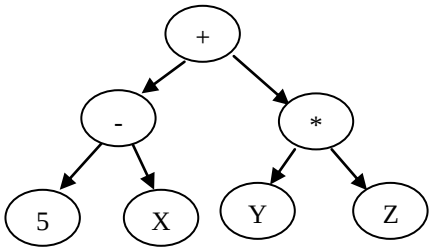

Figura 2.5 Árbol que representa la expresión matemática (5-

Los operadores evolutivos, como el crossover y la mutación, aunque distintos de aquellos del algoritmo genético, son muy fáciles de implementar debido a la representación conveniente de los individuos.

Crossover: El crossover es aplicado a dos árboles de una generación seleccionando un nodo cualquiera en cada padre e intercambiando el subárbol que tiene como raíz (nodo superior) a estos nodos. De esta manera se crean dos individuos nuevos. En la figura 2.6 se ilustra este operador. Por tener esta forma, en general la descendencia es muy distinta a los padres lo que hace que el algoritmo pueda recorrer más rápido el espacio de soluciones posibles.

Mutación: La mutación afecta a solo un individuo de la población reemplazando con un valor aleatorio el contenido de un nodo. Es importante que este operador mantenga la integridad del individuo afectado por lo que debe cambiar el nodo con valores que mantengan la coherencia del árbol. Por ejemplo, si un nodo interno es un operador que acepta una sola variable y como es de esperarse tiene un solo hijo, el operador de mutación no puede cambiar este por un operador binario, o debe tener la capacidad de lidiar con estas situaciones. En la figura 2.7 se puede ver que el nodo elegido para ser mutado no pudo haber tomado un valor de por ejemplo una constante o un operador unitario como seno o coseno.

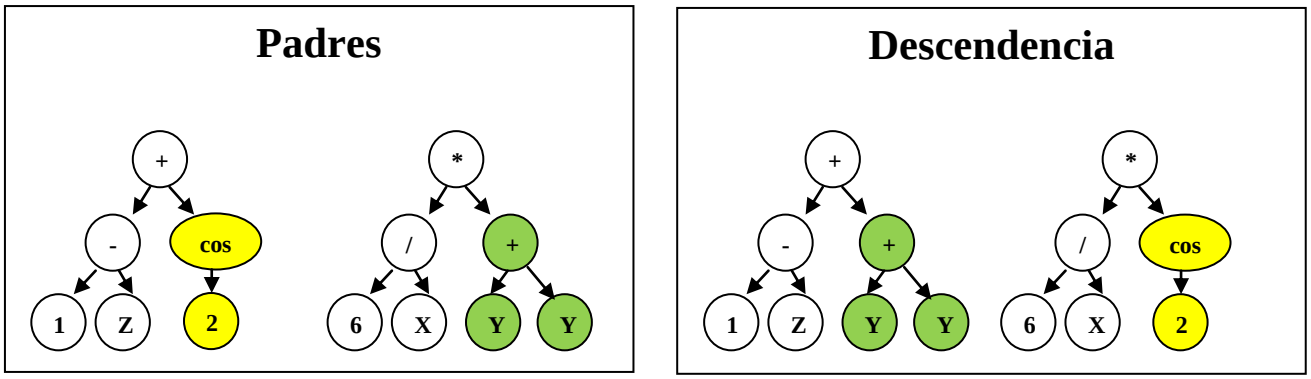

Figura 2.6 Operador de crossover en árboles

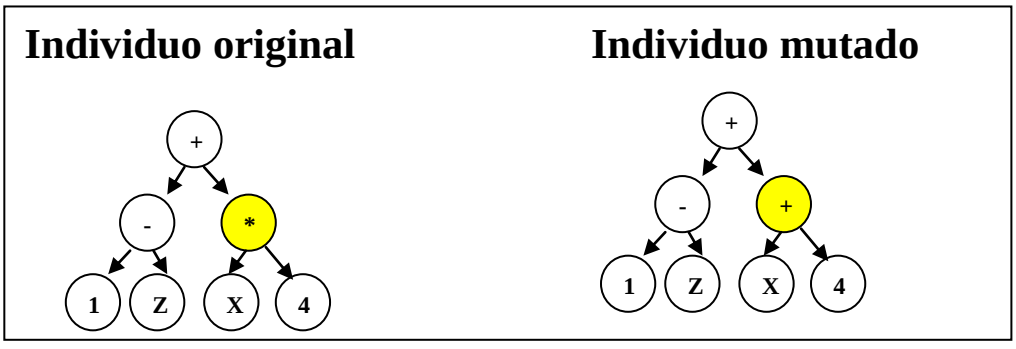

Figura 2.7 Operador de mutación en un árbol

2.3.3.1 Inicialización del algoritmo

Antes de resolver un problema utilizando programación genética es necesario definir cinco pasos con los que se podrá dar inicio al algoritmo (Koza, 1992). Estos, además de depender de la aplicación en particular, dependen en gran medida del diseñador. Por lo que para un mismo problema hay muchas formas de definirlos.

1. **Conjunto de terminales:** son las entradas al programa o expresión matemática, estos son las hojas de los árboles. En el caso de la figura 2.6, el conjunto de terminales son X, Y y Z.

2. **Conjunto de funciones:** corresponde al conjunto de funciones que operan sobre otras funciones y el conjunto de terminales. En el caso de la figura 2.6, el conjunto de funciones es $CF = \{+, -, *, \%\}$. Los elementos de este conjunto son los que conforman los nodos internos de los individuos (árboles).

3. **Función de fitness:** si se hace una comparación con la naturaleza, la función de fitness es la probabilidad que tiene un individuo de llegar a la edad de reproducción y reproducirse. En algoritmos genéticos el fitness es representado generalmente como una expresión matemática de tal manera que se pueda utilizar para controlar aplicación de los operadores genéticos y modificar las estructuras de la población. Esta función depende del problema y del enfoque que se le esté dando. No guarda relación con la representación de los individuos, en este caso árboles. La función de fitness es aplicada a los individuos una vez estos son interpretados, o sea, ya no están representados por árboles.

4. **Parámetros de control:** los parámetros de control son típicamente: el número de individuos por generación y el máximo de iteraciones o generaciones. Estos parámetros dependen en gran medida de la función de fitness. Si esta función es muy costosa computacionalmente ninguno de estos dos parámetros podrá ser muy elevado o el algoritmo podría demorar mucho o simplemente la capacidad computacional no sería suficiente.

5. **Criterio de término:** este criterio es aquel necesario para saber cuándo terminar con el algoritmo y de qué forma seleccionar el mejor resultado. Existen muchos criterios entre los cuales están un máximo de iteraciones, un valor de fitness determinado, una cota en el error, etc. En cuanto a la forma de seleccionar el resultado, el método más usado es seleccionar al individuo con mejor fitness.

Estos son los pasos principales para poder iniciar el algoritmo de programación genética. Además de estos, existen parámetros que se deben fijar para poder correr el algoritmo.

2.3.3.2 Parámetros

Inicialización de la población: Los parámetros en esta etapa son la función que define como se inicializa la población y la profundidad máxima que pueden tener los arboles en la población inicial.

Hay 3 métodos que son los más utilizados para inicializar la población (Koza, 1992):

- Método *full*: se construyen árboles perfectamente balanceados aleatoriamente. Esto quiere decir que cada árbol alcanzar la máxima profundidad en todas sus ramas.

- Método *grow*: se construyen los árboles seleccionando de forma aleatoria si el nodo será una función o un terminal. Esto se aplica en todos los nodos menos en aquellos que alcanzan la profundidad máxima.
- Método *ramped half-half*: se inicializa igual número de individuos para cada valor de profundidad desde 2 hasta la profundidad máxima. Para cada nivel de profundidad, la mitad de los individuos es creado utilizando el método *full* y la otra mitad con el método *grow*.

Tamaño de arboles: Se define la profundidad máxima de los árboles en las demás generaciones-

Operadores: Se fijan los operadores genéticos y sus probabilidades de ocurrencia.

Selección: Se especifica el tipo de selección para aplicar los operadores genéticos.

Existe una gran variedad de métodos de selección entre los que se encuentran:

- Ruleta: este método actúa como una ruleta donde cada individuo tiene una porción proporcional a su valor de fitness.
- Por torneo: se escogen dos individuos de forma aleatoria y se elige el de mejor fitness.
- *lexicographic parsimony pressure* (LPP): es igual a selección por torneo. La diferencia es que ante dos individuos de igual fitness, *LPP* escoge como mejor al que tenga menor tamaño, esto ayuda a evitar el *bloat* (Luke & Panait, 2002).

Datos: Se pueden cargar los datos: los ejemplos y los resultados. Los ejemplos son los puntos para los que se conoce su salida. En el caso de una regresión simbólica donde se busca un $f(x)$, los ejemplos serían los valores de x para los que se tiene el valor de la salida $f(x)$. Los resultados son los valores de la salida $f(x)$.

Sobrevivencia: Se escoge que porcentaje de nuevos individuos son necesarios para pasar a la siguiente generación, y el criterio para escoger a los individuos de la siguiente generación.

Entre los criterios para escoger los individuos que pasan a la siguiente población se encuentran:

- Reemplazo: la población es reemplazada por completo por los hijos, incluso si estos tienen peor fitness que sus padres.
- Elitismo total: los individuos son seleccionados por su fitness independiente si son hijos o padres.
- Medio elitismo: la mejor mitad de padres y la de hijos pasan a la siguiente generación.
- *keep best:* consiste en elegir al mejor individuo dentro de una familia (padre(s) y su(s) respectivo(s) hijo(s)). Este método ha probado ser de ayuda en la mantención de la diversidad de la población (Wilese & Goodwin, s.f.) (Gupta & Ghafir, 2012).

En la figura 2.8 se muestra un diagrama de bloques de las etapas del algoritmo de programación genética y los distintos conjuntos de parámetros asociados a cada etapa.

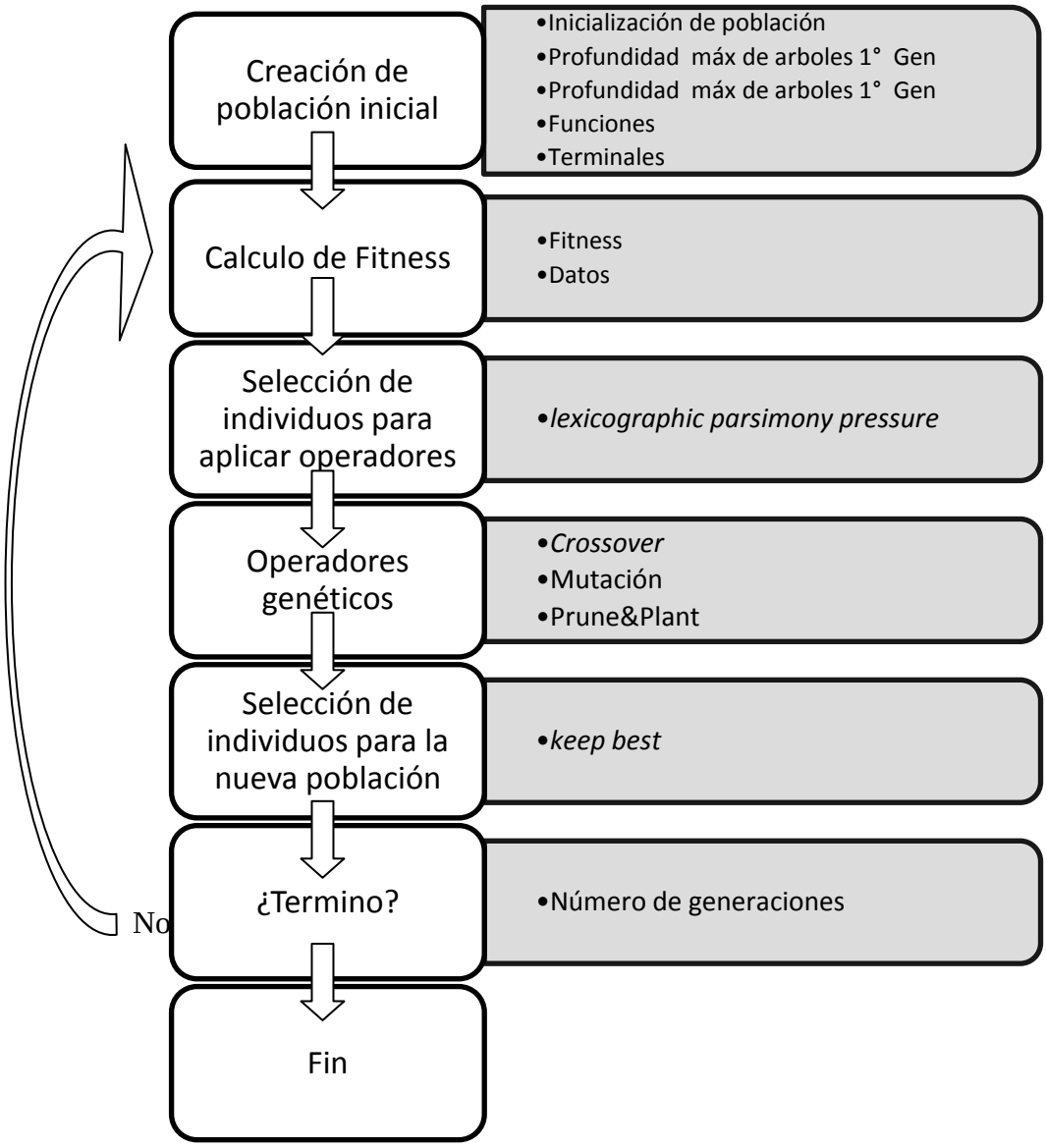

Figura 2.8 Diagrama de bloques del algoritmo de programación genética junto con sus parámetros asociados a cada etapa.

2.3.3.3 Bloat

Un problema importante en este algoritmo es que debido a la representación de los datos, los arboles a medida que pasan las generaciones pueden crecer indefinidamente lo que dificulta mucho su análisis. A este efecto se le denomina *bloat*. Una forma de evitarlo es agregando al algoritmo una restricción de un número máximo de profundidad de los árboles (Koza, 1992), de esta forma se puede controlar el tamaño de éstos. Otro método utilizado para este fin es la inclusión del operador Prune&Plant (podar y plantar) en el algoritmo genético.

2.3.3.3.1 Prune&Plant

El operador genético *Prune&Plant* está basado en la agricultura donde una planta se poda y los trozos podados son plantados para generar nuevas plantas. De la misma forma, en programación genética, este operador corta un individuo de profundidad igual o mayor a la profundidad máxima, en un nodo aleatorio y deja ambos árboles resultantes como individuos. Así, de un solo padre salen dos hijos como se muestra en la figura 2.9.

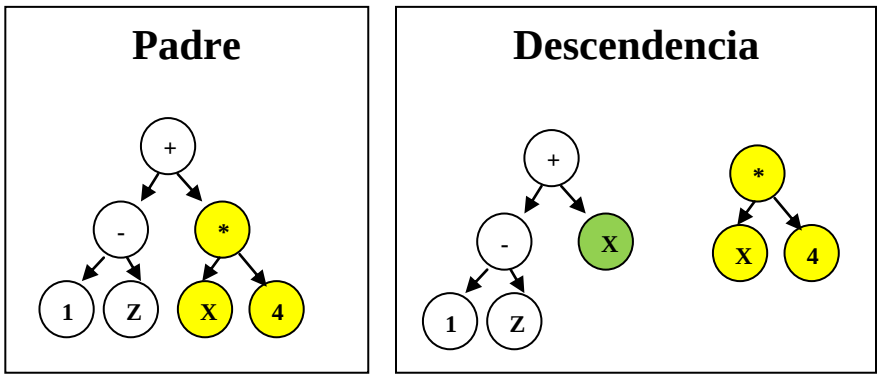

Figura 2.9 Operador *Prune&Plant* en un árbol

El nodo donde es separado el árbol padre se rellena por un elemento aleatoriamente. En el caso de la figura 2.9 al podar el árbol en '*', este nodo fue reemplazado por el terminal 'X' en el primer hijo.

Este operador además de evitar el efecto de *bloat* ayuda a encontrar buenas soluciones de manera más rápida y tiene una serie de ventajas ante otros métodos típicos de control de *bloat*, los que se muestran en (Alfaro-Cid, et al., 2008) y (Alfaro-Cid, et al., 2010).

Otro asunto no menor es la anulación de subárboles, esto quiere decir que puede haber un segmento de un árbol que al evaluarlo no sea de valor en la solución. Por ejemplo, en una expresión matemática puede darse el caso que se esté sumando un cero, y puede que ese cero este representado por un subárbol muy grande, lo que le quita posibilidades de evolucionar correctamente a dicho individuo además de ocupar recursos computacionales en exceso. Para evitar este fenómeno uno de los métodos más usados es la simplificación numérica que es explicada y comparada con otros métodos en (Kinzett, 2010) y (Kinzett, et al., 2009) .

2.4 Modelos sustitutos

En la optimización con algoritmos genéticos muchas veces los simuladores usados para evaluar la función de fitness son muy costos computacionalmente, lo que se traduce en tiempos prolongados para cada simulación. Para suplir esta falencia que muchas veces hace algunas optimizaciones impracticables, se construyen modelos sustitutos los que son una aproximación de la función de fitness original perdiendo precisión, pero ganando tiempo ya que la idea es que el tiempo por

evaluación de modelos sustitutos sea sustancialmente menor al modelo original. Solo en casos donde la función original sea muy costosa se justificará la construcción de modelos sustitutos.

Si bien existen muchos métodos para elaborar modelos sustitutos, la forma más básica se ilustra en el diagrama de flujo de la figura 2.10, donde se observa que el diseño de un modelo sustituto consta de 4 etapas principales que son:

1. **Diseño de experimentos:** esta primera parte consta del plan de muestreo para obtener los datos necesarios para la construcción del modelo.
2. **Simulación:** se ejecutan las simulaciones en el modelo costoso (software CDF o otro) de los valores de las variables de entrada que se obtuvieron en la etapa anterior.
3. **Construcción del modelo sustituto:** Lo principal en esta parte es definir qué modelo sustituto se va a usar (selección del modelo) y que parámetros tendrá dicho modelo (identificación de modelo).
4. **Validación:** Finalmente lo que se busca es determinar la capacidad de predicción del modelo.

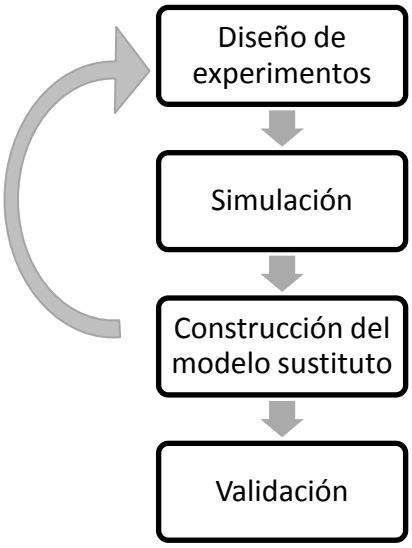

Figura 2.10 Etapas del diseño de un modelo sustituto

Un tipo de modelo sustituto consiste en la identificación de una función continua a partir de un set de datos limitado. Estos datos pueden ser valores exactos que provienen de evaluar una función u observaciones ruidosas que se obtienen a partir de experimentos reales. De todos modos, estos datos no pueden tener suficiente información como para construir un único modelo sustituto por lo tanto, existen muchos modelos que se ajustan a los datos.

Una forma de ilustrar como se puede obtener un buen modelo sustituto a partir de un conjunto de posibles modelos es con ayuda de teoría de regularización (ver (Tikhonov & Arsenin, 1997)) que impones restricciones a la estimación. En particular, se puede obtener un modelo sustituto resolviendo el siguiente problema de regularización de Tikhonov:

$$\min_{\hat{f}\epsilon S} H(\hat{f}) = \frac{1}{N_s}\sum_{i=1}^{N_s} L\left(f_i - \hat{f}(x^i)\right) + \alpha \int \left\|D^m \hat{f}\right\|_S dx \tag{2.5}$$

Donde $\hat{f}$ es un modelo sustituto, N_s es la cantidad de datos, L es la función de pérdida, f_i es la función costosa (que se está tratando de sustituir), S es el conjunto de modelos sustitutos en consideración, D^m representa la m-ésima derivada y α es un parámetro de regularización. En esta ecuación el segundo término representa una penalización, esto quiere decir que se premia a las funciones $\hat{f}$ más suaves. Respecto a la función de pérdida L, una de las más usuales es la función cuadrática, aunque también se ocupa la función lineal que es el valor absoluto del argumento, la función Huber, que es cuadrática para valores pequeños y lineal para los más grandes, o la función ε que es cero si el error esta en un rango determinado, y crece linealmente si no lo está. En la figura 2.11 se muestran los gráficos de estas 4 funciones de pérdida.

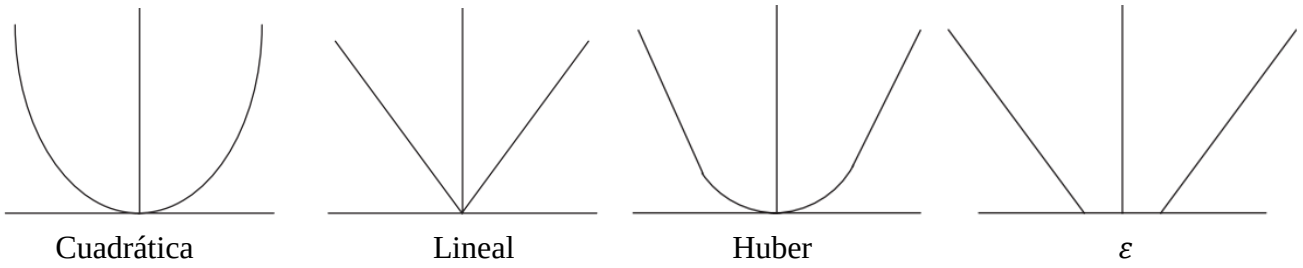

Figura 2.11 Funciones de pérdida usualmente utilizadas

Los procesos para identificar el parámetro α son típicamente basados en estimaciones de generalización de error, lo que se hace en la etapa de validación y será discutido en la sección 2.4.3.

2.4.1 Diseño de experimentos

El objetivo de esta etapa es muestrear el espacio de variables de diseño. Se busca un método no aleatorio que no dependa del modelo que muestree el espacio de tal manera que con pocos datos se pueda obtener una buena representación de éste.

Se han propuesto diversos métodos para el diseño de experimentos, en particular, existen dos muy utilizados que son arreglos ortogonales (Hedayat, et al., 1999) y "*Latin Hypercube*" (McKay, et al., 1979). Aunque también existen muchos autores que han intentado contrarrestar las falencias de estos métodos utilizando diversos enfoques como se muestra en (AB, 1994), (Leary, et al., 2003) y (Jones, et al., 1998). A continuación se detalla el muestreo *Latin Hypercube*.

2.4.1.1 Latin Hypercube

Latin Hypercube es una técnica de muestreo que asegura que todas las porciones del espacio de variables de diseño serán muestreadas. Esto lo hace suponiendo que cada variable de entrada tiene todas las porciones de su distribución representada por valores de entrada.

Si se quiere un conjunto de muestras de tamaño N_s, los rangos de cada variable de entrada se dividen en N_s intervalos de igual probabilidad $1/N_s$ y se saca una muestra de cada intervalo para cada variable sin tomar dos veces una muestra de un mismo intervalo. Un ejemplo de muestreo *Latin Hypercube* de 6 muestras para un problema de dos variables de entrada se muestra en la figura 2.12.

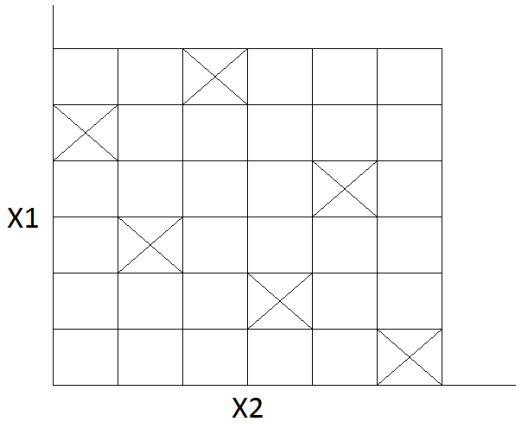

Figura 2.12 Muestreo *Latin Hypercube* de 6 muestras para un problema de dos variables de entrada.

La elección de diseño de experimentos es crucial en la construcción de modelos sustitutos ya que de él dependen en gran medida los resultados finales. Esto podría motivar a la optimización de un conjunto muestreado pero esto es muchas veces impracticable debido al tiempo que demora.

2.4.2 Construcción del modelo sustituto

Para construir el modelo sustituto a partir de los datos que se obtienen simulando aquellos puntos que da el diseño de experimentos, existen alternativas tanto paramétricas como no paramétricas. Dentro de los modelos paramétricos se encuentra la regresión polinomial y Kriging, entre otros, mientras que un ejemplo de los modelos no paramétricos son las funciones de base radial. La diferencia radica en que los primeros asumen la forma general de la función y ajustan el modelo por medio de parámetros mientras que los segundos no asumen una forma a priori.

