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

AJUSTE Y ANÁLISIS DE MODELOS DE EMPAQUETAMIENTO DE BATERÍAS BASADOS EN ALGORITMOS EVOLUTIVOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

FRANCISCO FELIPE BORN DE TORO

PROFESOR GUÍA:
PABLO ESTÉVEZ VALENCIA

PROFESOR COGUÍA:
JORGE VERGARA QUEZADA

COMISIÓN:
WILLIAMS CALDERÓN MUÑOZ

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: FRANCISCO FELIPE BORN DE TORO
FECHA: 2020
PROF. GUÍA: PABLO ESTÉVEZ VALENCIA

AJUSTE Y ANÁLISIS DE MODELOS DE EMPAQUETAMIENTO DE BATERÍAS BASADOS EN ALGORITMOS EVOLUTIVOS

Las baterías de ión litio tienen un gran número de aplicaciones y se prevee que su demanda aumente significativamente en los próximos años. Éstas están compuestas por varias celdas distribuidas de manera de optimizar el espacio pero manteniendo una separación suficiente entre ellas para mantener una ventilación adecuada evitando que la temperatura se eleve sobre cierto umbral. El diseño de empaquetamiento se refiere al proceso por el cual se decide la distribución de las celdas. Actualmente se utilizan programas computacionales de multifísica que simulan el comportamiento de distribuciones en escenarios reales. Sin embargo, las simulaciones requieren bastante tiempo y no proveen información sobre el comportamiento de las variables involucradas en el sistema, por lo que es difícil realizar optimizaciones en los diseños.

Con anterioridad se formuló un modelo fenomenológico que permite evaluar una distribución de celdas de manera general en pocos segundos y predecir lo que ocurre al realizar modificaciones a las variables de diseño o entrada aportando de esta forma al proceso de diseño óptimo de empaquetamiento de baterías. Sin embargo, el modelo fenomenológico obtiene diferencias significativas con los programas computacionales de multifísica, por lo que en trabajos posteriores se han utilizado algoritmos evolutivos para ajustar el modelo y mejorar su desempeño al predecir las salidas. En esta memoria de título se ajusta el modelo fenomenológico utilizando el algoritmo de evolución gramatical y se estudia el sentido físico de las expresiones obtenidas.

Se utilizó el algoritmo de evolución gramatical debido a que permite acotar el espacio de búsqueda de soluciones a través de un conjunto de reglas que definen cómo se generan las expresiones en el proceso evolutivo. En particular, permite acotar a un espacio de soluciones de menor complejidad matemática en comparación a las soluciones encontradas usualmente con algoritmos evolutivos. Con el algoritmo de evolución gramatical se ajustó el modelo fenomenológico y se midió su desempeño comparando las salidas que entrega con las salidas teóricas. Las raíces de los errores cuadráticos medios entre las salidas obtenidas y las teóricas son de 1.93[m/s], 4.68[Pa] y 1.09[°C] para la velocidad, presión y temperatura respectivamente. Luego se definió un conjunto de curvas de las variables de salida en función de las variables de entrada que permiten validar el sentido físico de un modelo. Del trabajo se concluyó que el algoritmo de evolución gramatical permite obtener soluciones de baja complejidad matemática y con desempeño similar al mejor obtenido en trabajos anteriores en cuanto a la predicción de salidas.

Agradecimientos

Memoria de Título desarrollada en el marco de ANID / FONDAP / 15110019.

TABLA DE CONTENIDO

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

Actualmente existe una gran preocupación por el medio ambiente, por lo que se han tomado medidas globales para realizar cambios que disminuyan la contaminación. Uno de los grandes cambios es el uso de energía proveniente de fuentes renovables. Dentro de este contexto, las baterías de ión-litio han entrado al mercado con dos aplicaciones importantes: el uso de baterías en vehículos eléctricos y respaldo en sistemas eléctricos de potencia. Por estos motivos, se predice que la demanda de las baterías ión-litio aumentará en los próximos años y así mismo su importancia. A modo de ejemplo, en [1] se indica que el 99.3 % de los vehículos vendidos el año 2015 tenían un motor de combustión interna mientras que en el año 2030 se predice que el 25 % de todos los autos vendidos serán completamente eléctricos o híbridos, necesitando una gran cantidad de baterías ión-litio. Lo anterior motiva el interés por diseñar mejores baterías y en particular optimizar el diseño del empaquetamiento de éstas.