2.4.3 Validación

La validación del modelo sustituto es muy importante ya que verifica que aproxime al modelo original y que no se ajuste demasiado a los datos (overfitting). Hay dos métodos muy utilizados que son la separación de muestras y la validación cruzada.

2.4.3.1 Separación de muestras

La separación de muestras consiste en dividir el conjunto de muestras obtenido en el proceso de diseño de experimentos en tres subconjuntos: entrenamiento, validación y prueba. El subconjunto de entrenamiento se utiliza para entrenar el modelo y obtener un conjunto de soluciones. El subconjunto de validación por su parte sirve para escoger la mejor solución de las entregadas al entrenar el sistema. Por último, el subconjunto de prueba es utilizado para obtener resultados de la mejor solución. Esto se hace con el fin de garantizar que las medidas de desempeño de las soluciones no son dependientes de un subconjunto en particular.

2.4.3.2 Validación cruzada

Es muy utilizado cuando el número de muestras disponibles es pequeño. Se divide el conjunto de muestras en k subconjuntos de igual tamaño. Un modelo sustituto es construido k veces, cada vez dejando un subconjunto distinto como conjunto de validación, por lo tanto, el proceso que se hace en el método de separación de muestras es repetido k veces. El error final es el promedio de los

errores que se calcularon en cada iteración. La ventaja de este método es que entrega un sesgo muy pequeño y la varianza es menor si se compara con el de separación de muestras debido a que cada muestra está en el conjunto de prueba una vez y $k-1$ veces en el de entrenamiento independiente en cómo se escojan estos conjuntos. La dificultad es que se deben construir k modelos sustitutos en vez de uno solo como en el caso anterior.

Si se está eligiendo el mejor modelo por regularización de Tikhonov (ecuación 2.5), uno de los métodos más usados para calcular α es un caso especial de validación cruzada denominada *"leave one out"* que consiste en una validación cruzada con una cantidad de subconjuntos igual a la cantidad de muestras. De esta forma, se construirán tantos modelos sustitutos como muestras hayan y se probará la eficacia de cada modelo con una sola muestra.

2.5 Aplicaciones de modelos sustitutos

Softwares como *Modern Computational Structural Mechanics, Computational Electro-magnetics y Computational Fluid Dynamics* son simuladores que han probado ser de gran utilidad debido a su precisión. Estos programas juegan un rol clave en el proceso de diseño ya que ayudan a diseñadores y científicos a validar nuevos diseños y estudiar el efecto que produce cambiar un parámetro clave en un producto o proceso. Sin embargo, estas simulaciones son muy costosas y poco prácticas en la optimización con algoritmos evolutivos ya que demandan tiempos prohibitivos.

Debido a lo anteriormente expuesto, son muchas las aplicaciones que utilizan modelos sustitutos para reducir el costo computacional y tiempo que toma una simulación, entre éstas se pueden destacar aplicaciones en aerodinámica, termodinámica, electrónica y electricidad. A continuación se exponen algunas de las aplicaciones en cada una de estas áreas.

2.5.1 Aerodinámica

En (Oksuz & Akmandor, 2010) se hace una optimización multi-objetivo de aspas de turbina usando algoritmos genéticos y un modelo sustituto para reemplazar un CFD. En (Mengistu & Ghaly, 2008) se hace lo mismo pero usando redes neuronales para generar el modelo sustituto. Por otro lado, en (Diaz-Casas, et al., 2013) la optimización con modelos sustitutos utilizada en el diseño del eje de una turbina de viento tomando en cuenta tanto aspectos aerodinámicos como estructurales.

En (Nghia, et al., 2013) se utiliza un algoritmo para escoger el mejor modelo sustituto en la confección de la parte posterior de un auto.

2.5.2 Termodinámica

En (Ranut, et al., 2014) se comparan dos algoritmos genéticos para una optimización de un intercambiador de calor, uno de estos incorpora modelos sustitutos por lo que logra una gran reducción en el tiempo de optimización. En (Khatir, et al., 2013) se presenta el algoritmo de optimización multi-objetivo de un horno para la cocción de pan utilizando un modelo que sustituye al CFD ANSYS.

2.5.3 Electrónica

En (Song, et al., 2014) y (Kim & Kim, 2014) si bien el modelo sustituto se aplica a un problema de refrigeración, la aplicación consiste en el enfriamiento de dispositivos electrónicos. El primer caso es una optimización para el diseño de un sistema de refrigeración para dispositivos en general donde utilizan modelos de Kriging para reemplazar un CFD. El segundo es la optimización en el diseño de un sistema de refrigeración de LEDs de alta potencia.

En (Kotti, et al., 2014) se hace una optimización multi-objetivo para encontrar el mejor desempeño de inductores planos para ser utilizados en dispositivos de comunicación inalámbrica.

2.5.4 Eléctrica

La optimización multi-objetivo usando algoritmos genéticos del rendimiento de accionamientos eléctricos es abordada en (Zavoianu, et al., 2013). Se optimiza usando un algoritmo que incluye un procedimiento que genera modelos sustitutos sobre la marcha usando redes neuronales. Utilizando este enfoque se puede llegar a una reducción del 46-72% en el tiempo de optimización.

2.5.5 Otras Aplicaciones

En (Nguyen, et al., 2014) los modelos sustitutos son usados en métodos de optimización aplicados al análisis del rendimiento de construcciones. Mientras que en (Koziel & Ogurtsov, 2013) estos son usados para el diseño de antenas.

En (Patakar & Spalding, 1972) se desarrolla un algoritmo de optimización multi-objetivo con modelos sustitutos para la gestión de aguas subterráneas en áreas costeras. El objetivo de la optimización es minimizar tanto la parte económica como el impacto ambiental sujeto a satisfacer la demanda de agua.

2.6 Mecánica de fluidos computacional (CFD)

Computational Fluid Dynamics es un software que simula el comportamiento de fluidos newtonianos. Dichos fluidos son representados por las ecuaciones de Navier-Stokes. Para resolverlas se utilizan ecuaciones diferenciales parciales no lineales de continuidad, momentum y energía que explican los fenómenos de flujo de fluido, transferencia de calor y turbulencia. Estas ecuaciones se basan en tres relaciones físicas fundamentales: conservación de masa, segunda ley de Newton y conservación de energía (Wendt, 2009).

Conservación de masa: depende de la densidad del fluido ρ y el vector de velocidad $\vec{u}$.

$$\frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0 \tag{2.6}$$

Conservación de momentum: se calculan por eje y dependen de las velocidades u , v y ω (velocidades en los ejes x, y, z respectivamente), la densidad del fluido ρ , la presión p y las fuentes de momentum por eje, denotadas S_m.

$$\frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u \vec{u}) = \frac{\partial p}{\partial x} + \nabla \cdot (u \nabla \mathrm{u}) + S_{m_x} \tag{2.7}$$

(2.8) $$\frac{\partial(\rho v)}{\partial t} + \nabla \cdot (\rho v \vec{v}) = \frac{\partial p}{\partial y} + \nabla \cdot (\mathrm{v}\nabla \mathrm{v}) + S_{m_y}$$

(2.9) $$\frac{\partial(\rho \omega)}{\partial t} + \nabla \cdot (\rho \omega \vec{\omega}) = \frac{\partial p}{\partial z} + \nabla \cdot (\omega \nabla \omega) + S_{m_z}$$

Conservación de energía: depende de la energía interna del fluido E_i, la conductividad térmica k, la temperatura T, la presión, la velocidad, la generación interna de energía S_i y la disipación Φ.

(2.10) $$\frac{\partial(\rho E_i)}{\partial t} + \nabla \cdot (\rho E_i \vec{u}) = -p\nabla \cdot \vec{u} + \nabla \cdot (k\nabla \mathrm{T}) + \Phi + \mathrm{S_i}$$

Para completar el sistema, se agregan las ecuaciones de estado para gases ideales compresibles:

(2.11) $$p = \rho R T$$

(2.12) $$E_i = C_v T$$

Donde R es la constante universal de los gases ideales y C_v es la capacidad calorífica a volumen constante.

2.7 Modelo fenomenológico

El modelo fenomenológico utilizado en la presente memoria (Reyes, 2014) busca obtener la temperatura, densidad, presión y velocidad del fluido (aire) y la temperatura de las celdas centrales de un banco de baterías organizado como grilla escalonada. Se toman como datos de entrada el espaciado entre celdas, el flujo volumétrico y la corriente de uso de las celdas. Para ello se encuentran relaciones de entrada-salida en un bloque consistente en una celda central y 4 en las esquinas como se ilustra en la figura 2.13. Concatenando bloques de este tipo se puede construir un banco de baterías de cualquier tamaño. En este bloque, las columnas con 2 celdas (líneas rojas en la figura 2.13) son denominadas columnas de fluido mientras que la columna que contiene la celda central es denominada columna de celda.

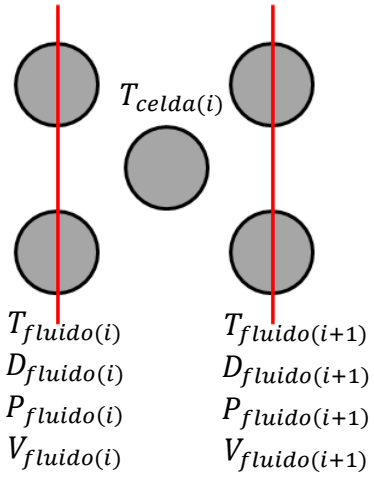

Figura 2.13 Bloque utilizado por el modelo fenomenológico

Este modelo se basa en los siguientes cinco principios:

1. Caída de presión en intercambiadores de calor para bancos de tubos (Zukauskas, 1987) :

(2.13) $$\Delta P = \frac{N_l f \chi \rho V^2}{2}$$

Donde ΔP es la diferencia de presión en $[Pa]$, N_l el número de celdas enfrentadas al fluido, f el coeficiente de fricción, χ el factor de forma, ambos adimensionales, ρ la densidad del fluido en $[kg/m^3]$ y V la velocidad del fluido en $[m/s]$.

2. Gas ideal:

(2.14) $$P = \rho R T$$

Donde P es la presión en $[Pa]$, ρ la densidad del fluido en $[kg/m^3]$, R la constante de gases ideales $[J/KgK]$ y T la temperatura en [°C].

3. Conservación de momento:

(2.15) $$P_i A_i + \dot{m}_i V_i^2 = P_{i+1} A_{i+1} + \dot{m}_{i+1} V_{i+1}^2 + F_{ext}$$

Donde P es la presión en$[Pa]$, A el área por donde pasa el flujo en $[m^2]$, $\dot{m}$ el flujo másico$[Kg/s]$, V la velocidad en $[m/s]$, F_{ext} las fuerzas externas que en este caso es la fuerza de arrastre en $[N]$ y el índice i es el lugar en el espacio donde se está evaluando la variable.

4. Conservación de energía:

(2.16) $$\dot{m}\left(c_p \Delta T + \frac{V^2}{2} + g\Delta z\right) = Q_{fluido} + W_{eje}$$

Donde $\dot{m}$ es el flujo másico en $[Kg/s]$, ΔT la diferencia de temperatura en $[K]$, c_p es el calor específico del fluido en $[J/KgK]$, V la velocidad del fluido en $[m/s]$, g la aceleración de gravedad en $[m/s^2]$, Δz la diferencia de altura en $[m]$, Q_{fluido} la energía transferida hacia el fluido en $[J]$ y W_{eje} el trabajo ejercido hacia el fluido en $[J]$.

5. Transferencia de energía:

(2.17) $$\dot{q} = -k_{celda} \Delta T$$

(2.18) $$\dot{q} = h_f \Delta T$$

Donde $\dot{q}$ es el calor transferido por unidad de área en $[J/m^2]$, ΔT la diferencia de temperatura en $[°K]$, k_{celda} el coeficiente conductivo de la celda y h_f el coeficiente convectivo del fluido. Estos dos últimos expresados en $[W/mK]$.

2.7.1 Calculo de variables de salida

Cada variable de salida del modelo se calcula mediante una ecuación fenomenológica, los parámetros de los que dependen provienen de relaciones obtenidas experimentalmente como el factor de fricción o la correlación Nusselt-Reynolds, y otros son obtenidos calculando sus valores

teóricos como el número de Reynolds. Las expresiones para las variables de salida del modelo fenomenológico se presentan a continuación:

Presión del fluido: la expresión para obtener la presión se basa en la ecuación 2.13 que corresponde a la caída de presión en intercambiadores de calor. Ésta obtiene la presión en el punto i dado el punto i+1, esto es porque la condición de borde se tiene en la salida del banco (i es 1 en la entrada del banco). Por lo tanto queda de la siguiente manera:

(2. 19) $$P(i) = -\frac{N_l f \chi \rho V(i)^2}{2} + P(i+1)$$

Donde $P(i)$ y $P(i+1)$ son las presiones del aire en la columna de fluido i e $i+1$ respectivamente (en $[Pa]$), N_l el número de celdas enfrentadas al fluido, f el coeficiente de fricción, χ el factor de forma, ambos adimensionales, ρ la densidad del fluido en $[kg/m^3]$ y $V(i)$ la velocidad del fluido en la columna de fluido i en $[m/s]$.

Velocidad del fluido: la velocidad es obtenida mediante la ecuación 2.20 que está basada en la ecuación 2.15 que es la conservación de momentum. En este caso se calcula la velocidad en i+1 dada la velocidad en i ya que la condición de borde es al comienzo del banco. La expresión es la siguiente:

(2.20) $$V(i+1) = \sqrt{\frac{(A\Delta P - F_{ext})}{\dot{m}} + V(i)^2}$$

Donde ΔP es la diferencia de presión en $[Pa]$, A el área por donde pasa el flujo en $[m^2]$, $\dot{m}$ el flujo másico$[Kg/s]$, V la velocidad en $[m/s]$, F_{ext} las fuerzas externas que en este caso es la fuerza de arrastre en $[N]$ y el argumento i corresponde a la columna de fluido.

Temperatura del fluido: esta temperatura es calculada usando la ecuación 2.21 que se obtiene a partir de la ecuación 2.16 que es la conservación de energía. Se calcula la temperatura en i+1 en función de la temperatura en i ya que la condición de borde se tiene al comienzo del banco. La expresión con la que se calcula la temperatura es la siguiente:

(2.21) $$T_f(i+1) = \frac{\left(\frac{q}{\dot{m}} - \frac{V(i)^2}{2}\right)}{c_p} + T_f(i)$$

Donde $\dot{m}$ es el flujo másico en $[Kg/s]$, T_f la temperatura del fluido en $[K]$, c_p es el calor específico del fluido en $[J/KgK]$, V la velocidad del fluido en $[m/s]$ y q la energía transferida hacia el fluido en $[J]$.

Densidad del fluido: La densidad del aire se obtiene mediante un procedimiento que consiste llevar el fluido a la velocidad del sonido, calcular la presión y la temperatura a esa velocidad para finalmente obtener la diferencia de densidad con la ecuación 2.14 que es la ley de los gases ideales y se ocupa la condición de borde en el banco para volver al rango de densidad correspondiente. Para mayor información consultar el libro (Fox, et al., 2009).

Temperatura de la celda: La temperatura en la celda es la única variable que no cuenta con una condición de borde, pero esta no es necesaria ya que se puede expresar en función de la

temperatura en el fluido. Para esto se utiliza la ecuación 2.18 que corresponde a la transmisión de energía, en ésta se despeja la temperatura de celda obteniendo la expresión 2.22.

(2.22) $$T_c(i) = \frac{\dot{q}}{A_{sup} h_f} + \frac{T_f(i) + T_f(i+1)}{2}$$

Donde $\dot{q}$ es el calor transferido por unidad de área en $[J/m^2]$, A_{sup} el área de una celda sin contar sus caras circulares, T_c la temperatura de la celda en $[°K]$, T_f la temperatura del fluido en $[°K]$ y h_f el coeficiente convectivo del fluido en $[W/mK]$ y el argumento i es el bloque donde se está evaluando la variable.

2.7.2 Parámetros de interés

A continuación se presentan los parámetros a ajustar, las variables de salida donde intervienen, y como son calculados en el modelo fenomenológico.

2.7.2.1 Número de Nusselt

En la ecuación 2.23 se muestra la definición del número de Nusselt.

(2.23) $$Nu = \frac{h_f D}{k_f}$$

Donde h_f es el coeficiente convectivo del fluido expresado en $[W/mK]$, D el largo característico del objeto por donde circula el fluido (en [m]) y k_f la conductividad del fluido en $[W/mK]$. Esta expresión es utilizada para determinar el valor de h_f para calcular la temperatura de la celda central con la expresión 2.22. Para poder obtener este valor es necesario conocer el número de Nusselt, por lo que se recurre a la correlación Nusselt-Reynolds que es una expresión obtenida experimentalmente por lo que depende de la aplicación.

Esta correlación fue obtenida en (Zukauskas, 1987) en un estudio del flujo de aire a través de bancos de tubos escalonados:

(2.24) $$Nu = \begin{cases} \left(0.9 Re^{0.4} Pr^{0.36} \left(\frac{Pr}{Pr_s}\right)^{0.25}\right) & para\ 10 < Re < 100 \\ \left(0.51 Re^{0.5} Pr^{0.36} \left(\frac{Pr}{Pr_s}\right)^{0.25}\right) & para\ 100 < Re < 1000 \\ \left(0.9 Re^{0.4} Pr^{0.36} \left(\frac{Pr}{Pr_s}\right)^{0.25}\right) & para\ 1000 < Re < 2 \cdot 10^{15} \end{cases}$$

Donde Pr es el número adimensional de Prandtl y Pr_s es el valor que toma este número para la superficie del sólido. Tal como las dos expresiones anteriores, Re es el número de Reynolds y Nu es el número de Nusselt (ambos adimensionales).

2.7.2.2 Factor de fricción y factor de forma

El factor de forma y el factor de fricción se utilizan para calcular el valor de la presión del aire como se desprende de la ecuación 2.19. Para obtener estos factores se utilizaron los resultados obtenidos experimentalmente en (Zukauskas, 1972). Estos valores son aproximados por la curva 2.25 dependiente del número de Reynolds.

(2.25) $$factor\ de\ fricción * factor\ de\ forma = A_{(S)} \cdot Re^{B(S)}$$

En la expresión 2.25 se muestra la aproximación que incluye a ambos factores. Los parámetros A y B dependen de la separación entre las celdas.

2.7.2.3 Coeficiente de arrastre

Para calcular la velocidad del fluido es necesario conocer las fuerzas externas que se ejercen sobre este como se ve en la expresión 2.20. En esta aplicación la fuerza externa corresponde a la fuerza de arrastre que se puede obtener con la ecuación 2.26.

(2.26) $$F_{arrastre} = \frac{1}{2} A \rho V^2 C_{arrastre}$$

Donde $F_{arrastre}$es la fuerza de arrastre en [N], A es el área por donde pasa el fluido en $[m^2]$, ρ es la densidad del fluido en $[kg/m^3]$, V es su velocidad en $[m/s]$ y $C_{arrastre}$ es el coeficiente de arrastre del cuerpo por donde pasa el fluido (adimensional). Este último se obtiene a partir de valores experimentales hechos por (Zukauskas, 1972) donde se estudió el coeficiente de arrastre del aire al pasar por un cilindro y su relación con el número de Reynolds.

Capítulo 3

Metodología e Implementación

3.1 Herramientas

En la presente memoria se utiliza el toolbox de MATLAB GPlab (Silva, 2014) junto con una serie de funciones personalizadas para la implementación de programación genética, el modelo fenomenológico también es un conjunto de funciones en MATLAB, mientras que los datos de entrenamiento, validación y prueba fueron obtenidos de ANSYS que es un software que cae en la categoría de *Computational Fluid-Dynamics*.

3.2 Modelación del problema

Uno de los factores que más afectan al prematuro envejecimiento de las baterías de ion-litio en aplicaciones de alta potencia (vehículos eléctricos o respaldos de sistemas eléctricos de potencia), son las altas temperaturas alcanzadas durante su uso, es por eso que es de vital importancia diseñar estas baterías para que su refrigeración sea óptima.

Para realizar este diseño, una herramienta posible son los algoritmos evolutivos donde es necesario realizar pruebas en laboratorio o simulaciones de una gran cantidad de configuraciones de bancos, por lo que el costo computacional es importante a la hora de escoger como obtener estos datos. Si bien las simulaciones en el CFD ANSYS demoran menos tiempo que las pruebas de laboratorio, y obtienen valores muy parecidos a la realidad, aun así requieren de un tiempo considerable. Dependiendo de la simulación pueden ser desde un par de minuto hasta varias decenas de minutos por simulación, y dado que los algoritmos evolutivos necesitan correr cientos o miles de simulaciones por generación, el tiempo total aún es demasiado alto. En situaciones como ésta se recurre a modelos sustitutos que puedan aproximar el software CFD para un subconjunto de casos.

Una configuración típica de banco de baterías es la grilla escalonada. Se trabajará con un modelo sustituto que representa el comportamiento térmico de un banco de baterías en grilla escalonada. El modelo basado en ecuaciones fenomenológicas descritas en la sección 2.7 entrega los valores de velocidad, densidad, temperatura y presión del fluido y la temperatura de las celdas centrales para distintos valores de flujo de entrada, separación entre las celdas y corriente que fluye por estas.

El problema del modelo fenomenológico es que el error con respecto al software ANSYS es muy grande para algunas variables de salida como la presión. Esto se debe en parte a que algunos parámetros del modelo fenomenológico se basan en datos empíricos que no se ajustan bien a las condiciones de un banco de baterías.

El objetivo general de la presente memoria es ajustar dicho modelo, encontrando mediante programación genética expresiones que aproximen de mejor manera el factor de forma y factor de fricción que son dos parámetros utilizados en el cálculo de la presión del fluido. Además, se propone ajustar el coeficiente de arrastre, que se utiliza para calcular la velocidad del fluido por medio de la fuerza de arrastre y finalmente la correlación Nusselt-Reynolds que es usada para obtener la temperatura en la celda central. Todas estas expresiones han sido obtenidas a partir de datos experimentales por lo que tienen un error asociado.

La programación genética fue escogida por ser una herramienta versátil que puede ser utilizada en regresiones simbólicas y tiene la ventaja de no suponer a priori la forma de la expresión.

3.3 Primeras pruebas con Programación Genética

Para familiarizarse con el algoritmo de programación genética, lo primero que se hizo fue abordar problemas simples de regresión simbólica, donde el objetivo fue identificar funciones polinomiales a partir de datos de entrada y salida.

Se corrió el algoritmo varias veces variando distintos parámetros para identificar su relevancia. Esto se hizo para comprender como afectan algunos parámetros en los resultados obtenidos por el algoritmo y los tiempos que demora. Se varió el número de individuos por generación, el máximo nivel permitido en árboles, los distintos métodos de selección entre otras cosas.

Además se probaron parámetros del toolbox (Silva, 2014) que según éste son métodos experimentales, ya que se ha demostrado una mejora en solo algunas aplicaciones. Estos parámetros fueron descartados al no mostrar una mejora sustancial y utilizar mucho tiempo en su procesamiento.

Luego, con el fin de abordar un problema más complejo y relacionado con el objetivo final, se intentó encontrar un modelo equivalente al modelo fenomenológico teniendo como salida solo la temperatura de la celda central y como entrada el flujo, la separación entre las celdas y la corriente que pasa por ellas. Estas entradas corresponden a las del modelo fenomenológico original por lo que los rangos de estas variables son los dados por el modelo.

3.3.1 Regresión simbólica

Como se mencionó anteriormente, los primeros problemas abordados fueron de regresión simbólica. En particular, la primera función a la que se intentó llegar fue la expresión 3.1, utilizando 201 puntos equidistantes como las muestras conocidas de la función $f_1(x)$.

(3.1) $$f_1(x) = x^3 - x^2 + x$$

Con $x \in [-1,1]$. Para esto, los parámetros iniciales fueron:

- **Set de terminales:** x
- **Set de Funciones:** {+,-,*,^3,^2}
- **Función de fitness:** error cuadrático medio (MSE)
- **Parámetros de control:** 1000 individuos y 10 generaciones
- **Condición de termino:** máximo de generaciones

La lista completa de parámetros utilizados para estos experimentos se encuentra en el apéndice A.1.

Con esta configuración se logró el resultado como se muestra en la figura 3.1, donde el árbol representa fielmente la función objetivo

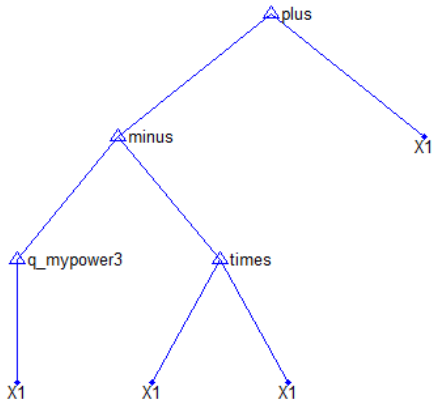

Figura 3.1 Árbol evolucionado con el algoritmo de programación genética para la función $x^3 - x^2 + x$

Para lograr esto también se corrió el algoritmo con un conjunto más grande de funciones agregando la división y exponencial, con lo que se llegaba a árboles equivalentes al de la figura 3.1 pero demorando más tiempo. Esto es de esperar ya que se tiene una mayor cantidad de combinaciones para crear los árboles. Además se variaron parámetros como el máximo nivel de profundidad de los árboles, pero al ser la función muy simple y sin ruido, se llegaba siempre al resultado correcto.

La siguiente función probada fue la mostrada en la función 3.2, utilizando 201 puntos equidistantes como las muestras conocidas de la función $f_2(x)$.

(3.2) $$f_2(x) = x^3 + 3x^2 - 4x + 6.$$

Con $x \in [-1,1]$. Este caso es más complejo ya que tiene constantes y hay que agregarlas al conjunto de terminales. Los parámetros iniciales fueron los siguientes:

- **Set de terminales:** {x, número aleatorio}
- **Set de Funciones:** {+,-,*,^3,^2}
- **Función de fitness:** MSE
- **Parámetros de control:** 1000 individuos
- **Condición de termino:** máximo de generaciones

La lista completa de parámetros utilizados para estos experimentos se encuentra en el apéndice A.2.

Para este caso se utilizaron más individuos por generación ya que la identificación de constantes es difícil en programación genética por lo que se requiere de la mayor diversidad posible. Por esta misma razón si bien se llega a funciones con error muy pequeño, esta función no es el objetivo. En la figura 3.2 se muestra mejor individuo obtenido hasta la generación 20, mientras que la expresión 3.3 corresponde a su expresión simplificada.

(3.3) $$f_{PG}(x) = 0.936\,\mathrm{x}^3 + 4.9\,\mathrm{x}^2 - 4.6\,\mathrm{x} + 6.4$$

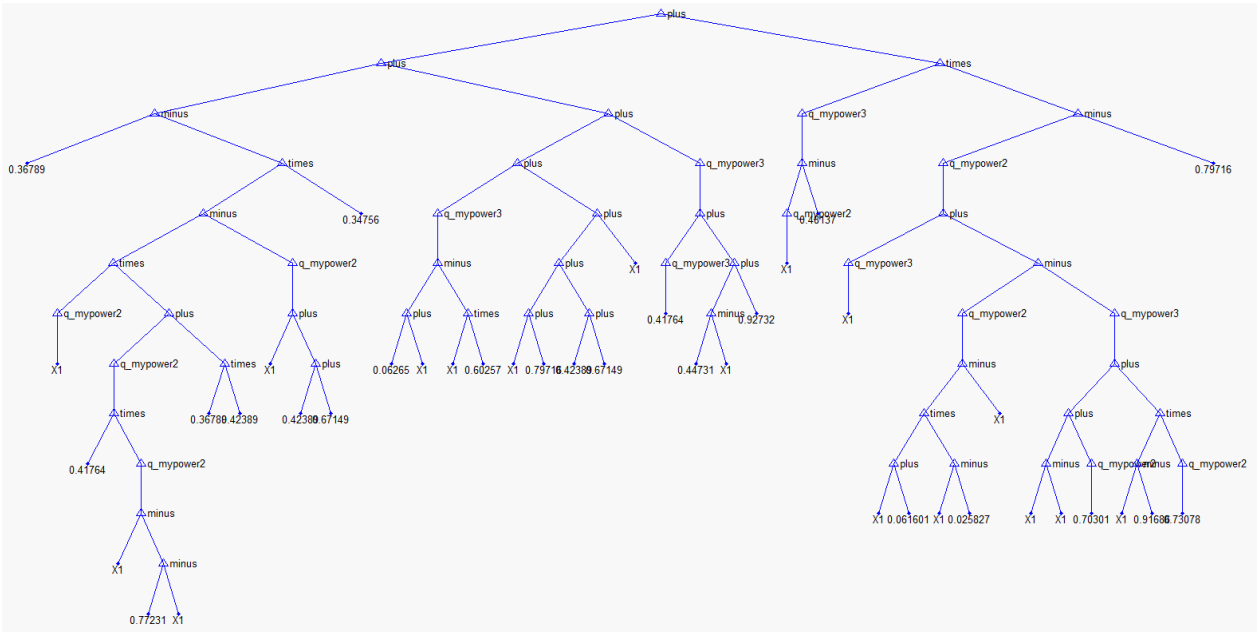

Figura 3.2 Árbol evolucionado con el algoritmo de programación genética para la función $x^3 + 3x^2 - 4x + 6$

A pesar de haber agregado solo constantes manteniendo el grado del polinomio y el número de variables, obtener una solución parecida a la función objetivo se vuelve muy complejo. La profundidad del mejor individuo creció de manera muy rápida llegando a 12 en la generación 6, manteniéndose en ese nivel hasta la solución de la expresión 3.3. De esta manera se comprueba que el *bloat* es un problema real, y que se presenta aun en problemas simples.

Algo que también se notó en este problema es que los arboles tienen partes que no son de utilidad, por ejemplo, en la figura 3.2 hay una resta que da cero, y sumas y restas a la variable de diseño de valores orden de $1 \cdot 10^{-3}$, los que son despreciables en comparación al rango de la variable.

3.3.2 Modelo fenomenológico

En este caso se trabajó con el modelo fenomenológico original. Utilizó un conjunto de entrenamiento de 500 datos obtenidos con *latin hypercube*, siendo las entradas el flujo, la separación entre las celdas y la corriente que pasa por ellas y como salida la temperatura de la primera celda. Los parámetros usados fueron:

- **Set de terminales:** {separación, corriente, flujo, números aleatorios}
- **Set de Funciones:** {+,-,*,/,^3,^2}
- **Función de fitness:** $MSE + \alpha\sqrt{nodos}$
- **Parámetros de control:** 2000 individuos
- **Condición de termino:** máximo de generaciones

La lista completa de parámetros utilizados para estos experimentos se encuentra en el apéndice A.3.

Se hicieron múltiples corridas de programación genética en ninguno de los casos se llego a un error lo suficientemente pequeño como para que el modelo fuera de utilidad. Sin embargo, el principal objetivo de este problema fue comprobar la relación de los resultados con la función de fitness utilizada.

Al utilizar la función MSE se observó que el fitness sigue decreciendo en la generación 70 donde el tiempo de la corrida del algoritmo genético asciende a 24 horas. Esto se ilustra en el gráfico de la figura 3.3

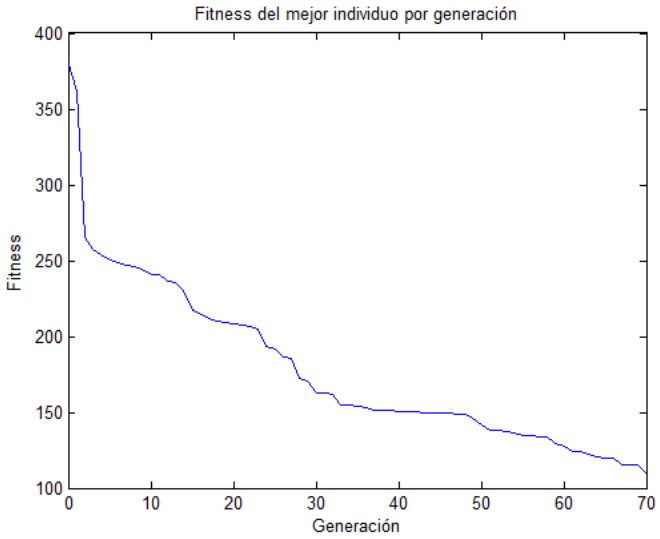

Figura 3.3 Valor del fitness del mejor individuo por generación para la función de fitness MSE

Junto con disminuir el fitness aumenta el tamaño del árbol del mejor individuo y su complejidad, llegando en esta última generación a 16 niveles y 245 nodos. Este crecimiento se debe a que en la función de fitness no se agrega ningún elemento que castigue a los árboles de gran tamaño. Es por eso que en la función de fitness se agregó un término de regularización por lo que la función de fitness resultante fue la que se muestra en la expresión 3.4

(3.4) $$fitness = \frac{1}{N}\sum_i \left(Y_i^{mod} - Y_i^{pg}\right)^2 + \alpha\sqrt{nodos}$$

Donde N es el total de puntos del conjunto de entrenamiento, Y_i^{mod} es el valor de la salida del modelo fenomenológico para el punto i, Y_i^{pg} el valor de la salida del modelo evolucionado con programación genética para el punto i, nodos es la cantidad de nodos del árbol del individuo en cuestión y α es un parámetro de ajuste.

Se corrió 6 veces el algoritmo variando en cada vez el valor de α, siendo estos 0, 0.5, 1,2 5 y 10. El límite de generaciones se estableció en 30. En el gráfico de la figura 3.4 se muestra la evolución del número de nodos que tiene el mejor individuo de cada generación en las corridas recién descritas.

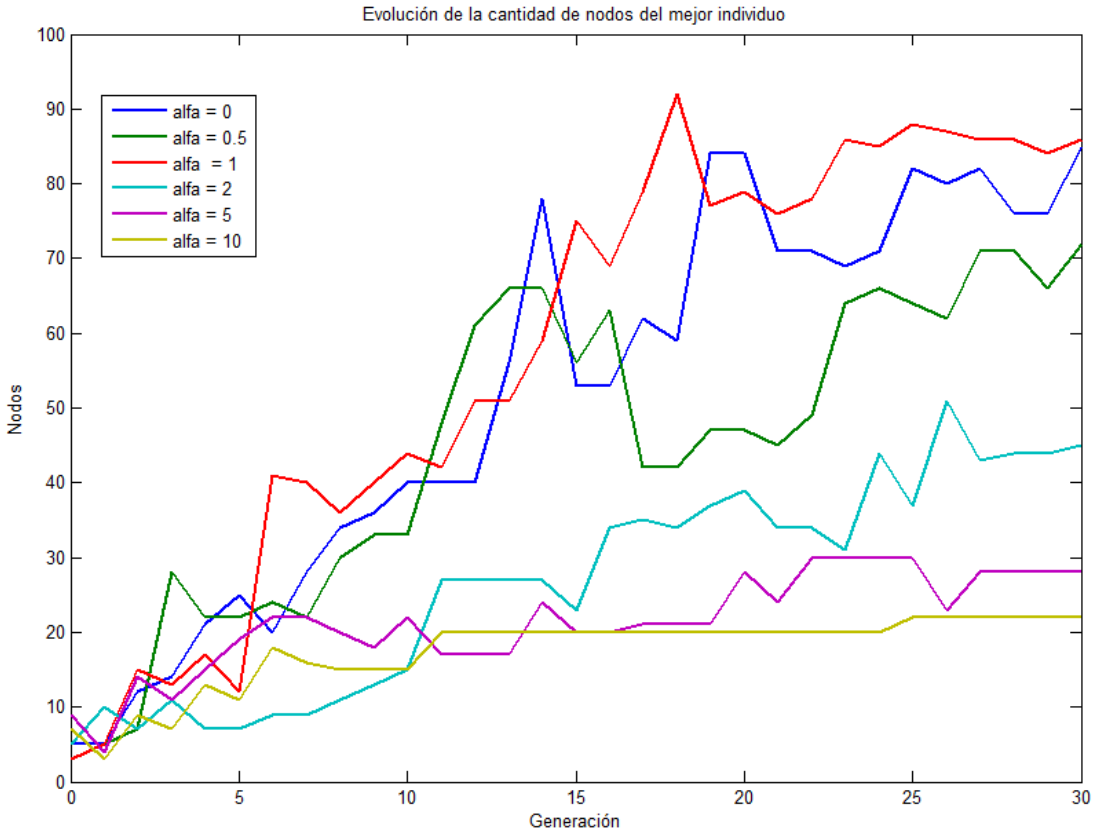

Figura 3.4 Evolución de la cantidad de nodos del mejor individuo en los casos de estudio

En el gráfico de la figura 3.5 se muestra la evolución de fitness del mejor individuo en los mismos 6 casos.

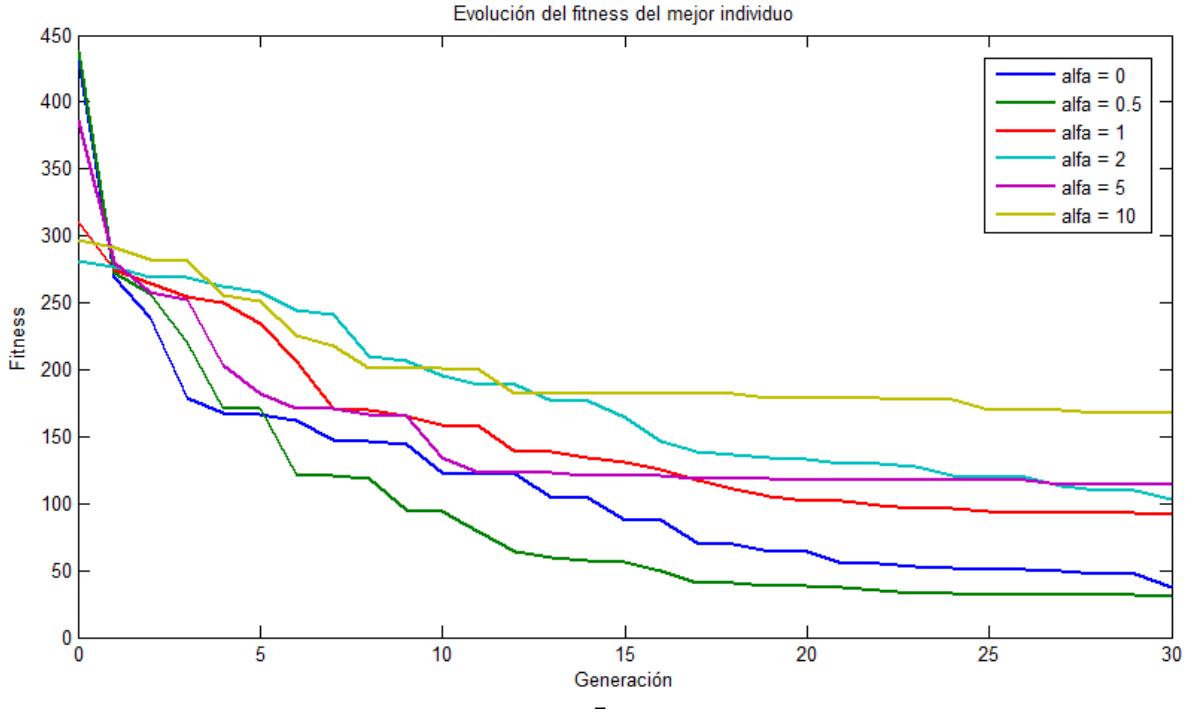

Figura 3.5 Evolución del fitness del mejor individuo en los casos de estudio

Del gráfico 3.4 y 3.5 se puede concluir que mientras mayor es α, menor es el número de nodos que tiene el individuo de mayor fitness, pero el fitness al que llega en la misma cantidad de generaciones es menor. Esto se debe a que la expresión del fitness 3.4 minimiza el error (primer término) y el número de nodos (segundo término) por lo que lograr un equilibrio entre estos dos objetivos es fundamental para obtener buenos resultados. En particular, al tener un $\alpha = 0.5$ se obtiene un fitness similar al de la corrida con MSE mientras que el número de nodos del mejor individuo es menor.

Otra conclusión importante de estas pruebas es que de todas las expresiones obtenidas, ninguna tenía las unidades físicas que correspondía (grados Celsius), por lo que si se requiere que las expresiones tengan un sentido físico se deben agregar restricciones al algoritmo.

3.4 Datos y Diseño de experimentos

Los datos que serán utilizados como estándar dorado provienen de simulaciones hechas en el software CFD ANSYS. Específicamente son 1500 simulaciones, cada una con distintos valores de las variables de diseño, las que son: corriente en la celda, separación entre celdas, flujo en la entrada, temperatura inicial del aire y diámetro de la celda. De estos 1500, 500 se usan como datos de entrenamiento, 500 de validación y 500 de prueba.

Cada una de las configuraciones representa un punto en el espacio de diseño. Para generar los puntos de los tres conjuntos, se utilizó el método *'latin hypercube'* (LH), construyendo cada conjunto por separado.

Para hacer los LH se deben definir rangos para cada variable de diseño, estos son los siguientes:

Corriente (I): 0 a 15 [A]

Separación (S): 0 a 1.5 [veces el diámetro de una celda]

Flujo (F): 0 a 200 [CFM]

Temperatura inicial del aire (Tin): 10 a 30 [°C]

Diámetro de las celdas (D): {18, 22,26, 32} [mm]

En la figura 3.6 se muestran un esquema del banco de baterías con las variables de diseño: diámetro de la celda (D), separación de las celdas (S), corriente(I), temperatura de entrada(Tin) y flujo de entrada (F).

Los rangos de corriente y separación son idénticos a los rangos impuestos en el modelo fenomenológico, el rango del flujo es un poco menor ya que para valores de flujo muy grandes la densidad comienza a tener un error muy grande. A la temperatura inicial del aire se le dio ese rango de valores ya que se espera que en condiciones reales el aire de entrada no sobrepase dichos umbrales. El caso de los diámetros de las celdas es distinto ya que son valores discretos, esto es porque esos 4 diámetros corresponden a las celdas más utilizadas, por lo que carece de sentido tener un rango continuo.

La variable corriente corresponde a la corriente que circula por una sola celda que se asume es igual en todas las celdas, ya que una simplificación del modelo es que todas las celdas son idénticas.

Generalmente en un conjunto de datos los porcentajes de entrenamiento, validación y prueba son un 80%, 10% y 10% respectivamente, en este caso se da el caso contrario ya que si bien el conjunto de entrenamiento es igual a los otros dos, se entrena con solo 125 datos a la vez debido a que los datos de entrenamiento deben ser usados para cada árbol por lo que si tenemos un conjunto muy grande de datos el algoritmo gastaría muchos recursos computacionales además de mucho tiempo. Es una práctica común en problemas de programación genética usar un conjunto pequeño como datos de entrenamiento y comprobar luego la validez de estos usando un conjunto diferente que represente mejor el espacio completo usando más datos. Esto se puede ver en (Koza, 1992) donde se aborda un problema de regresión simbólica.

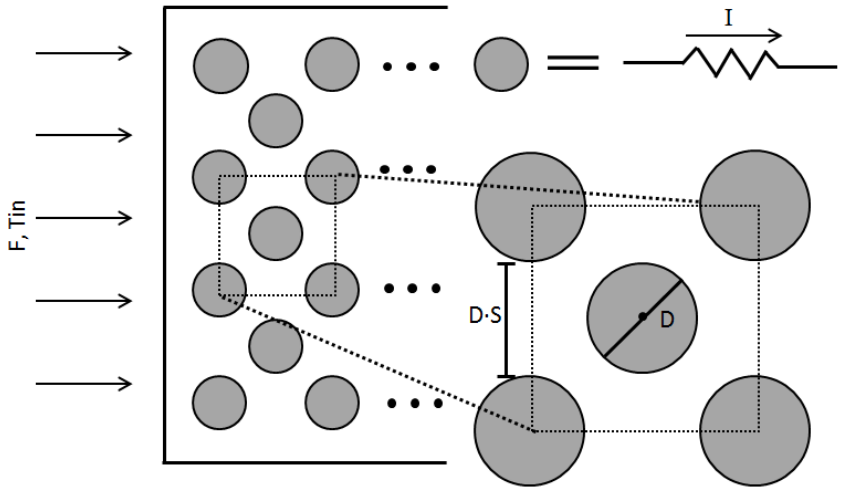

Figura 3.6 Esquema que representa las variables de diseño del modelo fenomenológico

3.5 Implementación de programación genética

En total se tienen tres expresiones que ajustar: número de Nusselt, coeficiente de arrastre, factor de fricción y factor de forma, los dos últimos referidos en esta memoria solo como factor de fricción y tratados como una sola expresión, ya que ambos aparecen en el cálculo de la presión del aire como parte de una multiplicación, además, las dependencias del factor de forma son un subconjunto de las dependencias del factor de fricción y como el objetivo es ajustar las variables de salida, en este caso la presión, no es necesario distinguir entre estas dos expresiones (ambas adimensionales).

Se evolucionó cada expresión por separado. Para esto, se implementó una función de MATLAB usando el toolbox GPLab que corre el algoritmo de programación genética (sección 2.3.3). En esta función es posible cambiar cada uno de los parámetros necesarios en el algoritmo. Existen en total 83 parámetros configurables, sin embargo, una gran parte de ellos son para etapas o características del algoritmo que no son de utilidad en el presente trabajo, siendo 10 los parámetros configurados.

3.5.1 Parámetros

Inicialización de la población: la profundidad máxima que pueden tener los arboles en la población inicial se fijó en 6 como se recomienda en (Koza, 1992). La función de inicialización utilizada fue *ramped half-half* (Koza, 1992). Como ya se conoce una aproximación a las expresiones que se desean evolucionar proporcionadas por el modelo fenomenológico, con el fin de aprovechar este conocimiento se le hizo una modificación al método *ramped half-half* que consiste en crear la mitad de los individuos de la manera recién descrita y la otra mitad a partir de la aproximación ya conocida. Los individuos de esta segunda mitad se diferencian entre sí por las constantes que son generadas aleatoriamente en torno al valor que tiene la expresión original.

Tamaño de arboles: la profundidad máxima de los árboles en todas las generaciones se fijó en 17 como se sugiere en (Koza, 1992).

Funciones: esta aplicación es escogieron las funciones {+,-,*,/,^,^2} donde la función división es modificada para que la división por cero sea 1. Este set de funciones se eligió debido a que estas son las funciones que tienen las expresiones originales.

Terminales: Se tienen 3 expresiones por evolucionar, cada una de estas en una corrida distinta de programación genética. Como se ajustó cada expresión por separado, se utilizan 3 conjuntos distintos de terminales. Dado que cada una de estas expresiones depende de distintas variables en el modelo fenomenológico, los sets de terminales corresponden a las constantes, parámetros y variables de las que depende cada expresión en el modelo original más otros parámetros. De esta manera, los conjuntos de terminales para cada caso son:

- factor de fricción y factor de forma: $\{rand, \frac{Vmf_{(i)}}{Vf_{ref}}, \frac{Df_{(i)}}{Df_{ref}}, S, Rem_{(i)}\}$
- Coeficiente de arrastre: $\{rand, \frac{A_{sup}}{A_{ctrl}}, Rem_{(i)}, \frac{Df_{(i)}}{Df_{ref}}\}$
- Nusselt: $\{rand, Rem_{(i)}\}$

Donde $rand$ es un número aleatorio entre -1 y 1, $Vmf_{(i)}$ es la velocidad media del fluido en la posición i en $[m/s]$, Vf_{ref} es una velocidad de referencia en $[m/s]$, $Df_{(i)}$ es la densidad del fluido en la posición i en $[kg/m^3]$, Df_{ref} es una densidad de referencia en $[kg/m^3]$, S es la separación de las celdas (adimensional), $Rem_{(i)}$ es el número adimensional de Reynolds en la posición i, A_{sup} es el área de la celda menos las dos caras circulares en $[m^2]$ y A_{ctrl} es el área que hay entre los celdas de una columna de fluido en $[m^2]$.

Tanto la velocidad como la densidad están divididas por un valor de referencia para normalizarlos y dejar esos valores adimensionales, ya que las expresiones que se están buscando son adimensionales. De esta forma, no importa cómo se opere con el conjunto de terminales, el resultado siempre será adimensional.

Operadores: Se utilizaron los operadores: crossover, mutación y *prune&plant*(sección 2.3.3.3.1) y sus probabilidades de ocurrencia son 80%, 10% y 10% respectivamente. Estos valores se basan en los propuestos en (Koza, 1992).

Selección: El método de selección utilizado fue *"lexicographic parsimony pressure"*(LPP).

Sobrevivencia: Para pasar a la siguiente generación se debe generar otra población completa. El método para determinar quienes pasan a la siguiente generación es *keep best.*

Condición de término: se hace una inspección manual cada 10 generaciones y se detiene el algoritmo cuando el individuo de mejor fitness parece haber convergido (Koza, et al., 2003).

3.6 Función de fitness

Se tiene un conjunto de entrenamiento de 500 datos, pero no es posible evaluar la función de fitness para cada individuo con todos los datos debido a restricciones computacionales y de tiempo. Para solucionar este problema se utilizó un método de submuestreo llamado *vertical slicing* (Korns, 2007).