El empaquetamiento de baterías se refiere principalmente a la distribución de los componentes (celdas) de un banco de baterías, sujeto a ciertas restricciones, tal como puede ser la capacidad de entregar cierta cantidad de corriente sin que aumente de forma contraproducente la temperatura y mantener un tamaño de banco aceptable. El diseño del empaquetamiento es un proceso complejo debido a que su comportamiento general no es fácil de predecir. Por lo que los diseños se basan más en la metodología de prueba y error, donde se configuran de distintas maneras según: disposición de las celdas, tamaño, forma, separación entre ellas, etc. Realizar tantos experimentos en el laboratorio no es factible por lo que generalmente se utilizan *softwares* de simulación de fluido dinámica (CFD: *Computational Fluid Dynamics*). Sin embargo, este método tiene dos deficiencias: el programa actúa como una caja negra que no permite ver las interacciones que hay entre las diversas variables físicas involucradas en el proceso y es costoso en tiempo. De aquí surge la motivación de aportar en el diseño óptimo de empaquetamiento de baterías, en particular disminuir el tiempo y comprender el efecto de algunas de las variables que interactúan.

Con el objetivo anterior, en [2] se realizó un modelo fenomenológico (de aquí en adelante será utilizado como modelo base). En éste se asumió una disposición de las celdas en forma de grilla y se encontraron expresiones teóricas para la temperatura de las celdas, la velocidad promedio del fluido y la presión del fluido. Lo anterior en función de la corriente en las celdas, separación entre ellas, su diámetro, el flujo de entrada de aire y la temperatura del aire. El modelo fenomenológico consta de varias ecuaciones relacionadas a la dinámica de fluidos (en la Sección 2.2 se realiza un análisis detallado del modelo fenomenológico), las cuales a su vez dependen además de las entradas, de algunos parámetros del modelo. Entre los parámetros de interés se encuentra el número de Nusselt, el coeficiente de fricción y el coeficiente de arrastre, los cuales tienen una dependencia fuerte de la configuración que tienen las celdas y del tipo de flujo del fluido alrededor de éstas, ya sea un flujo laminar o turbulento. Con el objetivo de conocer el desempeño del modelo, se simularon distintas configuraciones de empaquetamiento con el CFD comercial ANSYS®. Para distintos valores de las variables de entradas se compararon las salidas que entrega el CFD ANSYS® con las salidas que entrega el modelo fenomenológico. Las salidas que entrega el CFD ANSYS® se utilizaron como *ground truth* durante todo el trabajo.

Debido a que el modelo fenomenológico [2] es una aproximación del proceso fluido dinámico, éste entrega diferencias significativas respecto al CFD, por lo que en trabajos posteriores a Reyes [2] han buscado ajustar este modelo para que tenga un mejor desempeño. Villa [3] seleccionó los 3 parámetros de interés mencionados anteriormente como posibles expresiones a optimizar, esto debido a que se utilizaron suposiciones sobre ellos en base a experimentos similares para obtener el modelo fenomenológico. Con el fin de encontrar mejores expresiones de los 3 parámetros, y considerando las limitaciones del software utilizado (MATLAB), Villa acotó el problema a 5 celdas y se realizó una optimización de los 3 parámetros en forma independiente utilizando el algoritmo evolutivo de programación genética (PG). Posteriormente, en [4] se traspasó el código al lenguaje de programación JAVA para sobrepasar las limitaciones de MATLAB. En [5] se utilizaron algoritmos de optimización multi-objetivo [6] para optimizar simultáneamente los 3 parámetros de interés, mejorando los resultados de Villa pero manteniendo el problema acotado a configuraciones de 5 celdas. Finalmente, en la memoria de Aguilar [7] se mantuvo la utilización del programación genética y optimización multi-objetivo, pero se estudiaron configuraciones de 5, 53, 102 y 151 celdas para encontrar un ajuste más general y robusto del modelo base. El detalle de la evolución de los trabajos anteriores se encuentra en la Sección 2.6.

Debido al funcionamiento heurístico de programación genética se obtuvieron variadas expresiones de los parámetros buscados que generan un modelo con mejor desempeño que el modelo base para predecir las salidas. Sin embargo, los trabajos anteriores tenían por objetivo ajustar los datos sin tomar en cuenta la complejidad e interpretabilidad de las expresiones. Es por esto que no todas las expresiones encontradas en los trabajos previos son de fácil comprensión. Es necesario considerar que las soluciones debieran representar fenómenos físicos, por lo que deberían ser capaces de predecir correctamente lo que ocurre al realizar variaciones en alguna de sus variables dependientes. Esto motiva el trabajo actual de encontrar expresiones interpretables físicamente y de baja complejidad.