Vertical Slicing: Los datos de entrenamiento están divididos en dos matrices, la de diseño de experimentos (DoE), y la de los resultados, los que son obtenidos al simular los puntos de la matriz DoE en ANSYS. Este método consiste en ordenar ambas matrices de acuerdo al orden ascendente de la matriz de resultados y separarlas en conjuntos de igual cantidad de muestras. Luego, en el comienzo de cada generación se escoge de forma aleatoria un subconjunto, el que se utilizará como conjunto de entrenamiento en todos los individuos de la generación. En este caso se utilizaron subconjuntos de 125 muestras ya que con este valor el cálculo de la función de fitness demora un tiempo razonable (0.7 segundos por individuo).

Para calcular el fitness de un individuo lo primero que se hace es interpretar el árbol obteniendo la expresión que representa. Esta expresión es la entrada a un modelo fenomenológico modificado. Este, en vez de tener la expresión original del parámetro objetivo (coeficiente de fricción, coeficiente de arrastre o número de Nusselt), contiene la expresión del individuo.

Luego, se obtienen las salidas del modelo fenomenológico modificado para los datos del conjunto de entrenamiento. Dado que el modelo es iterativo, se puso como condición adicional que si se itera 10 veces sin converger, el modelo entrega el resultado que lleva hasta el momento.

Teniendo las salidas del modelo, estas se pueden restar con las salidas del software ANSYS para obtener el error ε

$$\varepsilon_i = f_i^{ansys}(x) - f_i^{pg}(x) \tag{3.5}$$

Donde x es el vector de variables de diseño, f^{ansys} es la salida deseada (salida del CFD ANSYS), f^{pg} es la salida obtenida con el individuo evolucionado con programación genética e i es la muestra (del conjunto de entrenamiento) donde se está evaluando el error.

Se implementaron dos funciones de fitness, ambas son variaciones del error cuadrático medio (MSE) que pretenden ayudar en el control del *bloat.* La primera es un MSE con regularización y la segunda es un MSE con una ponderación. La función de fitness que corresponde al MSE se muestra en la expresión 3.6

$$MSE = \frac{1}{N}\sum_i(\varepsilon_i)^2 \tag{3.6}$$

Donde N es la cantidad total de puntos en los que se evalúa el error. Utilizar el MSE como función de fitness tiene la ventaja que castiga más a un individuo mientras los puntos estén más alejados, dando una menor penalización a los valores parecidos, esto hace que el algoritmo tienda a corregir los valores alejados, sin embargo, esta estrategia es débil ante *outliers* y además asume que la distribución del error es gaussiana.

3.6.1 MSE con regularización

Siguiendo la idea de la expresión 2.5 (sección 2.4), al MSE se le suma un valor que es proporcional al tamaño del individuo, por lo tanto la función de fitness ($fitness_{reg}$) es la que se muestra en la expresión 3.7

(3.7) $$fitness_{reg} = MSE + \alpha\sqrt{nodos}$$

Donde $nodos$ es la cantidad de nodos del árbol del individuo y α es un parámetro de ajuste.

3.6.2 MSE con ponderación

La función de fitness ($fitness_{pond}$), que utiliza una ponderación para castigar a individuos muy grandes, como la propuesta en (Cai, et al., 2006), se muestra en la expresión 3.8

(3.8) $$fitness_{pond} = MSE \cdot \left(1.5 + \frac{(nodos - L)}{2(1 + |nodos - L|)}\right)$$

Donde $nodos$ es la cantidad de nodos que tiene el individuo y L es un número entero que define la cantidad de nodos para que la función que multiplica al MSE tome un valor de 1.5 que es la mitad de su recorrido ya que está acotada por [1,2]. En la figura 3.7 se puede ver el gráfico de esta función.

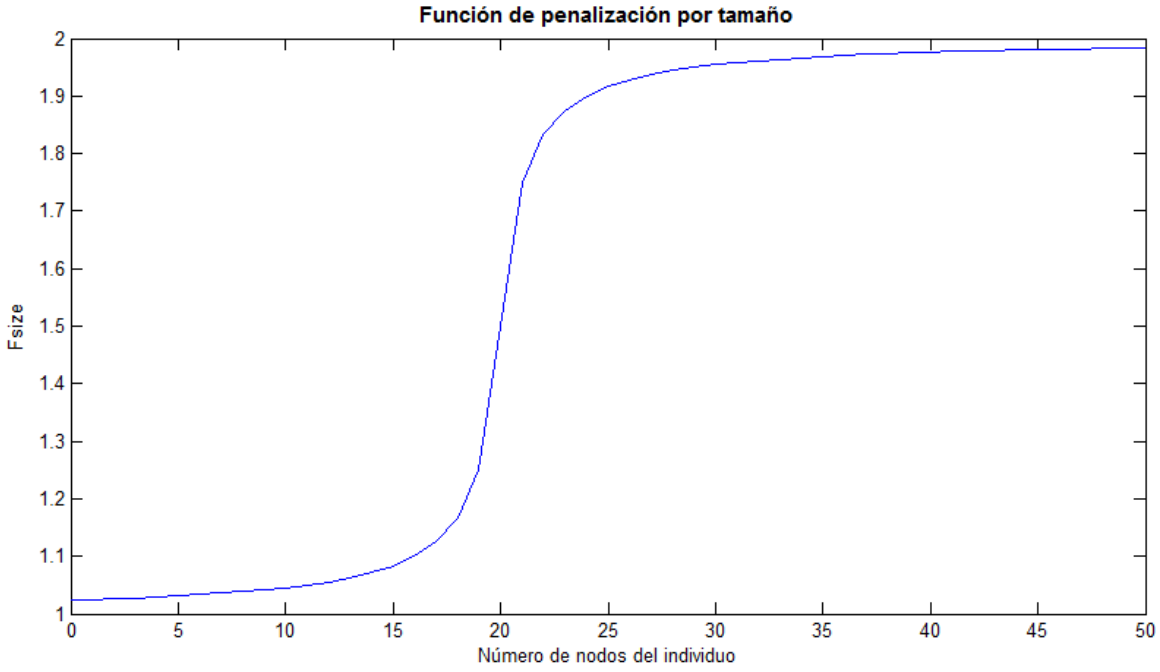

Figura 3.7 Gráfico de la función de penalización por tamaño con L = 20

Como se puede ver en el gráfico de la figura 3.7, para L=20 los individuos con árboles con menos de 15 nodos reciben una penalización pequeña, pero desde ahí en adelante, el fitness de un individuo puede ser hasta 2 veces mayor por tener un árbol muy grande.

3.7 Modificaciones al modelo fenomenológico

Al modelo original se le hicieron una serie de cambios con el fin de cumplir con los requerimientos del presente problema.

Se agregaron como variables de entrada la temperatura inicial del aire y el diámetro de la celda. La salida es cambiada de acuerdo a la variable que se necesite (ya que se hacen tres corridas distintas), por ejemplo, el primer valor de la presión, la temperatura de la celda central o el valor en la salida de la velocidad.

Otra modificación fue hecha al modelo fenomenológico fue cambiar la forma en que se calcula la densidad del fluido. Se hizo esta simplificación debido a que ésta es una primera aproximación al problema y para disminuir la cantidad de ecuaciones en el modelo por lo que disminuye la carga computacional. La densidad del aire, por lo tanto, es calculada en función de la temperatura en el mismo punto. Esta relación es obtenida mediante la tabla 3.1 que muestra la densidad del aire conocidos para algunos valores de temperatura (www.TheEngineeringToolbox.com, s.f.). Los valores que no aparecen en la tabla son obtenidos mediante una interpolación lineal.

Calculando la densidad del aire de esta forma ya no es necesario calcularla como se explicó en la sección 2.7.1 por lo que se reduce el número de expresiones a evaluar.

Tabla 3.1 Densidad del aire en función de la temperatura (www.TheEngineeringToolbox.com, s.f.)

Temperatura [°C]	Densidad [kg/m³]
-50	1.534
0	1.293
20	1.205
40	1.127
60	1.067
80	1

A continuación se presenta un pseudo-código del modelo fenomenológico luego de las modificaciones, donde el índice i representa las columnas de fluido que son aquellas columnas compuestas por dos celdas en el bloque elemental utilizado en el modelo. Un bloque tiene 2 columnas de fluido que encierran a la celda central.

En este pseudo-código los valores entre paréntesis son aquellos necesarios para calcular el valor que está a la derecha del paréntesis y que corresponde a una de las salidas del modelo original. Por ejemplo, la velocidad media, el coeficiente de fricción y el número de Reynolds son calculados para poder obtener la presión del fluido.

```
Definir constantes y parámetros
Definir Condiciones iniciales
while (errores mayores que e)
      Calcular (coef arrastre en 1,fuerza arrastre en 1)velocidad fluido en 1
      Calcular (Reynolds, factor de fricción) presión fluido en 1

      for (cada columna de fluido(i)-1)
            Calcular (velocidad media, fricción, Reynolds)presión fluido en i
            Calcular (Coef. de arrastre, F de arrastre)velocidad fluido en i+1
            Calcular (calor especifico)temperatura fluido en i+1
            Calcular densidad fluido en i+1
            Calcular (Nusselt, conductividad)temperatura de la celda en i
      end
end
```

Las constantes y parámetros que se definen en un principio son las dimensiones del banco de baterías y la cantidad de celdas, constantes como la presión atmosférica, la resistencia interna de las baterías que se asume son todas iguales, y otros parámetros de interés como la superficie de una batería, el volumen de control, y el área de este volumen.

Las condiciones iniciales son la temperatura inicial del fluido que es entregada por el usuario, la velocidad de inicio que es función del flujo y de las dimensiones del banco, la densidad del aire en la entrada que depende de la temperatura inicial y la presión en la salida que se asume es igual a la presión atmosférica.

Luego, al entrar en el ciclo iterativo se calculan las variables de salida del modelo original que son: la presión, velocidad, temperatura y densidad del fluido y la temperatura de las celdas. Es de aquí de donde también se pueden obtener los coeficientes de interés para el presente trabajo ya que tanto el coeficiente de arrastre, el coeficiente de fricción y el número de Nusselt son expresiones que dependen de otras variables calculadas dentro del ciclo por lo que pueden ser evaluadas.

Capítulo 4

Resultados

Se decidió evolucionar cada expresión por separado debido a la facilidad de implementación y a que si se evolucionan todas juntas, la complejidad del problema se ve aumentada ya que el algoritmo debe buscar no solo una expresión que reduzca el error de una variable de salida del modelo fenomenológico, si no que tres expresiones que reduzcan tres error de distintas variables simultáneamente lo que se traduce en un costo computacional y de tiempo mucho mayor.

Cada una de las tres expresiones elegidas para ser ajustadas fue evolucionada tomando en cuenta el error en la variable de salida del modelo fenomenológico en la que aparece, de esta forma, al evolucionar el factor de fricción se utilizó el error en la presión del aire, para el coeficiente de arrastre se utilizó la velocidad del aire y finalmente para el número de Nusselt se utilizó la temperatura en la celda central.

Cada expresión fue evolucionada más de una vez variando distintos parámetros del algoritmo de programación genética. Estas corridas fueron hechas utilizando el conjunto de entrenamiento. Luego, del conjunto de expresiones obtenidas al finalizar todas las corridas se seleccionó la mejor comparando los resultados obtenidos con el conjunto de validación. El criterio utilizado para elegir la mejor expresión es cumplir que la media y varianza del error del modelo evolucionado (modelo con la expresión evolucionada) deben ser menores a las del modelo fenomenológico original. Aquel modelo que tenga la menor media y cumpla con la anterior es considerado el mejor modelo.

Finalmente se compara el modelo fenomenológico original con el mejor modelo seleccionado utilizando el conjunto de prueba. A continuación se muestran los resultados obtenidos para los tres coeficientes.

4.1 Factor de Fricción

Se realizaron 11 corridas del algoritmo de programación genética para este coeficiente, cada una con una función de fitness distinta. Una se hizo utilizando el error cuadrático medio (MSE), 5 fueron hechas con MSE con regularización (sección 3.6.1) con 5 valores distintos para el parámetro α, y 5 con MSE con ponderación (sección 3.6.2) con 5 valores distintos del parámetro L. Los valores de α son 0.1, 0.5, 0.75, 1 y 2 mientras que los de L son 10, 20, 30, 40 y 50. Todos los parámetros del algoritmo de programación genética utilizados en estas corridas se encuentran en el apéndice B.1

4.1.1 Modelo original

En el modelo fenomenológico original el factor de fricción se calcula como se indica en la expresión 4.1

(4.1) $$factor\ de\ fricción = A_{(s)} \cdot Reynolds^{B(s)}$$

Donde Reynolds es el número de Reynolds, $A_{(s)}$ y $B_{(s)}$ son funciones lineales por tramo dependientes de la separación de las celdas.

El error de la presión del modelo fenomenológico original con respecto a ANSYS utilizando el conjunto de validación tiene una media de 53.2[Pa] y una varianza de $4.1 \cdot 10^4$. Estos valores son considerables ya que en un banco de celdas escalonadas con 4 bloques la caída de presión va de 20[Pa] a 80[Pa][1]. Tener un error promedio de 56[Pa] para un banco de un bloque no es aceptable ya que se espera que la caída de presión sea menor por lo que escapa de los rangos de operación.

En la figura 4.1 se muestra el histograma del error para el modelo fenomenológico original. En éste se observa que la distribución del error sigue una similar a una *power law*. Una forma de identificar esta distribución es graficándola en escala logarítmica en ambos ejes. En la figura 4.2 se muestra este gráfico donde se observa una tendencia aproximadamente lineal, que es la característica que tiene la distribución *power law* en este tipo de gráfico.

Es importante señalar que el error en la presión del modelo fenomenológico original tiene muy pocos valores negativos y son muy cercanos a cero lo que quiere decir que este modelo subestima la presión que entrega ANSYS. Esto se debe a que la expresión utilizada en el modelo para calcular la presión solo toma en cuenta la presión dinámica mientras que ANSYS entrega la presión total, por lo que es normal que se subestime la presión ya que al modelo le falta calcular la presión estática.

Al ajustar el error por una *power law* se obtiene la expresión 4.2

(4.2) $$error = 10^{-0.6} P_{modelo}^{1.2291}$$

Donde $error$ es el error en la presión del modelo fenomenológico original y P_{modelo} es la presión obtenida por este modelo. Utilizando esta expresión para calcular el error se obtiene la distribución de la figura 4.3, la que es muy similar a la figura 4.1.

En la figura 4.4 se muestra un gráfico que identifica con puntos azules aquellos puntos del conjunto de prueba que el modelo fenomenológico obtiene con un error menor a 100[Pa] y de rojo los que no. En este gráfico los ejes son la separación de las celdas y el flujo de entrada. Éste se hizo para determinar si los puntos extremos que se observan en la figura 4.1 corresponden a alguna sección particular del espacio de variables de diseño. Efectivamente los puntos con error mayor a 100[Pa] se concentran en una separación menor a 0.5 y aumentan con el flujo de entrada.

Sacando los puntos del conjunto de validación con separación menores que 0.5 la media del error baja a 4.8 [Pa] y la varianza a 223.5, lo que es de esperar ya que la distribución que tiene el error se caracteriza por mejorar de manera significativa al quitar los valores extremos.

[1] Estos datos fueron obtenidos experimentalmente por Francisco Moser, quien estudió el comportamiento térmico de bancos de batería de ión-litio en configuración de grilla escalonada y alineada.

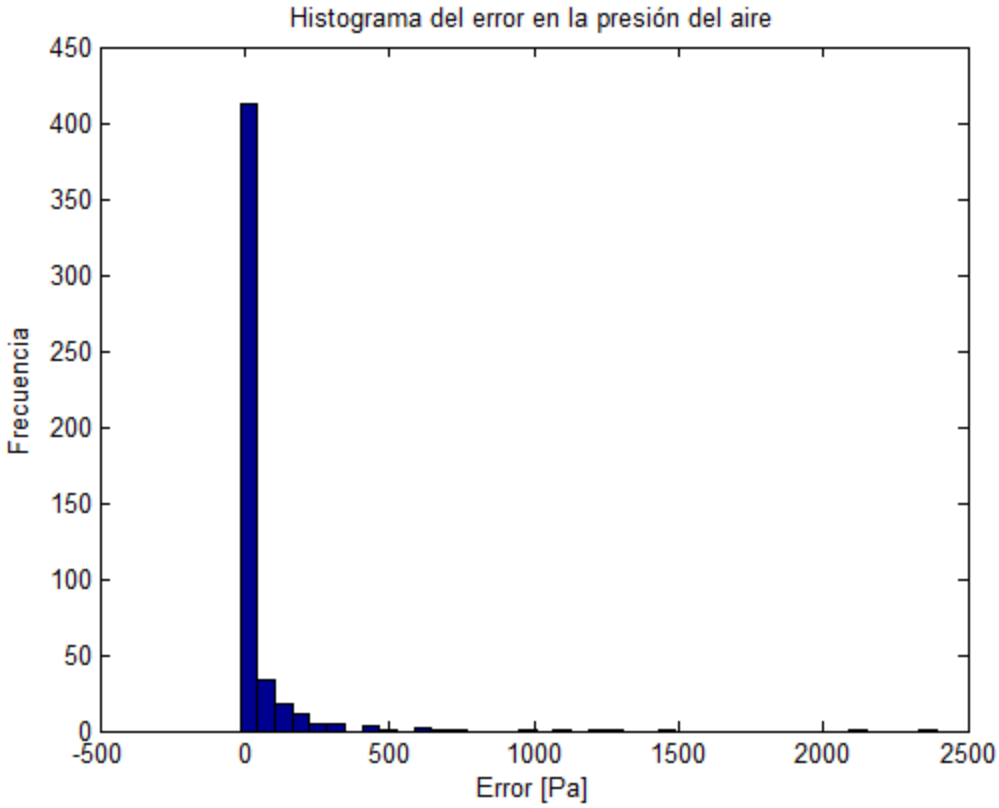

Figura 4.1 Histograma del error en la presión del aire para el modelo fenomenológico original

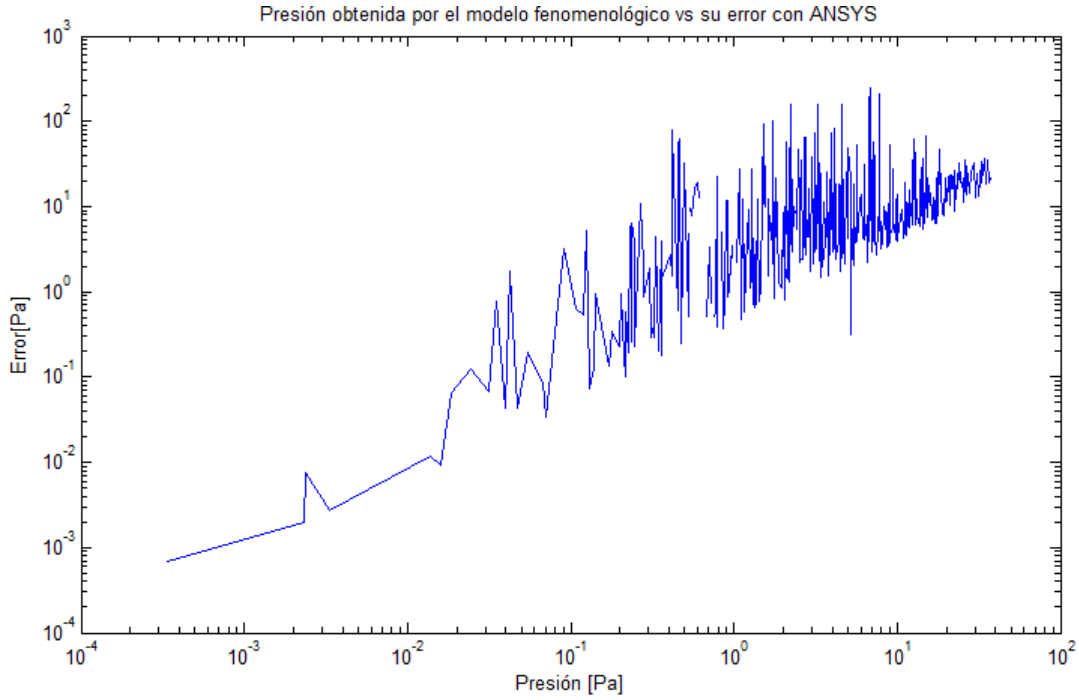

Figura 4.2 Presión del aire obtenida usando el modelo fenomenológico vs su error con ANSYS

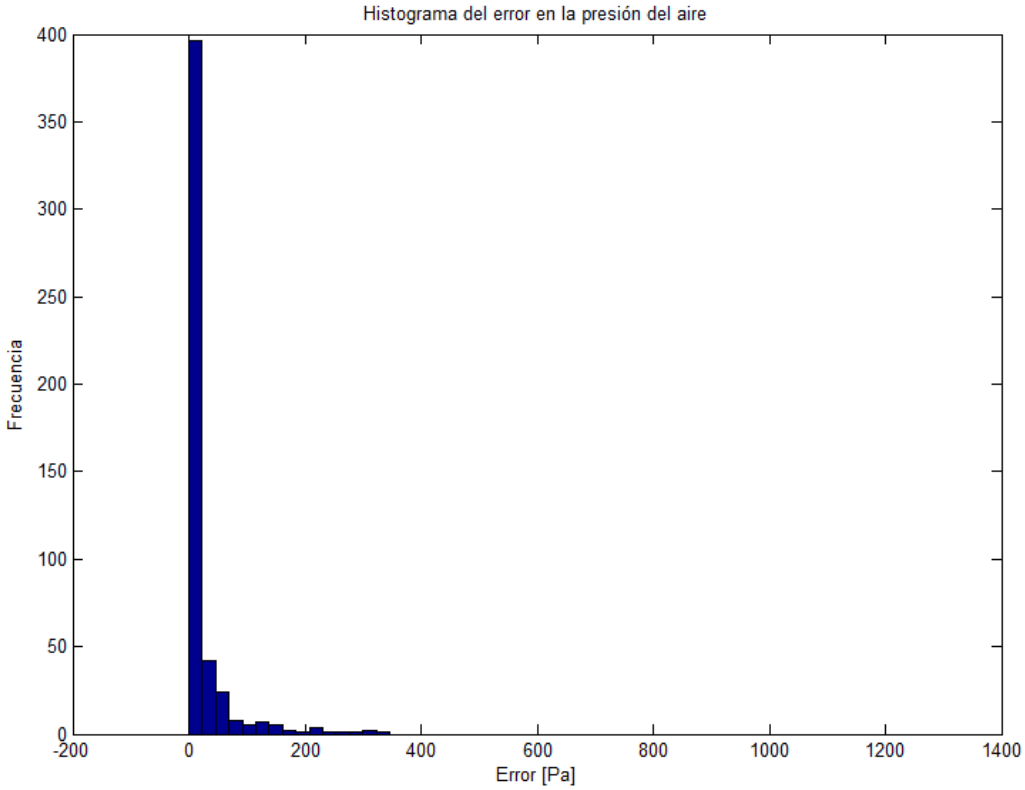

Figura 4.3 Distribución *power law* obtenida con el ajuste al error

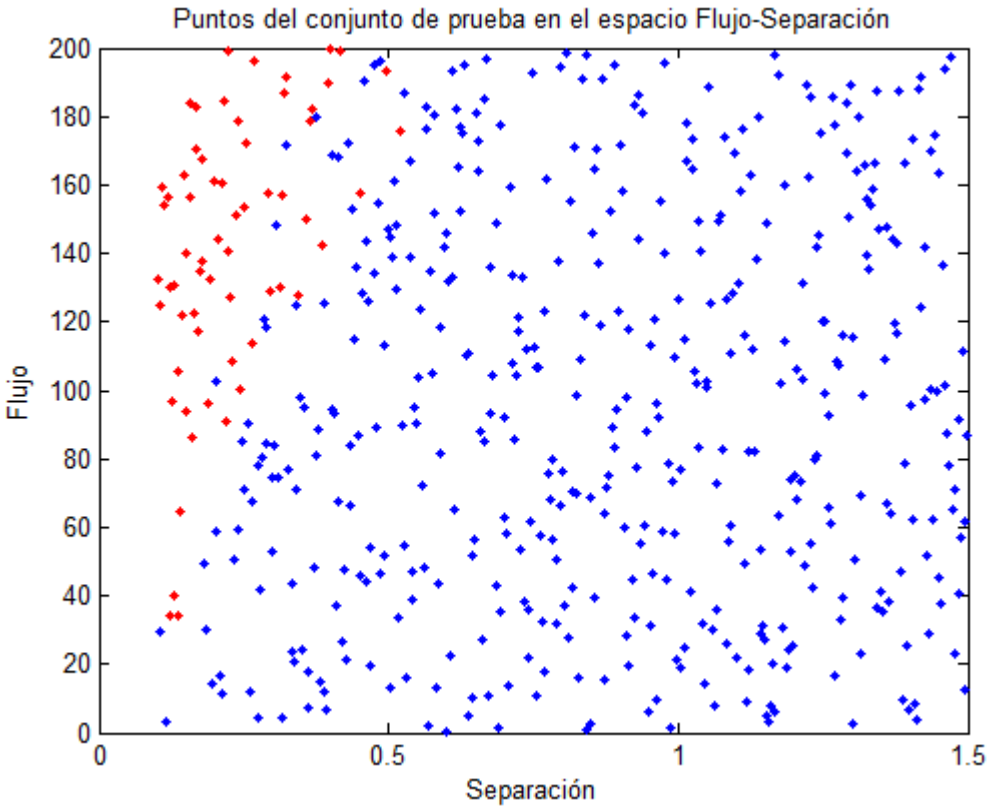

Figura 4.4 Puntos del conjunto de prueba con un modulo del error menor y mayor a 100[Pa] de azul y rojo respectivamente. Usando el modelo fenomenológico original

4.1.2 Expresión evolucionada con MSE

En 24 generaciones el algoritmo demoró 5 días de los cuales 24 horas se gastaron en su última generación por lo que se decidió terminar la corrida por tiempo. El aumento creciente en el tiempo en cada generación se debe al crecimiento de los arboles, ya que como se muestra en la figura 4.5, a medida que aumentan las generaciones, el promedio del tamaño de los arboles también aumenta por lo que el tiempo en procesar un árbol es cada vez mayor.