En la presente memoria, se continúan los trabajos previos para ajustar el modelo fenomenológico. Para esto se propone encontrar nuevas expresiones para los parámetros de interés utilizando un algoritmo distinto a PG, llamado evolución gramatical. Se propone este algoritmo debido a su capacidad para guiar el espacio de búsqueda según criterios impuestos en la gramática. Finalmente, se analizan los efectos de las variables de entrada sobre los mejores modelos obtenidos en trabajos previos y sobre el modelo obtenido en esta memoria. Para realizar esto se generaron curvas modificando solamente una variable a la vez. Se propone para trabajos futuros generar las mismas curvas en ANSYS® para poder validar los resultados del modelo obtenido con los datos *ground truth*.

1.1. Objetivo general

Ajustar y analizar un modelo fenomenológico de empaquetamiento de baterías utilizando evolución gramatical.

1.2. Objetivos específicos

- Obtener nuevas expresiones para los 3 coeficientes de interés del modelo fenomenológico usando el algoritmo de evolución gramatical.
- Definir un conjunto de criterios básicos que debe cumplir un modelo para que satisfaga un grado de interpretación física.
- Validar los modelos en base al cumplimiento del conjunto de criterios básicos definido anteriormente.

1.3. Estructura de la memoria

El trabajo de título se organiza en 6 capítulos:

1. **Introducción:** Se presenta la motivación del tema de memoria, se aborda la problemática de manera general, destacando los trabajos previos y se listan los objetivos de la memoria.
2. **Marco Teórico:** Se revisan los antecedentes necesarios para el desarrollo del trabajo de título. Partiendo por la estructura de la batería a la que se quiere encontrar un modelo, seguido del software de multifísica CFD utilizado para generar las bases de datos que se utilizan en el trabajo para evaluar el desempeño de los modelos y el detalle del modelo fenomenológico que se quiere ajustar a la base de datos anterior. Luego se detallan de forma general los algoritmos evolutivos, en particular las técnicas de programación genética y evolución gramatical. Finalmente se describe la evolución que han seguido los trabajos previos hasta ahora.
3. **Metodología e Implementación:** En este capítulo se describe la metodología utilizada en el trabajo. Primero se describe la implementación del algoritmo de evolución gramatical utilizado para obtener nuevas expresiones para los coeficientes de interés. Luego se muestra cómo se analizó el efecto de las variables de entrada en el modelo para estudiar la interpretación física.

4. **Análisis de Resultados:** Se muestran las expresiones obtenidas para los coeficientes de interés. Luego se muestran las curvas de variables de salida (velocidad del fluido, presión del fluido y temperatura de celda) en función de las distintas entradas. Se analiza el desempeño de las expresiones obtenidas según su error frente al CFD. Finalmente, se analizan las curvas generadas para validar la interpretación física del modelo obtenido.
5. **Discusión de los Resultados:** Se describen los posibles casos que podrían haber ocasionado los resultados obtenidos.
6. **Conclusiones:** Se detallan las conclusiones del trabajo realizado y sus contribuciones. Luego se finaliza el informe con propuestas de trabajo futuro.

Capítulo 2

Marco Teórico

En este capítulo se introducen los antecedentes necesarios para el desarrollo del trabajo de título. Primero se describe el modelo de empaquetamiento de batería utilizado, segundo el software CFD ANSYS con el cuál se obtiene la base de datos para los trabajos y tercero, el modelo fenomenológico para el empaquetamiento de baterías que se busca ajustar. Luego se da una explicación general de lo que es un algoritmo evolutivo (AE), seguido de 2 tipos en particular: programación genética (PG) y evolución gramatical (EG). El primero fue utilizado en trabajos previos, mientras que el último es el algoritmo utilizado en este trabajo.

2.1. Modelo de batería y *software* CFD ANSYS

Ya que un *pack* de batería puede tener distribuciones de celdas diversas, se debe elegir una configuración en especial, esta sección sirve para entender el tipo estudiado y cómo fue generada la base de datos para evaluar el desempeño de los modelos. El problema fue acotado al estudio de un empaquetamiento de baterías de ión-litio escalonado, donde se tienen celdas cilíndricas distribuidas como un arreglo intercalado. En la Figura 2.1 se muestra a modo de ejemplo un *pack* de baterías que se encuentra en el mercado con columnas intercaladas de 4 y 5 celdas (distribución 4-5-4). Dado el tipo de celda a utilizar, la configuración de la batería queda completamente definida indicando el número de celdas, su diámetro y su separación. Cabe destacar que para esta memoria de título se utilizó sólo una configuración que consiste en 5 celdas con distribución 2-1-2.