Los valores de media, varianza y cantidad de nodos para los mejores 10 individuos de esta corrida de programación genética se encuentran en el apéndice C.1.

De los 10 individuos estudiados, dos de ellos tienen una media -3.7[Pa] y una varianza de 1529, diferenciándose en el número de nodos ya que uno tiene 54 mientras que el otro 58. Estos dos individuos representan funciones muy complejas, y que no son similares a la expresión original del factor de fricción por lo que no tienen un significado físico claro. Otros individuos con buen fitness si bien tienen menos nodos, siguen siendo funciones muy complejas y sin similitud alguna a la expresión 4.1. Como ejemplo, la expresión 4.3 corresponde a la de un árbol de 40 nodos, una media en el error de la presión de 10.5[Pa] y una varianza de 930.3

(4.3) $$factor\ de\ fricción = \left(0.006 \cdot 17^{-2^{Re^{0.36^{2\sqrt{SV\left(\frac{D_f}{D_{ref}}\right)^{Re}+\frac{D_f}{D_{ref}}(S-1)+S}}}}} - \frac{0.9}{S^2+5S\frac{Vf}{V_{ref}}} \right)^2$$

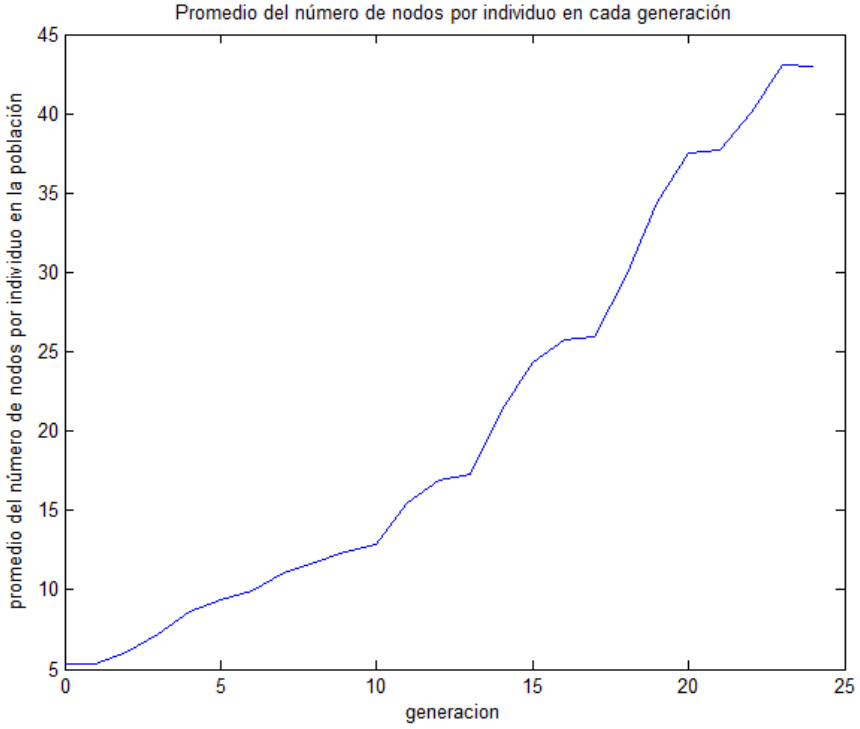

Figura 4.5 Promedio de la cantidad de nodos por individuo en cada generación

Donde Re es el número de Reynolds, V_f es la velocidad del aire, V_{ref} es una velocidad de referencia, D_f es la densidad del aire, D_{ref} es una densidad de referencia y S es la separación entre las celdas.

4.1.3 Expresión evolucionada con MSE con regularización

Se hicieron 5 corridas distintas del algoritmo de programación genética para 5 valores del parámetro α ya que este define la relación entre la minimización del error y la minimización del tamaño de los individuos.

De las 5 corridas, 4 se detuvieron en la generación 30 debido a que exhibían muy poca o nula mejora en la función de fitness del mejor individuo. Por otro lado, la corrida con α más pequeño y por lo tanto, aquella que daba más prioridad a la minimización del error, comenzó a demostrar el mismo comportamiento que la corrida hecha con MSE, demorando casi 24 horas en su última generación, por lo que se detuvo en la generación 24.

En la figura 4.6 se muestra el gráfico del promedio de la cantidad de nodos por individuo en cada generación para cada valor de α. En este se observa que para $\alpha = 0.1$ los árboles crecen muy rápido mientras que para $\alpha = 2$ el crecimiento es mucho más lento. Para los casos intermedios el comportamiento es muy similar ya que en la generación 30 llegan a valores muy parecidos, sin embargo este comportamiento comienza en la generación 21 y 26 para $\alpha = 0.5$ y $\alpha = 0.75$ respectivamente ya que para generaciones anteriores tienen un crecimiento mayor al de $\alpha = 1$, el que tiene un crecimiento constante.

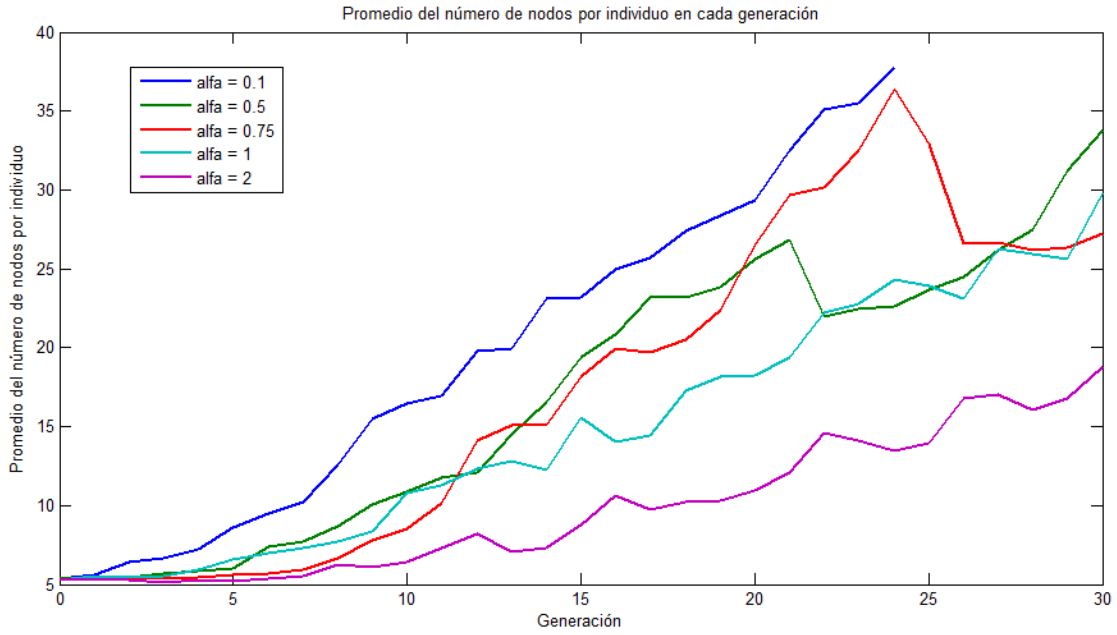

Figura 4.6 Promedio de la cantidad de nodos por individuo en cada generación para cada valor de alfa

De las 50 expresiones estudiadas (los mejores 10 individuos de cada caso), las 20 que corresponden a las corridas con α más grande (1 y 2) obtiene individuos con media y varianza superior o similares a las de la expresión original. Esto es debido a que los valores de α son muy elevados ya que se obtienen expresiones de 3 o 4 nodos, por lo que la minimización del tamaño de los árboles se priorizó mucho. Para $\alpha = 0.1$ se obtiene el mejor resultado con una media de

0.4[Pa] , varianza 813.8 y un árbol de 25 nodos. Al simplificar la expresión obtenida para este individuo se obtiene la expresión 4.4.

(4.4) $$factor\ de\ fricción = 4\left((9S)^{(2.81\cdot S)^{-2}} + \left(\frac{0.3021}{S^2}\right)^2 - 0.6\right) Re^{-0.2655}$$

Donde S es la separación entre las celdas y Re es el número de Reynolds.

La expresión 4.4 tiene la misma forma que la expresión 4.1. La diferencia son las funciones $A_{(s)}$ y $B_{(s)}$ ya que ahora B es una constante y $A_{(s)}$ no es lineal.

4.1.3.1 Ajuste fino

A la expresión seleccionada se le hizo un ajuste fino manualmente. El objetivo de esta prueba es identificar si la media y varianza disminuyen para pequeñas variaciones en distintos coeficientes de la expresión. Los coeficientes que se modificaron fueron el exponente del número de Reynolds, la expresión completa y el primer, segundo y tercer término del trinomio.

Se varió cada coeficiente por separado en un 1%. Si tanto la media como la varianza disminuyen se sigue buscando la menor media con incrementos sucesivos de un 1%. Si la media cambia de signo se hace una búsqueda con incrementos del 0.1% para encontrar la menor media.

En la tabla 4.1 se muestra un resumen del ajuste fino. La tabla completa se encuentra en el apéndice D.1.

Tabla 4.1 Resumen del ajuste fino para la expresión evolucionada con MSE con regularización

Constante	Valor	Media	Varianza
Expresión evolucionada		0,40793423	813,84
Exponente Rem	-1%	-1,02329937	914,95
	+1%	1,80589484	759,36
Expresión completa	-1%	1,0122505	782,72
	+1%	-0,19638204	853,73
Primer término trinomio	+2%	0,01921449	809,06
	+2.1%	-0,00022149	808,83
	+2.2%	-0,01965748	808,60
Segundo término trinomio	-1%	0,89036389	780,29
	+1%	-0,07449542	856,10
Tercer término trinomio	-5.5%	0,00933134	813,69111
	-5.6%	0,00208402	813,691593
	-5.7%	-0,00516331	813,692192

En la tabla 4.1 se puede ver que tanto para el exponente en el número de Reynolds, la expresión completa y el segundo término del trinomio al variar su valor, la media y varianza tienen un

comportamiento contrario, ya que una aumenta y la otra disminuye. En cambio, al aumentar en un 2.1% y disminuir en un 5.6% el primer y tercer término respectivamente se obtienen mejoras tanto en la media como en la varianza.

De los dos ajustes realizados, el hecho al primer término del trinomio es el que presenta una disminución más significativa tanto en la media como en la varianza por lo que este es el ajuste adoptado.

4.1.4 Expresión evolucionada con MSE con ponderación

Se hicieron 5 corridas distintas del algoritmo de programación genética para 5 valores del parámetro L, ya que éste define la función que castiga a los individuos muy grandes. Las 5 corridas se detuvieron en las 30 generaciones pues ya no había mucho cambio en el fitness del mejor individuo y por limitaciones de tiempo.

El tiempo que tardaron tuvo un promedio de 4 días, siendo de 3 días para el menor valor de L y 5 para el mayor. Esto debido a que L está directamente relacionado con la cantidad de nodos que tiene los árboles. Mientras mayor es éste, el costo computacional de operar con los árboles es mayor.

En la figura 4.7 se muestra un gráfico del promedio de la cantidad de nodos por individuo en cada generación. En éste se puede ver que a mayor L los árboles tienen más nodos. Los únicos valores que no siguen esta regla son L=40 y L=50. Esto puede ser debido a que como son número grandes, los individuos evolucionan sin restricciones ya que la función de penalización para valores pequeños no tiene mucho impacto en la función de fitness. Luego, cuando se llega a un valor de nodos determinado la función comienza a penalizar más fuertemente a los individuos por lo que es de esperar que L=40 tenga un estancamiento cercano a 20 nodos. El hecho de que la corrida hecha con L=40 crezca más rápido que la con L=50 puede ser debido al carácter aleatorio de los operadores genéticos.

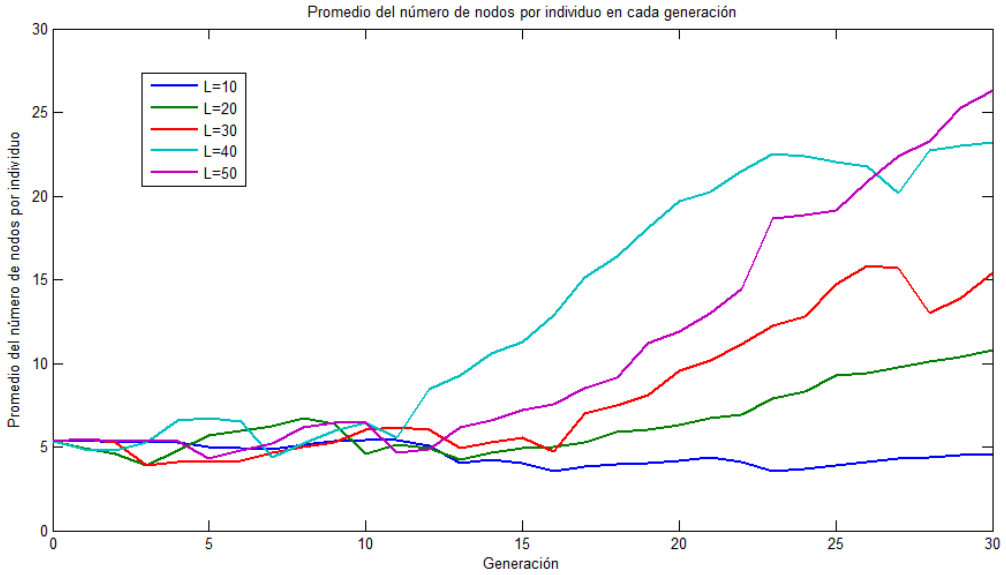

Figura 4.7 Promedio de la cantidad de nodos por individuo en cada generación para cada valor de L

Se tomaron los 10 mejores individuos de cada corrida por lo que en total se obtuvieron 50 expresiones, de las cuales, en la mayoría se observa una disminución en el error respecto a la

expresión original pero un aumento en la varianza. La expresión 4.5 corresponde al mejor individuo que tiene una media del error de 7.3, una varianza de $2.7 \cdot 10^3$ y un árbol de 6 nodos.

(4.5) $$factor\ de\ fricción = \frac{0.25371}{S^3}$$

La expresión 4.5 mejora tanto el error como la varianza de la expresión original pero no es una función del número de Reynolds por lo que no tiene una forma similar a la función original.

4.1.4.1 Ajuste fino

Se realizó el mismo procedimiento que para el caso anterior (sección 4.1.3.1) esta vez modificando la expresión 4.5. Se varió el numerador, el exponente del denominador y el denominador, sin lograr una disminución simultanea en la media y varianza. Esto se puede ver en la tabla 4.2 que muestra el ajuste realizado.

Tabla 4.2 Resumen del ajuste fino para la expresión evolucionada con MSE con ponderación

Constante	Valor	Media	Varianza
Expresión evolucionada		-7,31908565	2705,38408
Exponente	-1%	-4,58457017	3378,42946
	+1%	-10,2009149	2222,46445
Numerador	-1%	-6,63749918	2779,76109
	+1%	-8,00067212	2637,25578
Denominador	-1%	-8,00755683	2636,59949
	+1%	-6,64424756	2778,99406

4.1.5 Selección de la mejor expresión

Tomando las expresiones de las secciones 4.1.2, 4.1.3 y 4.1.4, se eligió la que tuviese menor media y varianza, y además tiene dependencias que son consistentes con la física del problema. La expresión 4.5 es la expresión finalmente seleccionada que corresponde a la expresión 4.4 incluyendo el ajuste fino.

(4.6) $$factor\ de\ fricción = 4\left(1.021 * (9S)^{(2.81 \cdot S)^{-2}} + \left(\frac{0.3021}{S^2}\right)^2 - 0.6\right) Re^{-0.2655}$$

Esta expresión tiene una media del error en la presión de $-2.2 \cdot 10^{-4}$ y una varianza de 808.8, estos valores son obtenidos usando el conjunto de validación.

En la figura 4.8 se muestra el histograma del error en la presión del modelo fenomenológico usando la expresión evolucionada. Al igual que el modelo fenomenológico original, este tiene valores extremos que si bien están en un rango mucho menor, influyen tanto en el promedio como en la varianza. Al realizar el gráfico para identificar si estos puntos se concentran en alguna parte del espacio de variables de entrada se encontró que los valores de error mayores a 50[Pa] están todos concentrados en separaciones menores a 0.5 y flujos mayores a 100[CFM] como se muestra en la figura 4.9.

Al excluir los puntos del conjunto de validación con separación menores a 0.5 la media del modelo evolucionado disminuye a -1.7[Pa] y la varianza a 14.

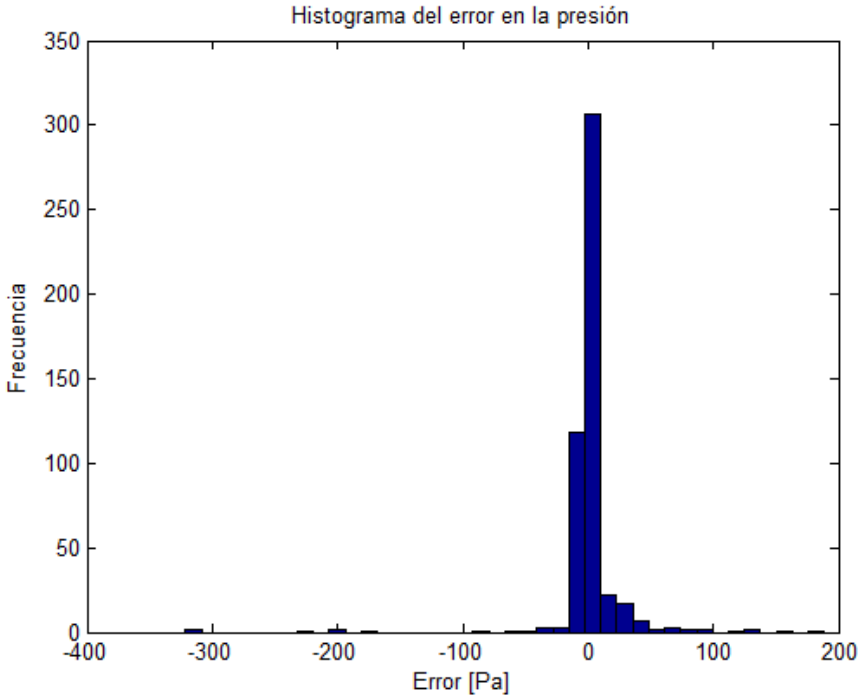

Figura 4.8 Histograma del error en la presión del aire para el modelo fenomenológico evolucionado

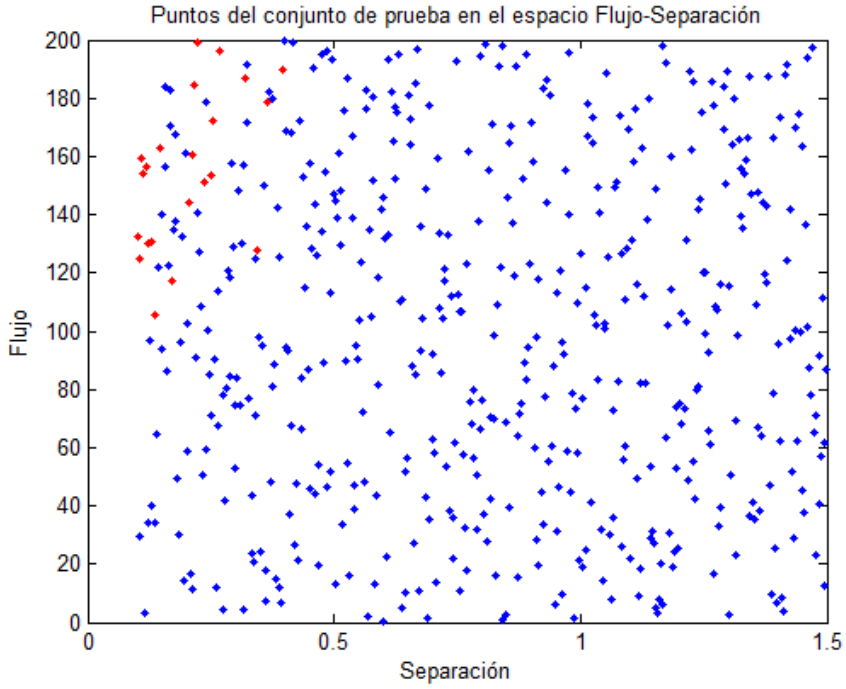

Figura 4.9 Puntos del conjunto de prueba con un modulo del error menor y mayor a 50[Pa] de azul y rojo respectivamente. Usando el modelo fenomenológico evolucionado

4.1.6 Comparación de la expresión evolucionada con el modelo fenomenológico original

En la tabla 4.3 se puede ver la media y la varianza del error en la presión para el modelo fenomenológico original y el que es obtenido al utilizar la expresión evolucionada. Estos valores se obtuvieron a partir del conjunto de prueba.

Tabla 4.3 Media y varianza del error de la presión en modelo fenomenológico original y evolucionado

Modelo	Media	Varianza
Fenomenológico original	56,098	26464,33
Fenomenológico evolucionado	0,62	1163,86

Como se ve de la tabla 4.3 utilizar la expresión evolucionada supone una mejora tanto en la media como en la varianza del error en la presión del aire.

El modelo evolucionado tiene errores tanto positivos como negativos en un rango menor que el del modelo original. Al ser el error lo que se está estudiando, es importante su valor absoluto. Por esto, la distribución del segundo modelo supone una mejora al primero ya que tiene un mayor número de errores cercanos a cero y acerca al origen los puntos que tenían errores muy grandes en el modelo fenomenológico original. En la figura 4.10 se muestran sus respectivos histogramas, en ellos se puede ver la diferencia en rangos así como la mayor cantidad de errores cercanos a cero que tiene el modelo evolucionado.

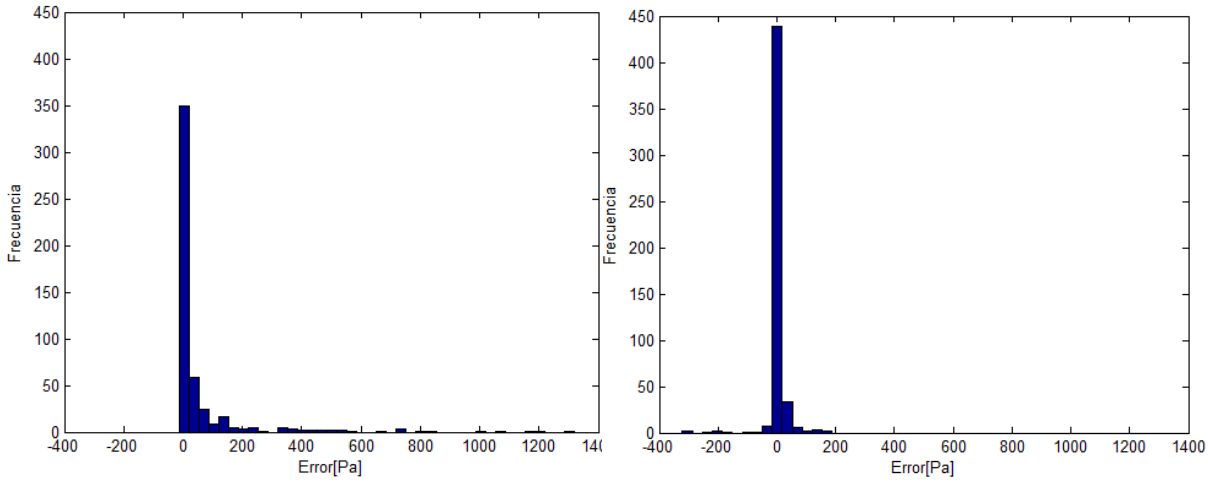

Figura 4.10Histogramas del modelo fenomenológico original (izquierda) y evolucionado (derecha)

Estos gráficos se obtuvieron evaluando los modelos en el conjunto de prueba.

4.2 Coeficiente de arrastre

Se realizaron 4 corridas del algoritmo de programación genética para el coeficiente de arrastre. El objetivo de estas pruebas es corroborar si el operador genético Prune&Plant es de utilidad en el control del bloat. Para esto, dos de las corridas se hicieron con el operador y las otras dos sin él. Ambos casos (con y sin Prune&Plant) se componen de una corrida con función de fitness MSE con regularización (con $\alpha = 0.5$) y una con MSE con ponderación (con $L = 20$). Los valores escogidos en ambos casos fueron basándose en las corridas hechas para el factor de fricción. Se eligió aquel valor que tuviera el promedio del error más bajo tomando en cuenta los 10 mejores individuos y tomando en cuenta además, el tiempo que demora el algoritmo. Los parámetros del algoritmo de programación genética utilizados en estas corridas se encuentran en el apéndice B.2

4.2.1 Modelo original

El coeficiente de arrastre en el modelo original tiene la forma que se indica en la expresión 4.7

(4.7) $$Coeficiente\ de\ arrastre = 3.3481 \cdot Re^{0.179}$$

Como se ve en la expresión 4.7 el coeficiente depende solo del número de Reynolds. Sin embargo, este coeficiente también puede depender de la configuración geométrica del banco, así como de la densidad y velocidad del aire, por lo que la dependencia del modelo se debe a supuestos y simplificaciones.

El error respecto a Ansys en la velocidad del aire usando el conjunto de validación tiene un promedio de -4.9[m/s], una varianza de 117.5 y un rango de 68[m/s]. Este error representa aproximadamente el 6% del rango que toma esta variable de salida por lo que es posible ajustar el coeficiente para obtener valores con menos error en la velocidad del aire.

En la figura 4.11 se muestra el histograma del error en la velocidad del aire evaluando el modelo fenomenológico original en el conjunto de validación. En esta figura se observa que la distribución del error es asimétrica positiva, lo que se corrobora al calcular el *skewness* del error que es 2.3 lo que quiere decir que hay valores más separados de la media a la derecha de este. Además, se puede ver que el promedio está tan a la izquierda que la mayoría de los puntos tienen errores negativos, esto quiere decir que la mayoría de las veces la velocidad que calcula el modelo fenomenológico supera a la obtenida por el CFD ANSYS.

En la figura 4.12 se muestra un gráfico que representa por puntos azules los puntos con error menor a 10[m/s] y con rojo los mayores a 10[m/s], estos puntos están distribuidos en el rango de la variable de la separación entre celdas. En este gráfico se observa que los valores positivos de error están todos concentrados en los puntos con separaciones pequeñas. En las demás variables de diseño los puntos de interés están distribuidos a lo largo de todo su rango por lo que no son de interés.

En este caso no es muy útil intentar reducir el rango de la variable ya que si se reduce el promedio del error disminuirá pero hacia el lado negativo ya que los valores positivos son los que hacen que el modelo tenga un error tan cercano a cero. De eliminarse los puntos con separaciones pequeñas el promedio disminuye a -8.2[m/s]. Lo que es contrario al objetivo que es centrar el promedio en cero.

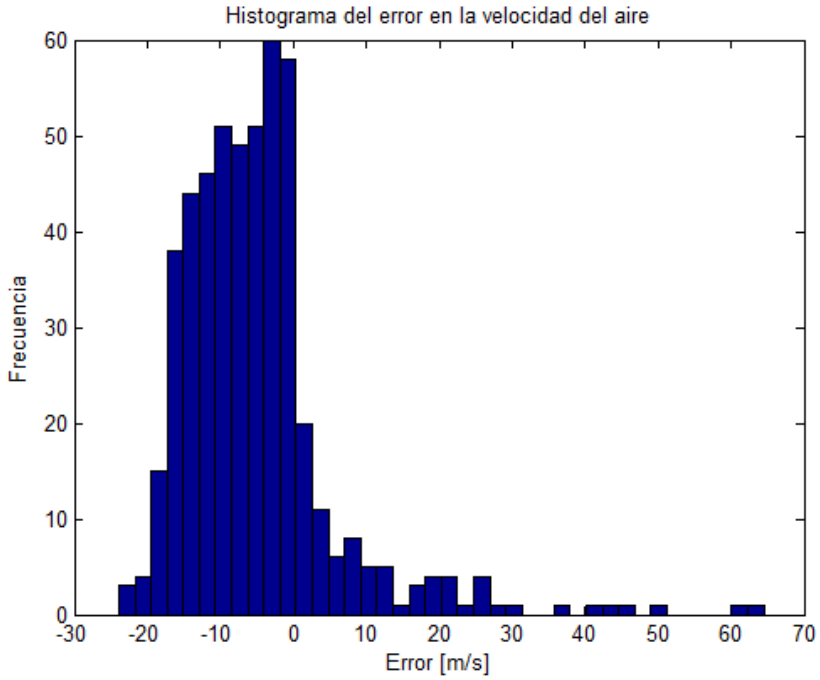

Figura 4.11 Histograma del error en la velocidad del aire del modelo fenomenológico original

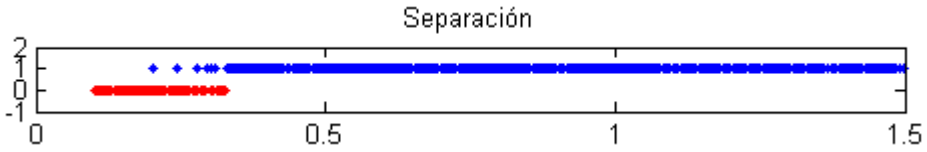

Figura 4.12 Puntos del conjunto de validación con modeulo del error mayor a 10[m/s] (rojo) y menor a 10[m/s] (azul).

4.2.2 Expresión evolucionada con MSE con ponderación

Se corrió el algoritmo de programación genética 2 veces por 30 generaciones debido a que el mejor fitness comienza a converger y por una limitación de tiempo. En la figura 4.13 se muestra la evolución del promedio de la cantidad de nodos por individuo en cada generación. Mientras el algoritmo de programación genética sin *Prune&Plant* llega a tener un promedio de 11 nodos por individuo en la generación 30, usando el operador llega solo a un poco más de 7 nodos por individuo, por lo que se puede decir que en este caso este operador efectivamente ayuda al control del *bloat* pues hace crecer más lentamente a los árboles.

4.2.3 Expresión evolucionada con MSE con regularización

El algoritmo que no utiliza el operador *Prune&Plant*(P&P) se corrió por 30 generaciones mientras que el que si lo utiliza solo corrió 24 por limitaciones de tiempo. En la figura 4.14 se muestra la evolución del promedio de la cantidad de nodos por individuo en cada generación. En éste se observa que en la corrida hecha utilizando el operador P&P los individuos crecen ligeramente más rápido que la hecha sin este operador. Si bien en el caso anterior el operador efectivamente ayudó en el control del *bloat*, en este caso parece hacer lo contrario, por lo que no es posible concluir que sea un operador que controle el *bloat*.

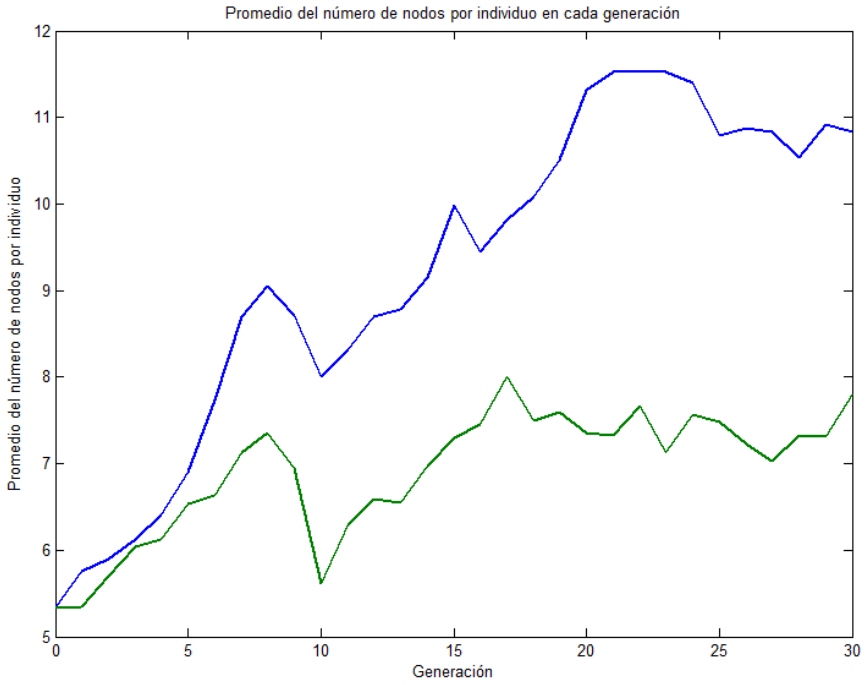

Figura 4.13 Promedio de la cantidad de nodos por individuo en cada generación para corridas con y sin Prune&Plant (MSE con ponderación)

.

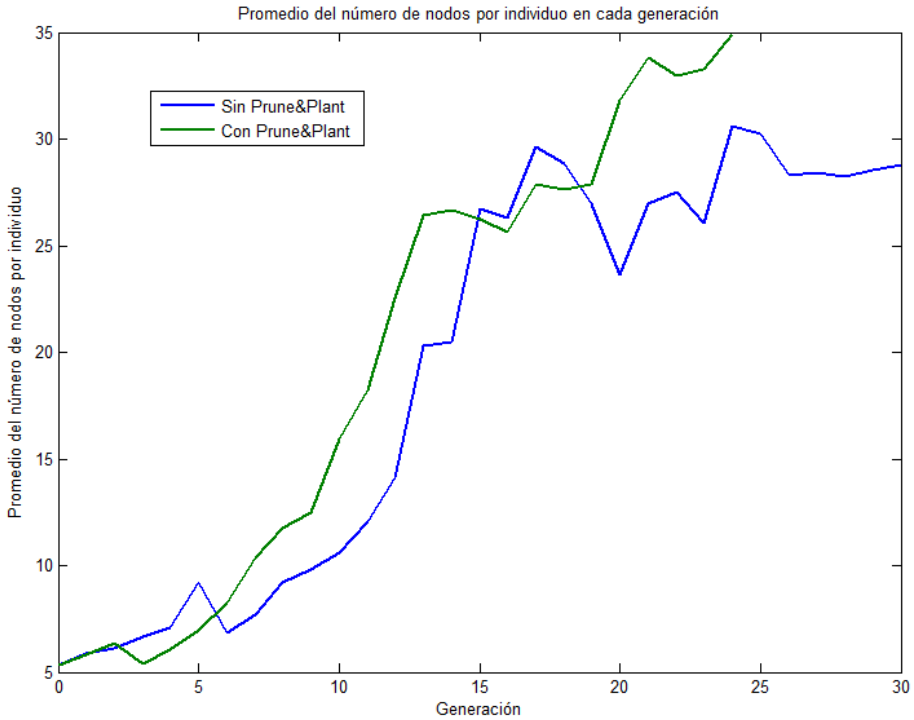

Figura 4.14 Promedio de la cantidad de nodos por individuo en cada generación para corridas con y sin Prune&Plant (MSE con regularización)

4.2.4 Selección de la mejor expresión

Se evaluaron los 20 mejores individuos de las cuatro corridas utilizando el conjunto de validación. Se encontró que todas expresiones evolucionadas con MSE con ponderación y muchas de las correspondientes a la otra función de fitness tenían puntos del conjunto en los que no convergía el modelo. De los 80 individuos estudiados solo 17 convergieron para todos los puntos del conjunto de validación. De estos, el individuo con mejor promedio del error se muestra en la expresión 4.8 cuya media es de -0.83 [m/s], varianza de 25 y un rango de 46.9[m/s].

(4.8) $$Coeficiente\ de\ arrastre = 4.78\left(0.636\left(4.81\,R^{2.3\cdot 0.49^{A}}\right)^{D^{\frac{A}{Re}}}\right)^{0.0187}$$

Donde Re es el número de Reynolds, D es una densidad del aire dividida en una densidad de referencia y A es la razón entre el área superficial de la celda y el área por donde pasa el fluido. Esta expresión agrega información sobre la densidad del fluido y la geometría del banco, y sigue dependiendo del número de Reynolds.

Al examinar el conjunto de validación se encontró que todos los individuos fallan para valores de flujo menores a 10 [CFM]. Al excluir los 25 puntos que tenían flujo en ese rango, todos los individuos convergieron en todo el conjunto de validación. En los apéndices C.4 y C.5 se encuentran las tablas con la media, varianza y cantidad de nodos de los mejores 20 individuos de las corridas con MSE con ponderación y regularización respectivamente. La media y varianza para el error en la velocidad del aire del modelo fenomenológico original pasaron a ser -5.2[m/s] y 122 respectivamente.

El mejor individuo encontrado tiene una media de -0.1[m/s], una varianza de 26.2, rango de 46[m/s]y un árbol de 26 nodos, cuya expresión simplificada es la 4.9

(4.9) $$Coeficiente\ de\ arrastre = 0.17\left(\frac{Re^{0.21}}{\left(\frac{D_f}{D_{ref}}\right)^{2}}\right)^{0.16}$$

Donde Re es el número de Reynolds, D_f es la densidad del fluido y D_{ref} es una densidad de referencia. Esta expresión además de depender del número de Reynolds como lo hace la expresión 4.7, depende de la densidad del aire lo que le agrega más información al coeficiente.

Finalmente se selecciono la ecuación 4.8 ya que tiene una menor varianza y rango. Además esta ecuación es útil para el rango completo de las variables de entrada a diferencia de la expresión 4.9 que debió acotar el rango de una variable para obtener buenos resultados.

En la figura 4.14 se muestra el histograma del error en la velocidad para el modelo evolucionado seleccionado. Al igual que en el caso del modelo original, los valores más alejados del cero son asociados a una separación muy pequeña (menores a 0.1) como se muestra en la figura 4.16 donde los puntos rojos son aquellos puntos del conjunto de validación con los que se obtiene un error superior a 15[m/s].

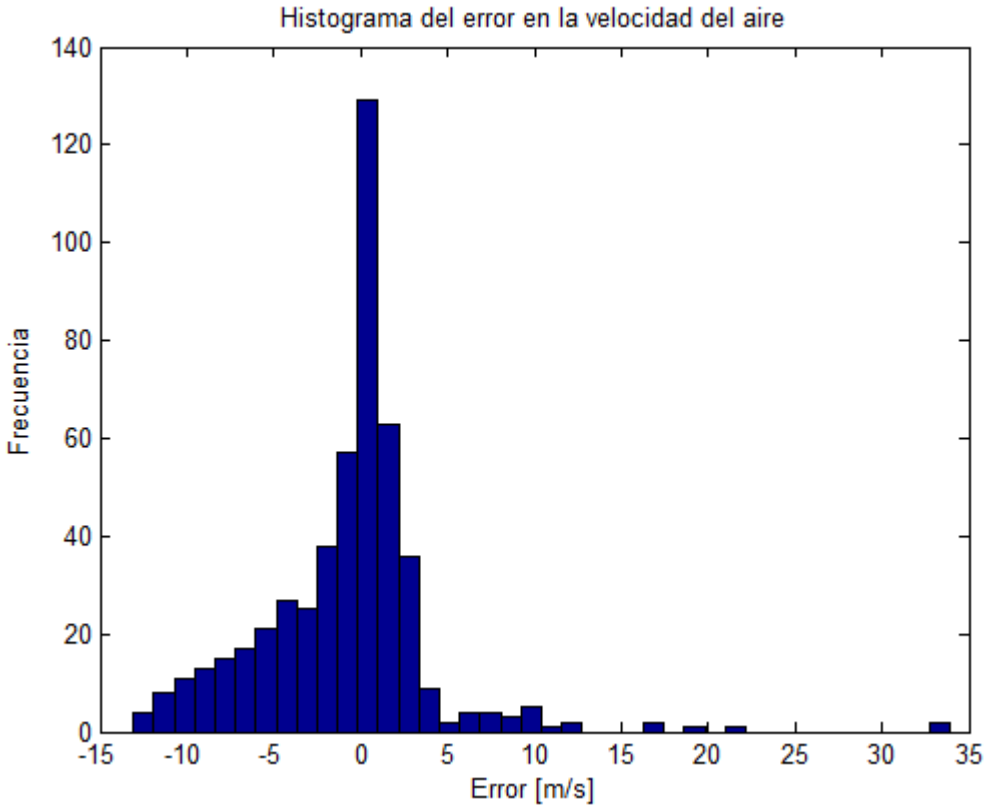

Figura 4.15 Histograma del error en la velocidad del aire para el modelo evolucionado

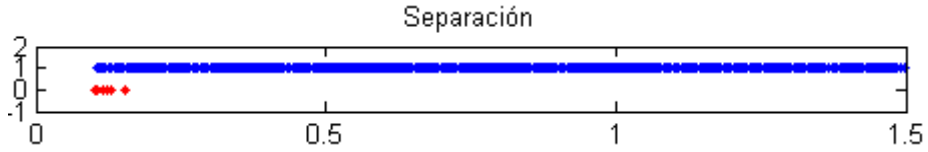

Figura 4.16 Puntos del conjunto de validación con error en la velocidad menor a 15[m/s] (azul) y mayor(rojo).

4.2.4.1 Ajuste fino

Se realizó un ajuste fino a la expresión 4.8 con el fin de obtener una reducción en la media y varianza. Para esto se siguió el procedimiento descrito en 4.1.3.1. Se varió el exponente 0.00187, y el coeficiente 4.78 que multiplica a toda la expresión. En la tabla 4.4 se muestra un resumen del ajuste hecho. La tabla completa se encuentra en el apéndice D.

Variando ambos parámetros de la expresión es posible obtener un una mejora tanto en la media como en la varianza del error en la velocidad. La mayor disminución en ambos casos fue al reducir en 20.7% el exponente obteniendo una media de de 0.0004 [m/s] y una varianza de 23.76 por lo que finalmente la expresión seleccionada es modificada tomando en cuenta esta mejora.

Tabla 4.4 Resumen del ajuste fino para la expresión seleccionada como coeficiente de arrastre

Constante	Valor	Media	Varianza
Expresión original		-0,84	25,01
Exponente	-1%	-0,797	24,93
	+1%	-0,879	25,09
	-20.6%	-0,002	23,79
	-20.7%	0,000	23,76
	-20.8%	0,004	23,76
Coeficiente 4.78	-1%	-0,52	24,84
	+1%	-1,15	25,30
	-2.6%	-0,02	24,80
	-2.7%	0,01	24,81
	-2.8%	0,04	24,82

4.2.5 Comparación de la expresión evolucionada con el modelo fenomenológico original

Finalmente para comparar el modelo seleccionado con el modelo fenomenológico original se utiliza el conjunto de prueba. Con este se hizo el histograma de los errores para ambos modelos, el que se muestra en la figura 4.17. Además, la media y varianza de los modelos se muestran en la tabla 4.5.

Efectivamente el modelo evolucionado supera en media y varianza al modelo fenomenológico original. Además, la distribución de su error tiene un pico más alto lo que es bueno ya que el objetivo ideal es lograr que todos los errores estén concentrados en cero.

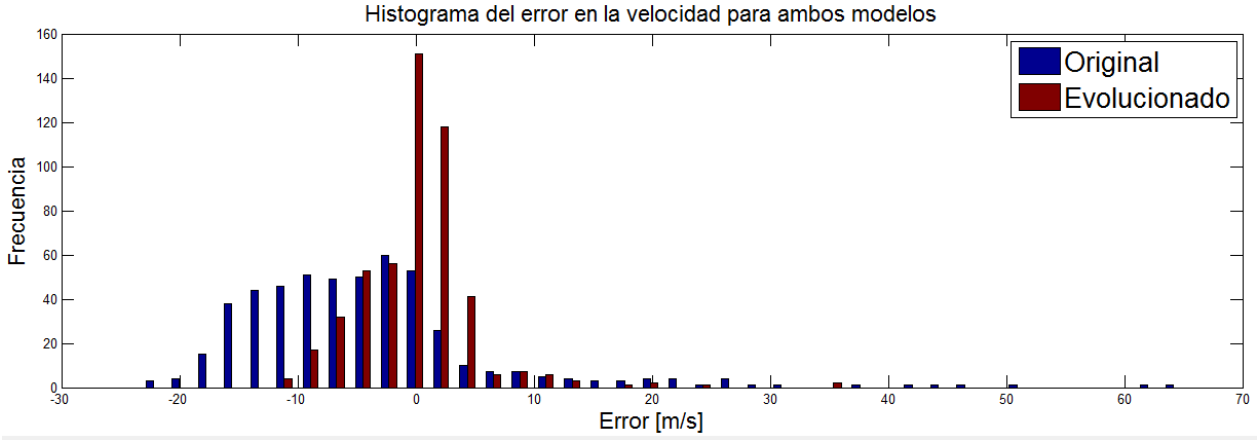

Figura 4.17 Histograma del error en la velocidad del modelo fenomenológico original (azul) y evolucionado (rojo)

Tabla 4.5 Media y varianza del error en los modelos usando el conjunto de prueba

Modelo	**Media**	**Varianza**
Fenomenológico original	-4.278	119
Fenomenológico evolucionado	-0.08179	21.760

4.3 Número de Nusselt

Se realizaron 3 corridas de programación genética usando la función de fitness de MSE con regularización cambiando en cada una el valor del parámetro α. Los valores utilizados fueron 0.3, 0.4 y 0.5. Todos los parámetros del algoritmo de programación genética utilizados en estas corridas se encuentran en el apéndice B.3

4.3.1 Modelo original

En el modelo fenomenológico original el número de Nusselt tiene una expresión general como la mostrada en (4.13)

$$Nusselt = C \cdot Re^{0.653} Pr^{0.36} \left(\frac{Pr}{Pr_s}\right)^{0.25} \tag{4.13}$$

Sin embargo, en el término Pr y Pr_s en el caso de estudio son constantes a sí que pueden ser incluidos en el término constante C quedando la expresión (4.14)

$$Nusselt = C' \cdot Re^{0.653} \tag{4.14}$$

En la expresión (4.14) el valor de C' varía en función del rango en que se encuentre el número de Reynolds y en la posición del banco en que se esté midiendo el número de Nusselt.

La temperatura de la celda central tiene un error medio de -4.7[°C] y una varianza de 56.6 lo que es un valor muy alto ya que un banco de baterías en operación varía su temperatura entre los 20°C, cuando está a temperatura ambiente y sin carga ni descarga, y los 40°C aproximadamente, cuando está en operación. Esto hace que el error sea casi un cuarto del rango de la temperatura de una celda en funcionamiento, por lo que esta media de error no es aceptable para el modelo.

En la figura 4.18 se muestra el histograma de error en la temperatura de la celda central para el modelo fenomenológico original. En este se observa que no hay errores positivos, esto quiere decir que el software CFD ANSYS siempre entrega valores de temperatura mayores a los que entrega el modelo fenomenológico. En el caso de la presión pasaba lo contrario, el CFD siempre entregaba valores menores (o la mayoría de las veces). ANSYS al tener presiones más pequeñas, se espera que las velocidades sean más altas y por lo tanto enfrié más la celda, por lo que las temperaturas de las celdas serán menores a las del modelo. Siguiendo este razonamiento, es consistente tener solo errores negativos.

La distribución mostrada en la figura 4.18 tiene gran similitud con una *power law*. Para saber si es posible aproximar el error por esta distribución se graficó el error en la temperatura en función de la temperatura obtenida por el modelo fenomenológico original, ambas en escala logarítmica. De ser una power law se espera que este gráfico muestre una recta. Como se muestra en la figura 4.19, si bien es posible observar un comportamiento lineal, la función tiene mucho ruido como poder sacar conclusiones.

La figura 4.20 muestra los puntos del conjunto de validación con un error mayor a 25[°C] de color rojo. Esto es para identificar si dichos puntos pueden ser aislados. Como se observa en esta figura, estos errores se concentran en altas corrientes y flujos bajos. En este punto de operación se espera que las celdas tengan altas temperaturas ya que la corriente hace que la celda genera calor y el flujo bajo hace que la transferencia de calor con el aire sea lenta. El gran error presentado en torno a este punto de operación puede deberse a que a altas temperaturas la generación de calor se

deba a causas distintas a la perdida por la resistencia interna, como por ejemplo por reacciones químicas.