Figura 2.1: Ejemplo de un *pack* de baterías real con distribución 4-5-4.[1]

Para poder evaluar una configuración durante el proceso de diseño de empaquetamiento, se suelen utilizar modelos computacionales basados en Dinámica de Fluidos Computacional (CFD por sus siglas en inglés). Estos modelos obtienen resultados precisos si se contrasta con la realidad y tienen asociado un costo menor respecto a pruebas en laboratorio. En trabajos previos a esta memoria de título se utilizó el programa CFD ANSYS® y se modelaron distintas distribuciones variando el número de celdas, su espaciado y su diámetro. Una vez que se modela una configuración el *software* permite aplicar, entre otras cosas, distintos valores para la corriente de la celda, el flujo de aire que entra y la temperatura del aire; luego permite realizar mediciones a esa configuración. En particular se realizaron mediciones de la velocidad del fluido, su presión y la temperatura de las celdas. El programa entrega valores de las magnitudes en todo el espacio de la batería, tal como se puede observar en la Figura 2.2 donde se muestra la variación de la temperatura. Para efectos de comparación con el modelo fenomenológico se tienen los lugares de medición mostrados en la Figura 2.3: la temperatura se mide en las columnas que tienen una única celda central (mostradas en la figura con color azul) como el promedio de su magnitud en toda la celda; mientras que la velocidad del fluido y su presión son obtenidas como el promedio a lo largo de la línea amarilla que une las dos celdas centrales de las demás columnas. Con este método se generó una base de datos de las variables de salida *versus* las variables de entrada. En la Figura 2.4 se muestra un diagrama de la generación de la base de datos.

[1]Imagen obtenida de `https://es.aliexpress.com/i/32929547202.html` el 9 de Agosto del 2020.

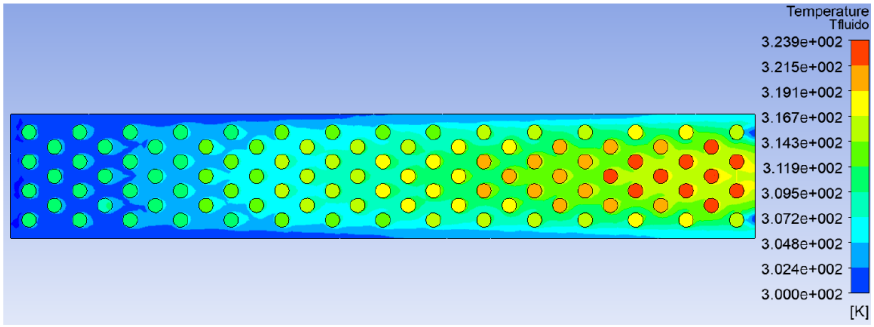


Figura 2.2: Ejemplo de resultado de simulación en ANSYS para la variable temperatura y una configuración con 102 celdas con distribución 4-3-4. Extraída de [7].

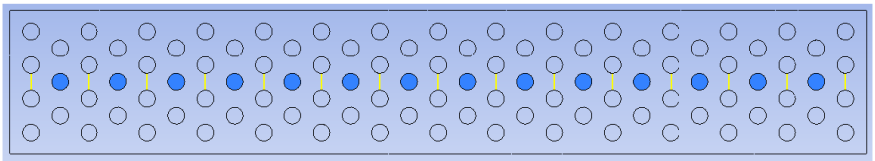


Figura 2.3: Lugares de medición de salidas en ANSYS para configuración con 102 celdas: en amarillo se muestra donde se mide la velocidad y presión del fluido, mientras que en azul se muestra donde se mide la temperatura de celda. Extraída de [7].

En el rectángulo rojo de la Figura 2.4 se indican las variables de entrada que se modificaron:

- **Separación entre las celdas** (S)**:** puede tomar valores continuos entre 0.3 y 1.5 [% diámetro].
- **Diámetro de las celdas** (D)**:** se utilizaron valores discretos de 18, 22, 26 y 32 [mm].
- **Corriente que circula por las celdas** (I)**:** puede tomar valores continuos entre 0 y 15 [A].
- **Flujo de aire a la entrada del pack** (F_{in})**:** puede tomar valores continuos entre 1 y 200 [CFM[2]].
- **Temperatura del aire de entrada** (T_{in})**:** puede tomar valores continuos entre 10 y 30 [°C].