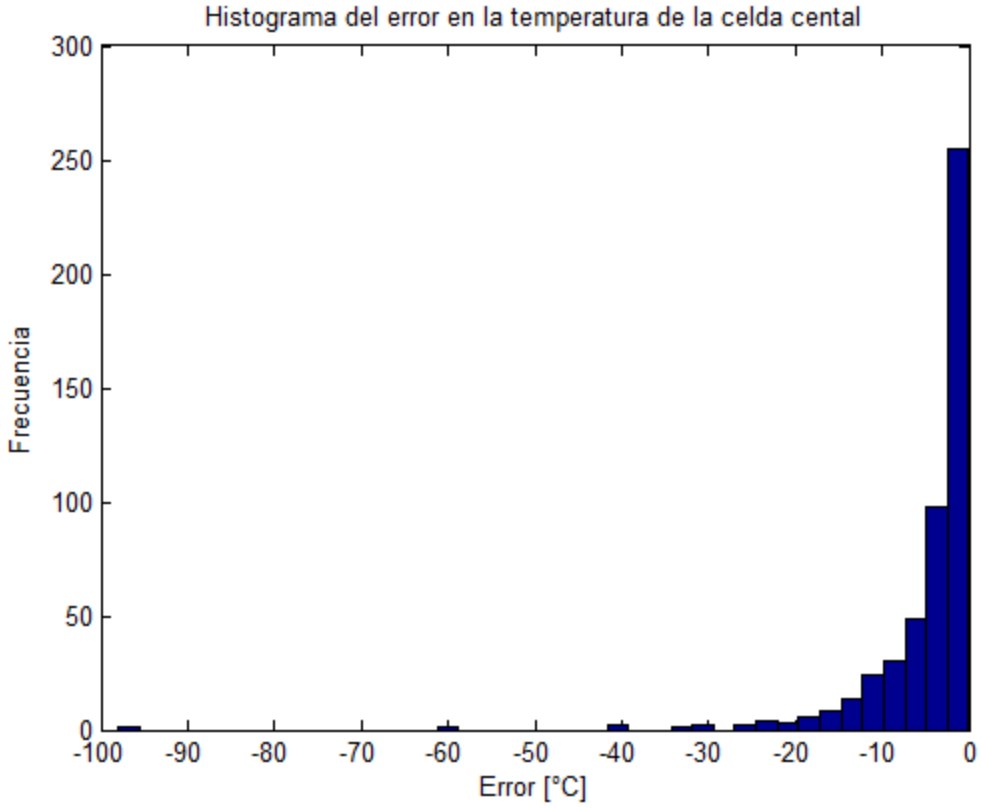

Figura 4.18 Histograma del error en la temperatura de la celda central para el modelo fenomenológico original

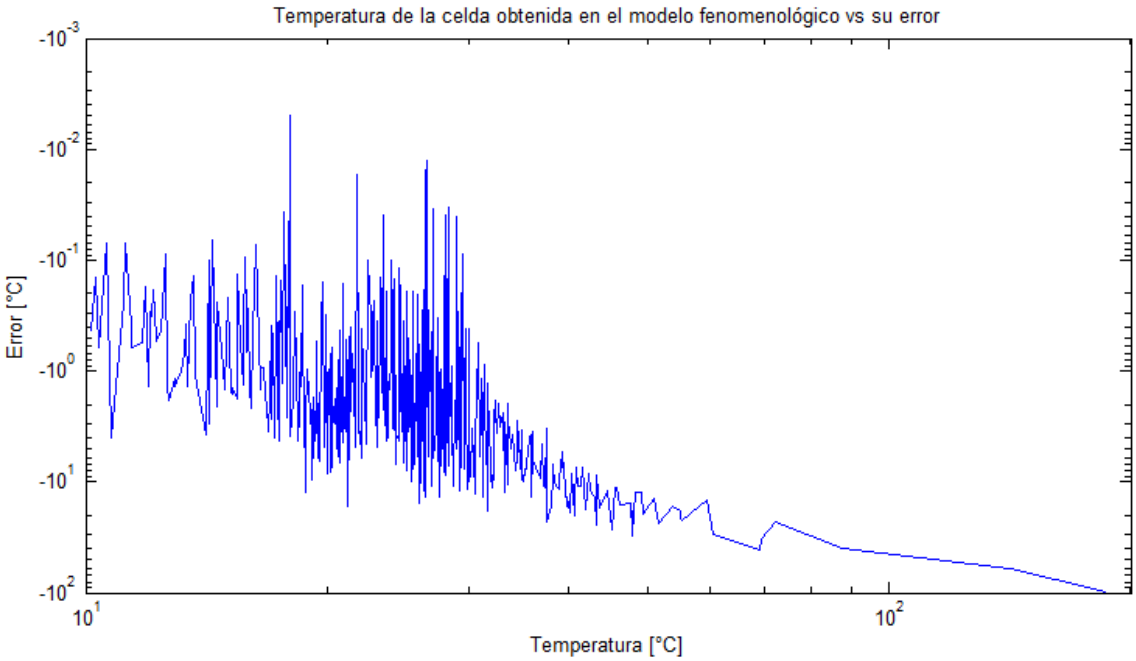

Figura 4.19 Temperatura obtenida por el modelo fenomenológico vs su error respecto a ANSYS

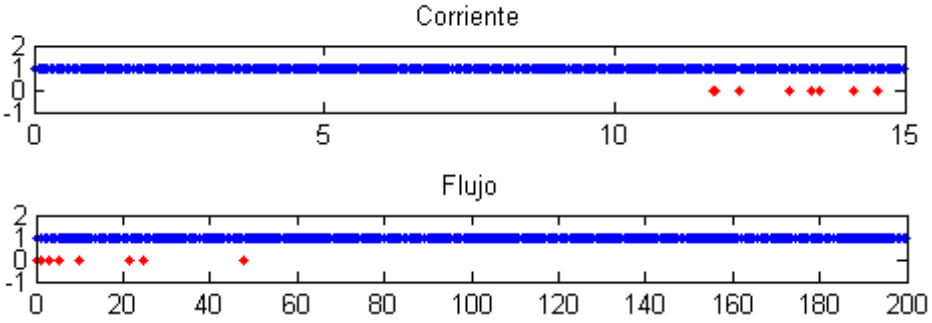

Figura 4.20 del conjunto de validación con un error mayor a 25[°C] (rojo) y menores (azul)

4.3.2 Evolución de la expresión

Se corrió el algoritmo de programación genética 3 veces utilizando como función de fitness MSE con regularización con valores de alfa 0.3, 0.4 y 0.5 durante 21, 28 y 30 generaciones respectivamente. Las 3 corridas se hicieron en el mismo tiempo (4 días), la diferencia en la cantidad de generaciones se debe a que a menor alfa, los arboles crecen más rápido por lo que aumenta el costo computacional de operar con ellos.

4.3.3 Selección de la mejor expresión

Para seleccionar la mejor expresión de estas 3 corridas del algoritmo de programación genética se tomaron en cuenta los 20 mejores individuos de cada corrida. La media, varianza y número de nodos de cada uno de estos individuos se encuentra en el apéndice C.6.

La mayoría de los individuos encontrados tienen solo 3 y 5 nodos y comparten la misma forma que la de la expresión 4.14 pero con distintos valores de las constantes. En el caso de los individuos de 3 nodos son solo el número de Reynolds elevado a una constante.

Además de estos individuos hay otro conjunto de individuos mucho más pequeño que tiene una gran cantidad de nodos. Éstos además de ser funciones tienen valores negativos en algunos puntos por lo que no son soluciones factibles ya que el número de Nusselt es siempre positivo.

La expresión 4.15 es la que obtiene el mejor promedio en el error de la temperatura y es también una de las más simples. Su media es de 0.33[°C], varianza 10.1 y tiene solo 3 nodos.

(4.15) $$Nusselt = Re^{0.677}$$

Donde Re es el número de Reynolds.

En la figura 4.21 se muestra el histograma del error en la temperatura de la celda central obtenida por el modelo evolucionado, donde se observa que la distribución está centrada en cero y tiene un rango menor a la del modelo fenomenológico original.

Este modelo tiene solo 5 errores cuyo modulo es mayor a 10[°C]. En la figura 4.22 se puede ver que estos valores corresponden a valores de flujo muy bajos.

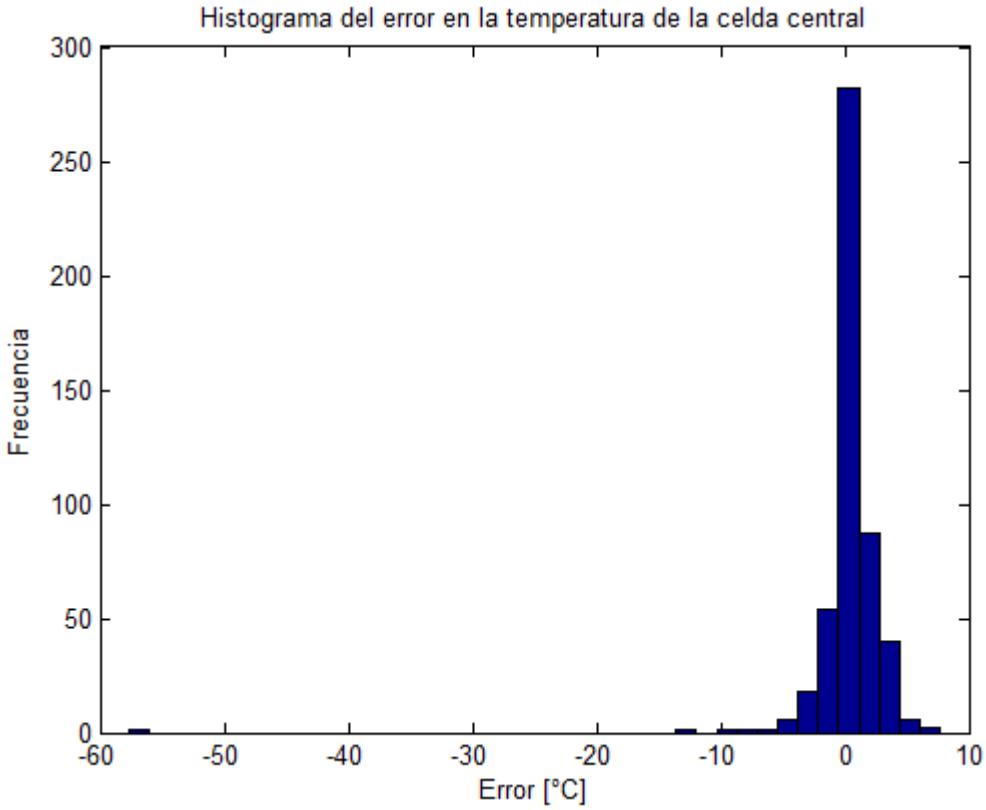

Figura 4.21 Histograma del error en la temperatura de la celda central del modelo evolucionado

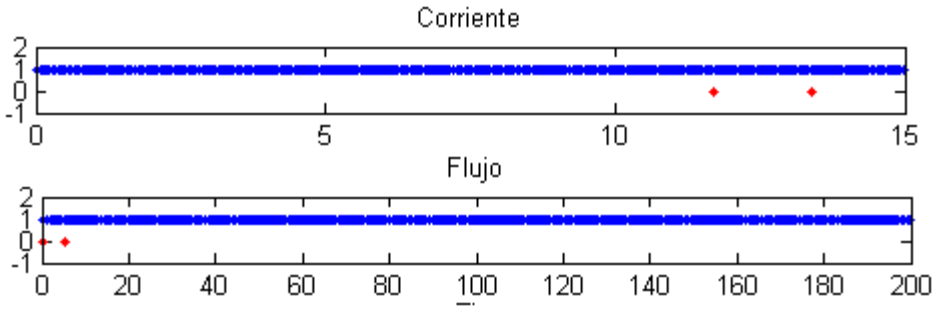

Figura 4.22 Puntos del conjunto de validación cuyo modulo del error es mayor a 10[°C] (rojo) y menor (azul)

4.3.3.1 Ajuste fino

Se hizo un ajuste fino de la expresión seleccionada. Para esto se varió el exponente del número de Reynolds y el coeficiente que lo multiplica. En la tabla 4.6 se puede ver lo que le ocurre a la media y varianza del error al variar en un 1% cada uno de los parámetros recién mencionados. En ambos casos, al aumentar o disminuir el parámetros en un 1% la media y la varianza tienen un comportamiento contrario por lo que no el ajuste no es de utilidad ya que se busca que ambas disminuyan.

Tabla 4.6 Resumen del ajuste realizado a los parametros de la expresión del número de Nusselt evolucionada

Constante	Valor	Media	Varianza
Expresión original		0,34	10,11
Exponente	-1%	0,25	10,69
	+1%	0,42	9,59
Coeficiente	-1%	0,32	10,29
	+1%	0,35	9,94

4.3.4 Comparación de la expresión evolucionada con el modelo fenomenológico original

Para comparar el modelo fenomenológico original con el evolucionado se utilizó el conjunto de prueba. Con este se hizo la tabla 4.7 que muestra la media y varianza de ambos modelos.

Tabla 4.7 Media y varianza del error en los modelos usando el conjunto de prueba

Modelo	**Media**	**Varianza**
Fenomenológico original	-4.646	41.87
Fenomenológico evolucionado	0.4	5.63

En la figura 4.23 se muestra el histograma del error para ambos modelos. En este se observa que el modelo evolucionado es efectivamente una mejora ya que disminuye el rango del error y tiene más errores en la cercanías de cero.

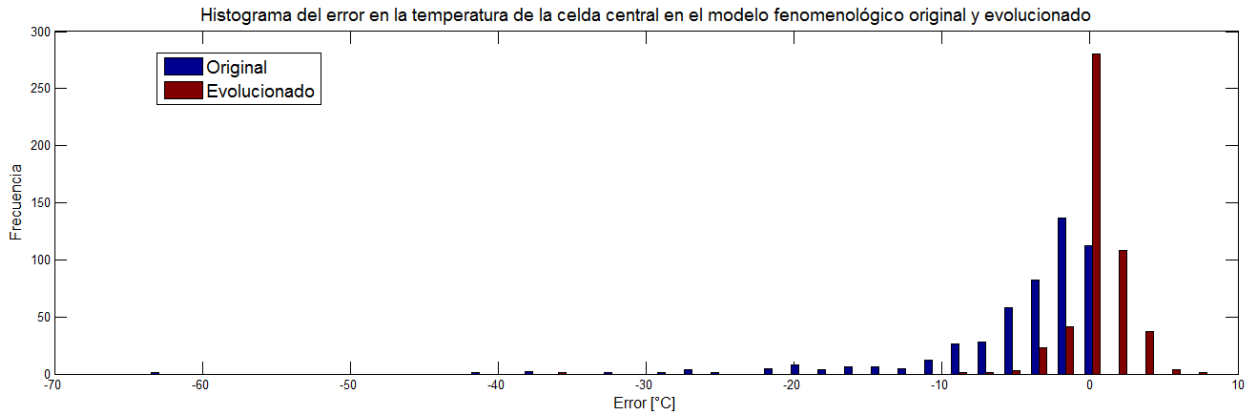

Figura 4.23 Histograma del error de la temperatura de la celda central para el modelo fenomenológico original (azul) y evolucionado (rojo)

Capítulo 5

Conclusiones

En esta memoria se describe una metodología para ajustar expresiones dentro de un modelo fenomenológico que describe el comportamiento térmico de bancos de baterías. Este ajuste fue llevado a cabo utilizando un algoritmo evolutivo de la familia de los algoritmos genéticos: programación genética. Con esta herramienta se ajustaron un total de tres expresiones del modelo: factor de fricción, coeficiente de arrastre y número de Nusselt, sobre las cuales se hizo un análisis comparativo entre la expresión original y las expresiones obtenidas utilizando la herramienta mencionada y un análisis numérico de las expresiones ajustadas.

En la primera parte de este trabajo se describió la situación actual del diseño de empaquetamiento de baterías mediante algoritmos evolutivos y el uso de modelos sustitutos para esto. El uso de modelos sustitutos está siendo muy utilizado actualmente debido a que cada vez se abordan problemas más complejos utilizando algoritmos evolutivos y construir o simular cada caso no es posible debido al tiempo que demora. El uso de estos modelos ha permitido reemplazar parcialmente los software CFD, que demoran mucho tiempo por simulación, haciendo posible la solución de problemas que sin estos serían impracticables por la gran cantidad de tiempo demandado. Existen muchos métodos para diseñar modelos sustitutos, algunos de ellos asumen conocida la forma que tiene la relación de entrada y salida, otros no, pero ninguno de estos se preocupa de las características físicas que pueda tener la expresión a la que llegan. Diseñar modelos sustitutos que tengan una interpretación física es un aporte importante ya que se puede obtener más información y se tiene la posibilidad de generalizar estos modelos.

Para lograr la implementación del sistema que ajuste las expresiones del modelo se escogió la herramienta de programación genética. En particular se utilizó el toolbox de Matlab GPlab. Éste algoritmo evolutivo tiene una serie de parámetros que intervienen en el entrenamiento del sistema de muchas formas distintas, por esto, lo primero que se hizo fue estudiar problemas más simples de regresión simbólica para familiarizarse con el algoritmo y sus parámetros. Luego, teniendo algunos parámetros definidos, por esta primera etapa, se decidieron las expresiones a ajustar del modelo, las que tenían como característica común que todas fueron obtenidas de manera experimental.

Para obtener los datos con los que se entrenó y probó el ajuste se utilizó el software de simulación ANSYS, donde se simuló un bloque de 5 celdas escalonadas. Esta elección de banco es debido a que el modelo fenomenológico utiliza un bloque como el recién descrito para sus cálculos y si se requiere un banco más grande simplemente se concatenan varios bloques.

Para definir las configuraciones que se simularon se utilizó un método llamado *'latin-hypercube'* (LH), que obtiene muestras del espacio de las variables de diseño de una manera uniforme, por lo que cubre todo el espacio. Se hicieron tres conjuntos con este método, el de entrenamiento,

prueba y validación, cada uno de 500 puntos del espacio de las variables de diseño. Se hicieron tres LH distintos porque se necesita que estos tres conjuntos representen el espacio completo.

A la herramienta de programación genética se le hicieron las modificaciones necesarias para que las expresiones quedaran en función de los parámetros correspondientes y que las unidades de los resultados fueran las adecuadas, esto incluye modificar la inicialización de los individuos, sus terminales y funciones, operadores genéticos y función de fitness.

Se incluyó un operador genético llamado Prune&Plant cuya función es evitar el bloat en el algoritmo, se comparó el algoritmo con y sin este operador para comprobar su eficacia. Se utilizaron tres funciones de fitness en distintas corridas del algoritmo, estas fueron el error cuadrático (MSE), el MSE con regularización y MSE con una ponderación. Esta elección fue hecha debido a que el objetivo es que el error del modelo con respecto al software sea cero y tiene la característica que mientras más alejado un punto está del valor deseado más lo castiga pero es muy sensible a valores atípicos. La ponderación y regularización tienen la finalidad de controlar el tamaño de los árboles para evitar el *bloat*.

Se estudió la distribución de los errores tanto en el modelo fenomenológico original como en los evolucionados. Finalmente se compararon estos dos modelos utilizando promedio, varianza y sus histogramas. Al hacer esto se observó que el error en la presión del modelo fenomenológico original sigue una distribución muy similar a una *power law* por lo que no se le puede comparar al modelo evolucionado utilizando la media y varianza, sin embargo, se puede decir que el este último tiene un mejor desempeño ya que disminuyó el rango del error, concentrándolo en las cercanías de cero y haciendo que la distribución tenga una forma más parecida a un impulso que el objetivo ideal. Por otro lado, se descubrió que los errores muy grandes del modelo fenomenológico original están asociados a separaciones pequeñas y flujos grandes por lo que si se restringe el espacio del espacio de variables de diseño, su desempeño debería mejorar.

Al evolucionar la expresión del coeficiente de arrastre se encontró que el operador Prune&Plant ayuda en el control del *bloat* en el caso de la función de fitness MSE con ponderación, mientras que no se notó una mejora en la función MSE con regularización por lo que no se puede concluir que este operador efectivamente sea un aporte en el control del *bloat*.

La expresión evolucionada del coeficiente de arrastre también supone una mejora al modelo fenomenológico original ya que pasa de un error promedio de -4.2[m/s] a -0.08[m/s] y de una varianza de 119 a 21, por lo que los errores están distribuidos en un rango mucho menor.

El caso del número de Nusselt resultó muy similar al del factor de fricción debido a la distribución que sigue el error en la temperatura. Pero al igual que en el otro caso, la expresión encontrada mejoró el rendimiento del modelo disminuyendo el rango del error y mejorando su promedio.

Las expresiones encontradas tienen las unidades correspondientes y cumplen con que las dependencias son de parámetros que tienen un sentido físico. En el caso del factor de fricción y número de Nusselt se encontró una expresión que es ajuste a la expresión original conservando la forma de esta mientras que para el coeficiente de arrastre se encontró una que si bien no guarda similitud con la expresión original, depende de parámetros que hacen sentido.

En los tres casos anteriores se encontró que el modelo tenía errores muy elevados en ciertos lugares del espacio del variables diseño muy específicos. Por esto, si se reduce el espacio de

variables, en particular si la separación entre celdas comienza desde 0.5 veces el diámetro de la celda y el flujo de entrada comienza a partir de los 10[CFM], el desempeño tanto del modelo fenomenológico original como el del evolucionado se verá beneficiado.

En resumen, la metodología empleada en este trabajo sirvió para ajustar las tres funciones objetivos por lo que aumenta la exactitud del modelo fenomenológico obteniendo errores promedio de presión, velocidad y temperatura menores a 1[Pa], 1[m/s] y 1[°C] respectivamente. Si bien no se logró que éste tuviese las mismas salidas que el software de simulación, está más cercano a este y dado que el objetivo es hacer del modelo una aproximación y no un reemplazo al software se puede decir que la metodología cumplió su objetivo. Estos resultados ayudan a que el modelo pueda ser utilizado como modelo sustituto del software de simulación lo que es muy importante ya que contar con un modelo sustituto en algoritmos genéticos de optimización disminuye drásticamente el tiempo que demora en llegar a un resultado. Además la herramienta utilizada es modular y muy sencilla de editar por lo que es posible seguir mejorando la metodología ajustando más parámetros o cambiando algunas funciones.

5.1 Trabajos Futuros

Existen diversos enfoques para poder mejorar los resultados obtenidos. El primero es aumentar el número de datos de entrenamiento para que se pueda describir de mejor manera el espacio de variables de entrada y de esta forma se pueda ajustar mejor, sin embargo, hay que encontrar un equilibrio entre el número de datos y el tiempo que demora el algoritmo en obtener resultados aceptables. Otra forma es explorar otras funciones de fitness que no asuman distribuciones en los datos como el criterio de mínima entropía del error o el criterio de máxima correntropía del error. También se debe considerar la posibilidad de disminuir los rangos en los que es válido el modelo ya que es posible que los fenómenos físicos cambien en diversos rangos y se vio que los modelos ajustados no funcionan muy bien en algunos extremos de los rangos de las variables de diseño. Otra alternativa a lo anterior es entrenar una expresión por separado para utilizarla en dichos rangos.

En cuanto a la implementación, si se requiere de mayor rapidez una mejora considerable en tiempo puede ser implementar el algoritmo en los lenguajes de programación C o C++, ya que al ser lenguajes de menor nivel, son más rápidos, lo que permitirá que se ahorre tiempo en el algoritmo. Además se pueden implementar bosques de árboles, esto se refiere a los individuos en el algoritmo de programación genética. Al tener un individuo como un conjunto de árboles, este puede representar las tres expresiones requeridas simultáneamente y dado que el modelo es iterativo donde las variables dependen unas de otras, puede ser mejor intentar ajustar las tres expresiones al mismo tiempo. El problema de esta implementación es que el tiempo que demore puede crecer rápidamente debido a la gran cantidad de combinaciones que se pueden hacer con árboles tres veces más grandes que con los que se trabajó en esta memoria.

Bibliografía

AB, O., 1994. Controlling correlations in latin hypercube samples. *J Stat Assoc,* p. 1517–1522.

Alfaro-Cid, E., Esparcia-Alcázar, A. & Sharman, K., 2008. Prune and Plant: A New Bloat Control Method for Genetic Programming. *Eighth International Conference on Hybrid Intelligent Systems.*

Alfaro-Cid, E. y otros, 2010. Bloat control operators and diversity in Genetic Programming: a comparative study. *Evolutionary Computation,* pp. 305-332.

Arora, P., White, R. & Doyle, M., 1998. Capacity fade mechanisms and side reactions in lithium-ion batteries. *Jurnal of the Electrochemical Society 145.*

Beltrán, R., 2008. *Complejidad de modelos: Sesgo y Varianza,* s.l.: s.n.

Bianchi, E., 2008. Elementos de Electroquímica: Electrólisis y acumuladores reversibles.

Cai, W., Pacheco-Vega, A., Sen, M. & K.T., Y., 2006. Heat Transfer Correlations by Symbolic Regression. *International Journal of Heat and Mass Transfer,* pp. 4352-4359.

Diaz-Casas, V., Becerra, J., Lopez-Pena, F. & Duro, R., 2013. Wind turbine design throug evolutionary algorithms based on surrogate CFD methods. *Optimization and Engineering 14.*

Fogel, L., Owens, A. & Walsh, M., 1966. Artificial Intelligence through Simulated Evolution. New York. *John Wiley.*

Fox, R., Pritchard, P. & McDonald, A., 2009. *Introduction to Fluid Mechanics.* 7 ed. s.l.:Wiley.

Gupta, D. & Ghafir, S., 2012. An Overview of methods maintaining Diversity in Genetic Algorithms. *International Journal of Emerging Technology and Advanced Engineering,* 2(5).

Hedayat, A., Sloane, N. & Stufken, J., 1999. Orthogonal arrays: theory and applications. *Springer, Series in Statistics, Berlin.*

Jin, Y., Olhofer, M. & Sendhoff, B., 2000. On Evolutionary Optimization with Approximate Fitness Funtions. *Genetic and Evolutionary Computation Conference,* pp. 786-793.

Jones, D., Schonlau, M. & Welch, W., 1998. Expensive global optimization of expensive black-box functions. *J Global Optim,* p. 455–492.

Khatir, Z. y otros, 2013. Multi-objective Computational Fluid Dynamics (CFD) design optimization in commercial bread-baking. *Applied Thermal Engineering 60.*

Kim, S. & Kim, K., 2014. Optimization of a Hybrid Double-Side Jet Impingement Cooling System for High-Power Light Emitting Diodes. *Journal of Electronic Packaging 136.*

Kinzett, D., 2010. Investigation of simplification threshold and noise level of input data in numerical simplification of genetic programs. *Evolutionary Computation (CEC).*

Kinzett, D., Johnston, M. & Zhang, M., 2009. Numerical Simplification for Bloat Control and Analysis of Building Blocks in Genetic Programming. *Evolutionary Intelligence,* pp. 151-168.

Korns, M., 2007. Large-scale, Time-constrained Symbolic Regression. En: *Genetic Programming Theory and Practice IV.* s.l.:Springer, pp. 299-314.

Kotti, M. y otros, 2014. Generation of surrogate models of Pareto-optimal performance trade-offs of planar inductors. *Analog Integrated Circuits and Signal Processing 78.*

Koza, J., 1992. Genetic Programming: On the Programming of Computers by Means of Natural Selection. *MIT Press.*

Koza, J. y otros, 2003. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence.* s.l.:Springer.

Koziel, S. & Ogurtsov, S., 2013. Multi-Objective Design of Antennas Using Variable-Fidelity Simulations and Surrogate Models. *IEEE Transactions on Antennas and Propagation 61*.

Leary, S., Bhaskar, A. & Keane, A., 2003. Optimal orthogonal array-based latin hypercubes. *Journal of Applied Statistical Science,* p. 585–598.

Lian, Y., Oyama, A. & Liou, M., 2010. Progress in design optimization using evolutionary algorithms for aerodynamic problems. *Progress in Aerospace Science 46.*

Lim, D., Jin, Y., Ong, Y. & Sendhoff, B., 2010. Generalizing Surrogate-Assisted Evolutionary Computation. *IEEE Transactions on Evolutionary Computation 14.*

Luke, S. & Panait, L., 2002. Lexicographic Parsimony Pressure. *Proceedings of GECCO.*

McKay, M., Conover, W. & Beckman, R., 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics,* pp. 239-245.

Mengistu, T. & Ghaly, W., 2008. Aerodynamic optimization of turbomachinery blades using evolutionary methods and ANN-based surrogate models. *Optimization and Engineering 9.*

Moser, F., 2015. *Medición Experimental del Comportamiento Térmico de una Batería de Ión-Litio,* s.l.: s.n.

Nghia, M. y otros, 2013. Evolution by Adapting Surrogates. *Evolutionary Computation 21.*

Nguyen, A., Reiter, S. & Rigo, P., 2014. A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy 113.*

Oksuz, O. & Akmandor, I., 2010. Multi-Objective Aerodynamic Optimization of Axial Turbine Blades Using a Novel Multilevel Genetic Algorithm.. *Jurnal of Turbomachinery-Transactions of the Asme 132.*

Patakar, S. & Spalding, D., 1972. Calculation procedure for heat, mass and momentum-transfer in 3-dimensional parabolic flows. *International Journal of Heat and Mass Transfer 15.*

Pesaran, A., Kim, G. & Keyser, M., 2009. Integration Issues of Cells into Battery Packs for Plug-In and Hybrid Electric Vehicles. *EVS-24 International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium.*

Pilat, M. & Neruda, R., 2013. Aggregate meta-models for evolutionary multiobjective and many-objective optimization. *Neurocomputing 116.*

Ranut, P., Janiga, G., Nobile, E. & Thevenin, D., 2014. Multi-objective shape optimization of a tube bundle in cross-flow. *International Journal of Heat and Mass Transfer 68.*

Reyes, J., 2014. *Reporte de modelo paramétrico,* s.l.: s.n.

Ruiz del Solar, J. & Held, C., 2012. *Inteligencia Computacional: Indroducción,* s.l.: Ramo EL4106 Inteligencia Computacional.

Rumelhart, D. & McClelland, J., 1986. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. *Cambridge: MIT Press.*

Silva, S., 2014. *GPLab: toolbox de programación genética para MATLAB.* [En línea] Available at: http://gplab.sourceforge.net/

Song, X. y otros, 2014. Surrogate-Based Analysis and Optimization for the Design of Heat Sinks Whith Jet Impingement.. *IEEE Transactions on Components Packaging and Manufacturing Technology 4.*

Tikhonov, A. & Arsenin, V., 1997. Solutions to ill-posed problems. *New York: Wiley.*

Ting, C.-K., 2005. On the Mean Convergence Time of Multi-parent Genetic Algorithms Without Selection. *Advances in Artificial Life,* p. 403–412.

Vetter, J. y otros, 2005. Ageing mechanisms in lithium-ion batteries. *Journal of Power Sources 147.*

Wendt, J., 2009. *Computational Fluid Dynamics: An Introduction.* s.l.:Springer.

Wilese, K. & Goodwin, S., s.f. *Convergence Characteristics of Keep-Best Reproduction,* s.l.: s.n.

www.TheEngineeringToolbox.com, s.f. *The Engineering Toolbox.* [En línea] Available at: http://www.engineeringtoolbox.com/air-properties-d_156.html [Último acceso: 3 Octubre 2014].

Yatchew, A., 1998. Nonparametric Regression Techniques in Economics. *Journal of Economic Literature,* pp. vol. 16: 669-721.

Zadeh, L., 1965. Fuzzy sets. *Information and Control 8,* p. 338–353.

Zavoianu, A. y otros, 2013. Hybridization of multi-objective evolutionary algorithms and artificial neural networks for optimizing the performance of electrical drives. *Engineering Applications of Artificial Intelligence 26.*

Zukauskas, A., 1972. Heat Transfer from Tubes in Crossflow. *Advances in Heat Transfer,* Volumen 8, pp. (93-160).

Zukauskas, A., 1987. Heat Transfer from Tubes in Crossflow. *Advances in Heat Transfer,* Volumen 18, pp. 87-159.

Apéndice A

Parámetros de programación genética para problemas introductorios

A.1 Polinomio sin constantes

- **Set de terminales:** {x}
- **Set de Funciones:** {+,-,*,^3,^2}
- **Función de fitness:** MSE
- **Parámetros de control:** 100 individuos y 10 generaciones
- **Condición de termino:** máximo de generaciones

- **Profundidad máxima de árboles de primera generación:** 6
- **Método de creación de individuos:** *Ramped Half-Half*
- **Profundidad máxima de árboles:** 17
- **Operadores genéticos:** crossover y mutación
- **Probabilidades de ocurrencia de operadores:** crossover(90%), mutación (10%)
- **Selección:** *lexicographic parsimony pressure*
- **Supervivencia:** keep-best

A.2 Polinomio con constantes

- **Set de terminales:** {x, número aleatorio}
- **Set de Funciones:** {+,-,*,^3,^2}
- **Función de fitness:** MSE
- **Parámetros de control:** 100 individuos y 10 generaciones
- **Condición de termino:** máximo de generaciones

- **Profundidad máxima de árboles de primera generación:** 6
- **Método de creación de individuos:** *Ramped Half-Half*
- **Profundidad máxima de árboles:** 17
- **Operadores genéticos:** crossover y mutación
- **Probabilidades de ocurrencia de operadores:** crossover(90%), mutación (10%)
- **Selección:** *lexicographic parsimony pressure*
- **Supervivencia:** keep-best

A.3 Modelo fenomenológico completo

- **Set de terminales:** {separación, corriente, flujo, números aleatorios}
- **Set de Funciones:** {+,-,*,/,^3,^2}
- **Función de fitness:** $MSE + \alpha\sqrt{nodos}$
- **Parámetros de control:** 2000 individuos
- **Condición de termino:** 30 generaciones

- **Profundidad máxima de árboles de primera generación:** 6
- **Método de creación de individuos:** *Ramped Half-Half*
- **Profundidad máxima de árboles:** 17
- **Operadores genéticos:** crossover y mutación
- **Probabilidades de ocurrencia de operadores:** crossover(90%), mutación (10%)
- **Selección:** *lexicographic parsimony pressure*
- **Supervivencia:** keep-best

Apéndice B

Parámetros de programación genética para expresiones del modelo fenomenológico

B.1 Factor de fricción

- **Set de terminales:** $\left\{rand, \frac{Vmf}{Vf_{ref}}, \frac{Df_{(i)}}{Df_{ref}}, S, Re\right\}$
  Donde rand es un número aleatorio entre -1 y 1, Vmf la velocidad media del fluido, Vf_{ref} una velocidad de referencia, S la separación entre celdas, Re el número de Reynolds, D_f la densidad del aire dentro de la batería y Df_{ref} una densidad de referencia.
- **Set de Funciones:** {+,-,*,/,^3,^2}
- **Función de fitness:** se corrió el algoritmo tres veces con tres funciones distintas
  - Error cuadrático medio (MSE)
  - MSE con regularización: se corrió 5 veces con 5 valores distintos para α: 0.1, 0.5, 0.75, 1 y 2.
  - MSE con ponderación : se corrió 5 veces con 5 valores distintos para *L*: 10, 20, 30, 40,y 50.
- **Parámetros de control:** 2000 individuos
- **Condición de termino:** inspección manual
- **Profundidad máxima de árboles de primera generación:** 6
- **Método de creación de individuos:** *Ramped Half-Half*
- **Profundidad máxima de árboles:** 17
- **Operadores genéticos:** crossover, mutación y Prune&Plant
- **Probabilidades de ocurrencia de operadores:** crossover(80%), mutación (10%), Prune&Plant(10%).
- **Selección:** *lexicographic parsimony pressure*
- **Supervivencia:** keep-best

B.2 Coeficiente de arrastre

- **Set de terminales:** $\{rand, \frac{A_{sup}}{A_{ctrl}}, Re, \frac{D_f}{Df_{ref}}\}$
  Donde rand es un número aleatorio entre -1 y 1, A_{sup} el área de la celda, A_{ctrl} el área entre dos celdas de la columna de fluido, Re el número de Reynolds, D_f la densidad del aire dentro de la batería y Df_{ref} una densidad de referencia.
- **Set de Funciones:** {+,-,*,/,^3,^2}

- **Función de fitness:** se corrió el algoritmo 4 veces, con 2 funciones distintas
    - MSE con regularización: se corrió 2 veces con $\alpha = 0.5$ (una vez con y una vez sin Prune&Plant).
    - MSE con ponderación : se corrió 2 veces con $L = 20$ (una vez con y una vez sin Prune&Plant).
- **Parámetros de control:** 2000 individuos
- **Condición de termino:** inspección manual
- **Profundidad máxima de árboles de primera generación:** 6
- **Método de creación de individuos:** *Ramped Half-Half*
- **Profundidad máxima de árboles:** 17
- **Operadores genéticos:** se usaron dos configuraciones distintas:
    - crossover, mutación y Prune&Plant
    - crossover y mutación
- **Probabilidades de ocurrencia de operadores:**
    - crossover(80%), mutación (10%), Prune&Plant(10%).
    - crossover(80%), mutación (20%),
- **Selección:** *lexicographic parsimony pressure*
- **Supervivencia:** keep-best

B.3 Número de Nusselt

- **Set de terminales:** $\{rand, Re\}$
  Donde rand es un número aleatorio entre -1 y 1 y Re es el número de Reynolds
- **Set de Funciones:** {+,-,*,/,^3,^2}
- **Función de fitness:** MSE con regularización se corrió 3 veces con distintos valores de α (0.3 ,0.4 y 0.5).
- **Parámetros de control:** 2000 individuos
- **Condición de termino:** inspección manual
- **Profundidad máxima de árboles de primera generación:** 6
- **Método de creación de individuos:** *Ramped Half-Half*
- **Profundidad máxima de árboles:** 17
    - **Operadores genéticos:** crossover, mutación y Prune&Plant
- **Probabilidades de ocurrencia de operadores:**
    - crossover(80%), mutación (10%), Prune&Plant(10%).
- **Selección:** *lexicographic parsimony pressure*
- **Supervivencia:** keep-best

Apéndice C

Promedio, varianza y nodos de los mejores individuos

C.1 Corrida de programación genética con MSE para el factor de fricción

Tabla C.1 Características de los mejores individuos de la corrida con MSE para el factor de fricción

	Mean	Var	Nodos
MSE	11,046032	1113,11975	62
	10,469685	930,39053	40
	10,553229	931,229202	25
	10,560024	931,356723	28
	10,744467	934,520381	25
	7,3606237	1349,96432	20
	10,425291	1469,46189	54
	13,806041	1649,19564	34
	-3,699231	1529,29122	54
	-3,699194	1529,28696	58

C.2 Corridas de programación genética con MSE con regularización para el factor de fricción

Tabla C.2 Características de los mejores individuos de la corrida con MSE con regularización para el factor de fricción

Alfa	Mean	Var	Nodos
0.1	55,97	41224,92	5
	58,26	41145,48	5
	60,46	41089,39	1
	60,73	41075,76	5
	0,59	815,89	25
	6,50	1039,89	27
	6,52	1039,23	27
	6,52	1039,23	32
	6,52	1039,23	33
	6,52	1039,23	33
0.5	60,84	41074,26	5
	-49,67	94480,39	21
	8,01	3039,98	20
	7,92	2986,05	22
	18,49	10087,30	19
	31,97	25172,22	6
	-116,42	1365563,36	18
	31,75	24738,21	8
	3,50	2012,22	17
	-24,17	65645,95	18
0.75	52,44	41493,64	1
	41,62	34146,36	8
	50,13	40526,99	3
	58,84	41259,90	3
	59,58	41116,17	3
	25,28	16396,31	22
	25,47	16810,18	18
	25,29	16395,79	22
	25,29	16395,79	22
	25,29	16395,79	26

Alfa	Mean	Var	Nodos
1	55,37	41326,65	2
	52,02	40888,90	3
	52,72	41476,89	2
	54,97	41322,32	3
	55,15	41312,98	3
	55,44	41298,11	3
	55,59	41290,91	3
	56,27	41280,11	2
	56,35	41275,68	2
	52,72	40847,89	3
2	49,70	40510,76	3
	50,09	40527,11	7
	53,69	41390,31	3
	41,81	36885,22	4
	47,65	41217,92	4
	53,39	41437,56	3
	55,41	41474,83	3
	57,25	41329,23	3
	57,74	41207,94	2
	51,44	41555,83	2

C.4 Corridas de programación genética con MSE con ponderación para el factor de fricción

Tabla C.3 Características de los mejores individuos de la corrida con MSE con ponderación para el factor de fricción

L	Mean	Var	Nodos
10	36,43	31753,71	6
	-7,32	2705,38	6
	32,29	27740,90	5
	37,81	33116,04	4
	38,25	33250,33	4
	37,32	32967,96	5
	36,31	32665,86	4
	36,24	32644,58	4
	39,62	33672,53	4
	35,72	32490,36	4
20	32,29	27740,90	5
	32,20	27564,80	8
	35,07	29262,66	8
	-17,62	3792,46	10
	-15,93	13011,32	16
	14,71	8860,64	11
	35,94	30562,50	12
	35,44	30579,25	12
	18,41	12061,78	13
	41,59	34290,80	10
30	41,79	34353,15	4
	48,99	40484,19	3
	47,90	41210,00	4
	45,99	40829,40	3
	51,63	41293,35	4
	53,05	41457,07	2
	57,29	40863,91	3
	52,79	41817,71	2
	37,64	40221,87	3
	41,89	42030,50	3

L	Mean	Var	Nodos
40	30,38	17774,17	19
	37,10	29801,10	15
	37,87	30788,79	12
	37,85	30736,99	15
	37,84	30735,88	16
	37,84	30735,88	17
	37,84	30735,88	16
	37,84	30735,88	18
	37,84	30735,88	18
	37,84	30735,88	18
50	48,28	41105,96	7
	48,28	41151,83	3
	49,51	40503,51	3
	51,63	41293,35	4
	52,50	40625,49	3
	53,05	41300,02	7
	54,45	41349,45	3
	54,27	41358,72	3
	52,21	41400,83	7
	52,70	40849,12	3

C.5 **Corridas de programación genética con MSE con ponderación para el coeficiente de arrastre**

Tabla C.4 Características de los mejores individuos de la corrida con MSE con ponderación para el coeficiente de arrastre

	P&P	Mean	Var	Nodos
MSE con ponderación	NO	1,39	70,76	7
		1,71	63,58	15
		4,03	95,67	9
		0,69	71,55	9
		2,87	95,39	7
		2,36	90,75	9
		-1,34	27,20	11
		2,46	91,31	12
		1,70	84,85	13
		-0,93	22,14	15
		-0,70	34,62	15
		-13,94	88,94	3
		-2,92	11,03	11
		-2,86	10,56	13
		-2,82	11,13	16
		-0,08	73,46	7
		4,58	151,85	13
		5,24	115,53	14
		5,59	124,39	11
		0,82	102,40	9

	P&P	Mean	Var	Nodos
MSE con ponderación	Si	1,39	8,44	13
		2,21	44,36	16
		1,80	41,30	16
		1,79	41,21	20
		1,51	44,09	11
		1,51	44,14	16
		5,12	124,25	7
		4,44	100,27	11
		4,46	111,20	5
		4,90	116,14	7
		5,71	121,52	17
		5,17	119,40	7
		4,06	102,61	20
		4,32	50,92	12
		2,95	112,63	5
		2,84	112,42	11
		6,97	187,99	3
		6,59	196,54	3
		5,93	199,97	3
		0,09	36,39	14

C.6 Corridas de programación genética con MSE con regularización para el coeficiente de arrastre

Tabla C.5 Características de los mejores individuos de la corrida con MSE con regularización para el coeficiente de arrastre

	P&P	Mean	Var	Nodos
MSE con regularización	NO	1,06	22,51	10
		0,77	31,05	39
		0,75	31,08	43
		0,75	31,07	43
		0,77	31,05	43
		0,77	31,05	45
		0,77	31,05	45
		0,77	31,05	43
		0,82	39,66	11
		1,97	57,98	20
		2,61	30,31	13
		1,14	23,51	39
		0,78	63,49	7
		-0,51	7,52	31
		1,28	23,81	37
		2,82	31,06	19
		2,92	32,41	19
		-0,11	26,22	26
		-0,60	17,66	28
		1,64	22,87	37

	P&P	Mean	Var	Nodos
MSE con regularización	NO	-0,88	26,29	24
		1,12	22,40	31
		3,56	49,40	11
		6,95	205,82	11
		-1,23	25,83	9
		-2,44	29,79	5
		-1,66	26,94	10
		-0,27	23,31	5
		-1,54	27,56	14
		0,32	20,54	19
		-0,95	25,27	10
		-0,65	23,72	9
		-2,45	29,75	7
		-1,38	25,39	11
		6,24	216,01	14
		-1,38	26,12	13
		-0,82	26,38	13
		-0,63	25,96	11
		-0,14	23,12	7
		-0,25	25,25	7

C.7 Corridas de programación genética con MSE con regularización para el número de Nusselt

Tabla C.7 Características de los mejores individuos de la corrida para el número de Nusselt

Alfa	Mean	Var	Nodos
0.3	1,82	10,68	3
	1,82	10,73	3
	1,82	10,73	3
	1,18	11,38	31
	1,33	11,31	31
	1,40	10,77	39
	1,59	11,86	28
	1,59	11,86	28
	1,59	11,86	28
	1,59	11,86	28
	1,59	11,86	28
	1,59	11,86	28
	1,56	11,99	30
	1,59	11,86	30
	1,59	11,86	30
	1,59	11,86	42
	1,59	11,86	42
	1,48	14,97	17
	1,48	14,97	17
	1,49	14,99	37

Alfa	Mean	Var	Nodos
0.4	0,83	7,53	3
	0,84	7,48	3
	0,80	7,66	3
	0,91	7,26	3
	0,96	7,12	3
	0,71	8,03	3
	0,69	8,12	3
	0,65	8,28	3
	1,03	6,97	3
	1,05	6,93	3
	0,61	8,50	3
	1,17	6,79	3
	1,24	6,79	3
	1,27	6,81	3
	1,27	6,81	3
	1,29	6,83	3
	1,02	8,04	5
	1,34	6,91	3
	0,34	10,11	3
	1,36	6,93	3

Alfa	Mean	Var	Nodos
0.5	0,82	7,57	3
	0,81	7,61	3
	0,80	7,65	3
	0,93	7,20	3
	1,06	6,91	3
	1,08	6,88	3
	1,20	6,79	3
	1,27	6,81	3
	1,27	6,81	3
	1,31	6,85	3
	1,33	6,88	3
	1,35	6,91	3
	1,36	6,94	3
	1,37	6,97	3
	1,13	7,18	5
	1,39	7,02	3
	1,41	7,06	3
	1,41	7,07	3
	1,42	7,10	3
	1,43	7,13	3

Apéndice D

Ajuste fino

D.1 Ajuste para la expresión seleccionada de factor de fricción

Tabla D.1 Valores de la media y varianza para los ajustes hechos a la expresión del factor de fricción

Constante	Valor	Media	Varianza
Expresión evolucionada		0,40793423	813,843176
Exponente Rem	-1%	-1,02329937	914,950679
	+1%	1,80589484	759,358225
Expresión completa	-1%	1,0122505	782,722498
	+1%	-0,19638204	853,725971
Primer término trinomio	-1%	0,6022941	816,339022
	+1%	0,21357436	811,417381
	+2%	0,01921449	809,061638
	+3%	-0,17514537	806,775946
	+2.1%	-0,00022149	808,829916
	+2-2%	-0,01965748	808,598895
	+2.3%	-0,03909347	808,368575
	+2.4%	-0,05852945	808,138955
	+2.5%	-0,07796544	807,910035
Segundo término trinomio	-1%	0,89036389	780,294111
	+1%	-0,07449542	856,100131
Tercer término trinomio	-1%	0,33546098	813,789428
	+1%	0,48040749	813,908523
	-2%	0,26298773	813,74728
	-3%	0,19051447	813,716732
	-4%	0,11804122	813,697784
	-5%	0,04556797	813,690435
	-6%	-0,02690528	813,694686
	-5.4%	0,01657867	813,690743
	-5.5%	0,00933134	813,69111
	-5.6%	0,00208402	813,691593
	-5.7%	-0,00516331	813,692192
	-5.8%	-0,01241063	813,692907
	-5.9%	-0,01965796	813,693738

D.2 Ajuste para la expresión seleccionada de coeficiente de arrastre

Tabla D.2 Valores de la media y varianza para los ajustes hechos a la expresión del coeficiente de arrastre

Constante	Valor	Media	Varianza
Expresión original		-0,84	25,01
Exponente	-1%	-0,80	24,93
	+1%	-0,88	25,09
	-2%	-0,76	24,86
	-3%	-0,71	24,79
	-4%	-0,67	24,71
	-5%	-0,63	24,65
	-6%	-0,59	24,58
	-7%	-0,55	24,51
	-8%	-0,51	24,45
	-9%	-0,47	24,38
	-10%	-0,43	24,32
	-20%	-0,03	23,79
	-21%	0,01	23,75
	-22%	0,05	23,71
	-20.1%	-0,02	23,79
	-20.2%	-0,02	23,78
	-20.3%	-0,02	23,78
	-20.4%	-0,01	23,77
	-20.5%	-0,01	23,77
	-20.6%	-0,02	23,79
	-20.7%	0,00	23,76
	-20.8%	0,00	23,76
Coeficiente 4.78	-1%	-0,52	24,84
	+1%	-1,15	25,30
	-2%	-0,21	24,78
	-3%	0,10	24,84
	-2.5%	-0,05	24,79
	-2.6%	-0,02	24,80
	-2.7%	0,01	24,81
	-2.8%	0,04	24,82
	-2.9%	0,07	24,82