Se realizaron diversas combinaciones de las variables de entrada que luego se simularon en ANSYS. El rectángulo azul de la Figura 2.4 representa las diversas distribuciones con las que se simularon las combinaciones de variables de entrada: de 5 celdas con distribución 2-1-2 y de 53, 102 y 151 celdas con distribución 4-3-4. Finalmente, en el rectángulo verde de la Figura 2.4 se indican las variables de salida: velocidad del fluido, presión del fluido y temperatura de celda, las cuales se obtienen simulando cada una de las combinaciones de entrada. Cabe destacar que según el número de columnas que tenga la configuración, es el número de velocidades, presiones y temperaturas que se obtienen de los lugares de medición. Tanto en trabajos previos como en éste se

[2]CFM son las siglas de la traducción al inglés de la unidad de medida “pies cúbicos por minuto” (*Cubic Feet per Minute*).

realizaron ajustes del modelo fenomenológico manteniendo la distribución de las celdas constante y por ello se obtienen distintos modelos para empaquetamientos de 5, 53, 102 y 151 celdas. En la Figura 2.5 se representa la división de la base de datos anterior según el modelo correspondiente.

Si bien se mencionaron 5 variables de entrada, éstas son solamente las que se hicieron variar al generar la base de datos. Ya que para realizar la simulación se deben especificar más datos, tales como: la resistencia de las celdas, distancia entre las celdas y la pared, tamaño del corte de estudio para realizar la simulación, tamaño de la grilla, etc. Estos últimos parámetros son fijos para todos los modelos.

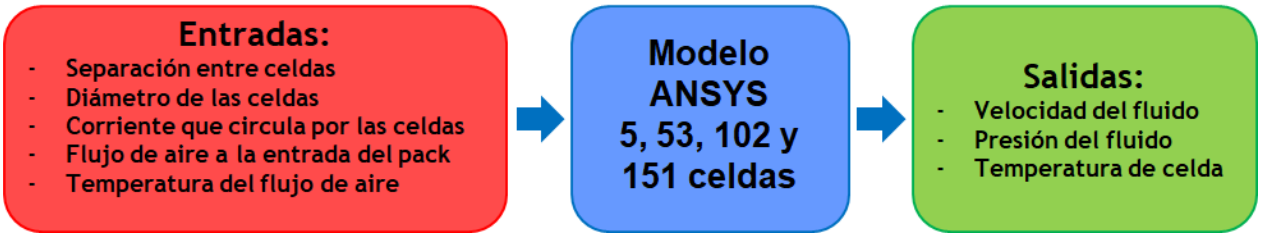


Figura 2.4: Diagrama que muestra de manera general en que consisten las bases de datos utilizadas para ajustar el modelo fenomenológico. Son generadas utilizando el *software* comercial CFD ANSYS. Variando el número de celdas se pueden contar 4 bases de datos distintas como se muestra en la Figura 2.5. En particular, en este trabajo se utilizó la base de datos para un modelo de 5 celdas.

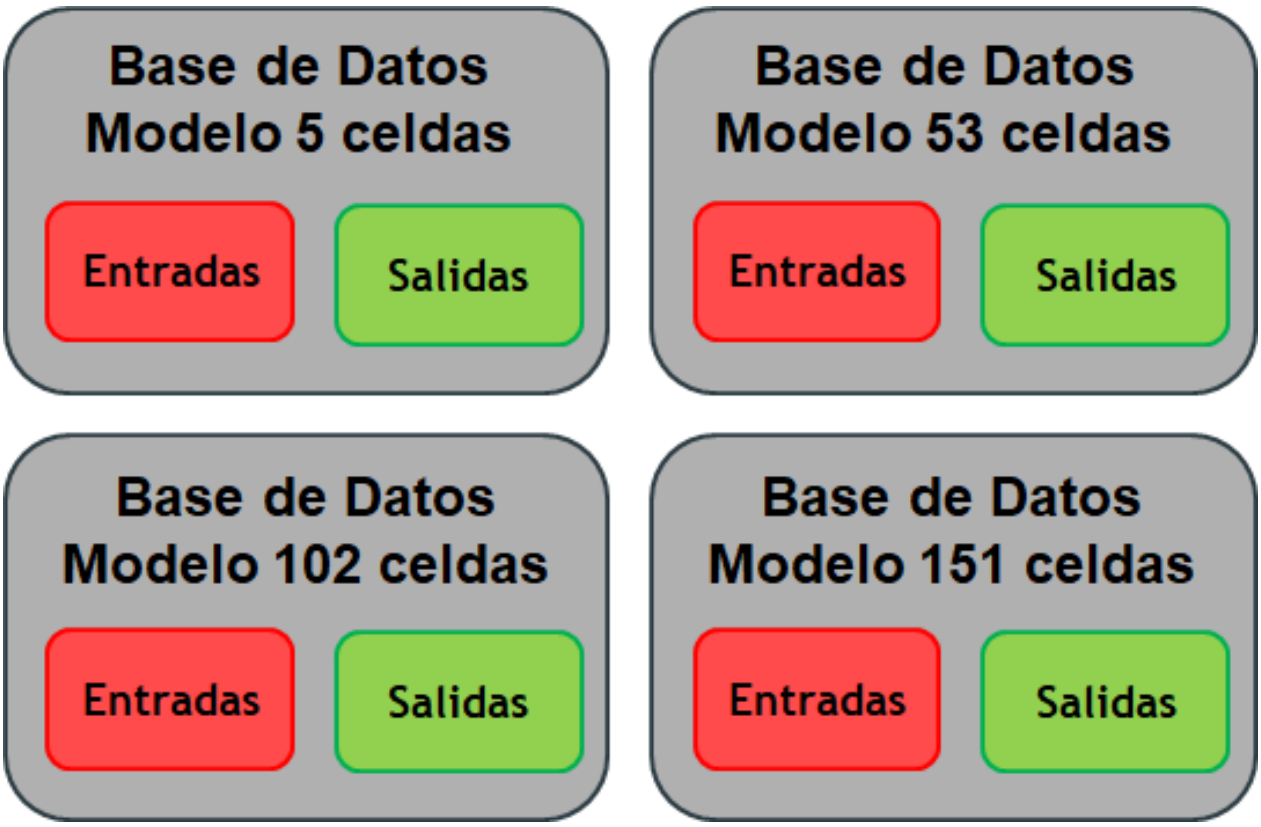


Figura 2.5: Diagrama que muestra en que consiste cada una de las 4 bases de datos generadas en trabajos previos. En este trabajo se utilizó la Base de Datos Modelo 5 celdas.

2.2. Modelo Fenomenológico

En esta sección se describe el modelo fenomenológico realizado por Jorge Reyes [2], que asume la estructura de un *pack* de baterías donde se tienen celdas cilíndricas distribuidas como un arreglo intercalado. Busca encontrar salidas similares a las obtenidas con el software ANSYS. Debido a que el modelo fenomenológico utiliza las ecuaciones diferenciales de manera explícita, se puede observar el comportamiento y la relación que hay entre las variables que interactúan. También permite obtener resultados más rápidos si se compara con realizar simulaciones con ANSYS que se basa en resolver ecuaciones diferenciales parciales. Cabe destacar que el objetivo es obtener una aproximación suficientemente buena de ANSYS que permita tener una idea general de cómo se comportaría el *pack* de baterías ante distintas configuraciones y utilizar el software únicamente para ajustes finos del diseño y su verificación.

Para realizar el modelo, se dividió la distribución de celdas en bloques y se estudió la interacción entre ellos. En la Figura 2.6 se observan estos bloques unidos (volúmenes de control) compuestos por 1 celda central y las cuatro esquinas de celdas contiguas. El modelo busca predecir las siguientes variables: temperatura de la celda central T_c, velocidad media del fluido V_f y presión media del fluido P_f. Estas 3 variables son funciones de: espaciado entre las celdas S, flujo del aire de entrada F_{in}, corriente que pasa por las celdas I, temperatura del flujo de entrada T_{in}, diámetro de las celdas D y número de la columna. Las variables de posición de las celdas se observan de mejor manera en la Figura 2.7, mientras que en la Figura 2.3 se puede ver donde se mide cada una de las variables de salida.

Para encontrar las variables de salida del modelo en la fila central de la batería como se muestra en la Figura 2.3, se utilizan ecuaciones que relacionan los estados de celdas contiguas. En el reporte de Reyes [2] se describen las ecuaciones principales: la ecuación para la caída de presión en intercambiadores de calor, la ecuación de gas ideal, la conservación del momento, la conservación de la energía y las ecuaciones que rigen la transferencia de calor que se describen más adelante. Como son ecuaciones de diferencias que relacionan el estado contiguo, se necesitan condiciones de borde, esto se muestra de mejor manera en el ejemplo de la Figura 2.8 donde se tiene una distribución 3-2-3 de 18 celdas. En la primera columna del ejemplo se tiene de condición de borde la velocidad V_{in} y en la última columna se tiene la presión P_{atm}, ambas señaladas en letras negras. Para entender las condiciones de borde, hay que tener en mente que el ventilador del *pack* se encuentra en el lado izquierdo y define la velocidad de entrada del fluido; mientras que el borde opuesto es donde termina el *pack* y se tiene la presión atmosférica. Además de las condiciones de borde, se muestran las columnas donde se miden las variables de salida del modelo: se tienen 3 medidas para la velocidad del fluido, 3 medidas para la presión del fluido y 3 medidas para la temperatura de celda; las cuales se muestran en rojo como V_{i}, P_{i} y T_{i} con $\text{i} = 1, 2, 3$ respectivamente.

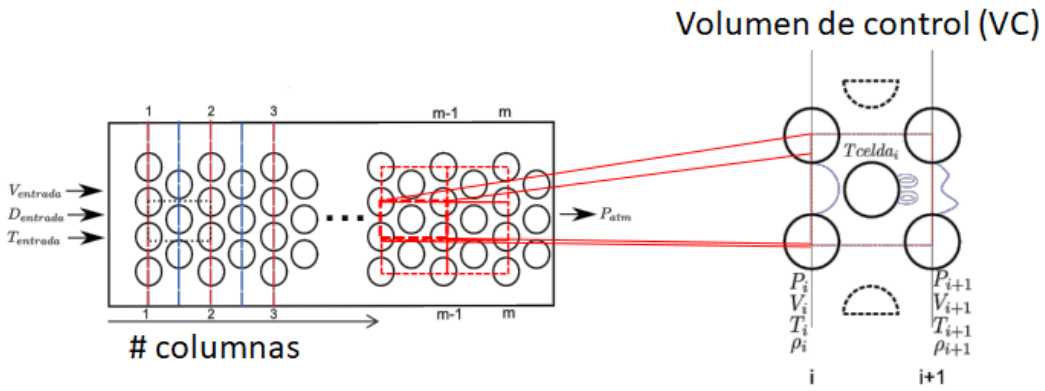


Figura 2.6: Estructura del empaquetamiento de una batería usada para obtener el modelo fenomenológico. Extraída de [8].

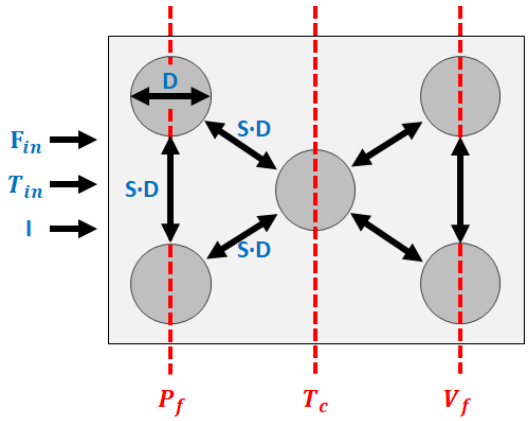


Figura 2.7: Esquema de las variables de entrada al modelo (en azul) y las variables de salida (en rojo). S = espaciado entre las celdas, F_{in} = flujo del aire de entrada, I = corriente que pasa por las celdas, T_{in} = temperatura del flujo de entrada, D = diámetro de las celdas, T_c = temperatura de la celda central, V_f = velocidad media del fluido y P_f = presión media del fluido.

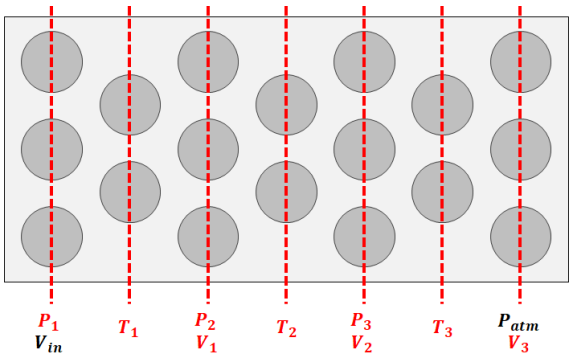


Figura 2.8: Representación gráfica donde se muestra en rojo los lugares de medición para cada variable de salida en un *pack* de batería con distribución 3-2-3 de 18 celdas. Y en letras negras se señalan las condiciones de borde para la velocidad y presión.

A continuación se señalan las ecuaciones principales utilizadas en el modelo fenomenológico.

Caída de presión

La caída de presión se modela en función del ordenamiento de las celdas, la forma de las celdas y la velocidad del fluido:

$$\Delta P = N_l f \chi \rho \frac{V^2}{2}, \tag{2.1}$$

donde ΔP es la caída de presión, N_l es la cantidad de columnas a la que se enfrenta el fluido, f es el factor de fricción del *pack*, χ es el factor de forma de la configuración, ρ es la densidad del fluido y V es la velocidad del fluido. El factor de fricción y el factor de forma se utilizan en conjunto como el **coeficiente de fricción**, el cual es uno de los parámetros de interés.

Gases ideales

La ecuación del gas ideal es:

$$P = \rho R T, \tag{2.2}$$

donde P es la presión del fluido, ρ es su densidad, T es su temperatura y R es la constante de gas ideal.

Conservación del momentum

La ecuación de conservación del momentum es:

$$P_{\mathrm{i}} A_{\mathrm{i}} + \dot{m}_{\mathrm{i}} V_{\mathrm{i}}^2 = P_{\mathrm{i+1}} A_{\mathrm{i+1}} + \dot{m}_{\mathrm{i+1}} V_{\mathrm{i+1}}^2 + F_{\text{ext}}, \tag{2.3}$$

donde P_{i} es la presión del fluido en la columna i, A_{i} es el área por donde pasa el fluido en la columna i, $\dot{m}_{\mathrm{i}}$ es el flujo másico que pasa por la columna i, V_{i} es la velocidad del fluido en la columna i y F_{ext} son las fuerzas externas que afectan el fluido (ver Figura 2.6). En este caso F_{ext} es la fuerza de arrastre que ejerce el cuerpo al fluido y se puede describir como:

$$F_{\text{arrastre}} = \frac{1}{2} A \rho V^2 C_{\text{drag}}, \tag{2.4}$$

donde A es el área proyectada del cuerpo y C_{drag} es el **coeficiente de arrastre** del cuerpo que es otro de los parámetros de interés.

Conservación de la energía

La ecuación de la conservación de la energía es:

$$\dot{m}(\Delta h + \frac{\Delta V^2}{2} + g \Delta z) = Q_{\text{fluido}} + W_{\text{eje}}, \tag{2.5}$$

donde Δh es la diferencia de entalpía, g es la aceleración de gravedad, Δz es la diferencia de altura, Q_{fluido} es la energía transferida hacia el fluido y W_{eje} es el trabajo ejercido hacia el fluido.

También se tiene que $\Delta h = c_p \Delta T$, donde c_p es el calor específico del fluido. De esta manera se puede relacionar la cantidad de energía transferida del fluido con un cambio de energía potencial, velocidad y temperatura.

Transferencia de calor

En este caso, los mecanismos por donde fluye el calor son la conducción a través de la celda y la convección entre la superficie del sólido y el fluido refrigerante. Las ecuaciones correspondientes son:

$$\dot{q} = -k_{\text{celda}} \Delta T \tag{2.6}$$

$$\dot{q} = h_{\text{fluido}} \Delta T, \tag{2.7}$$

donde $\dot{q}$ es el calor transferido por unidad de área, k_{celda} es el coeficiente conductivo de la celda y h_{fluido} es el coeficiente convectivo del fluido.

El coeficiente convectivo h se relaciona con el **número de Nusselt** del fluido, el cuál es el tercer parámetro de interés,

$$Nusselt = \frac{h_{\text{fluido}}}{k_{\text{fluido}}} L, \tag{2.8}$$

donde k_{fluido} es el coeficiente conductivo del fluido y L es un largo característico del objeto por donde circula el fluido; que en este caso Reyes utilizó el diámetro de la celda. El número de Nusselt se puede relacionar en función del número *Reynolds*

$$\textit{Reynolds} = \frac{VL}{\nu}, \tag{2.9}$$

donde V es la velocidad del fluido, ν es la viscosidad cinemática del fluido y el número *Prandtl*

$$\textit{Prandtl} = \frac{\nu}{\alpha}, \tag{2.10}$$

donde α es la difusividad térmica del fluido.

La correlación del número de Nusselt con el número de Reynolds y el número de Prandtl se conoce como:

$$\textit{Nu} = C \ Re^m \ Pr^n, \tag{2.11}$$

donde C, m y n son constantes que dependen del tipo de configuración y rango de aplicación del modelo.

Con las ecuaciones anteriores se realiza el algoritmo mostrado en la Figura 2.9. Es un proceso iterativo para las m columnas que obtiene las variables de salida T_c, V_f y P_f. Teniendo esto en

cuenta, el objetivo de la memoria es ajustar el modelo paramétrico modificando la expresión para el número de Nusselt, el coeficiente de arrastre y el coeficiente de fricción para que al utilizar el esquema de la Figura 2.9 se puedan obtener resultados cercanos a los que entrega el CFD ANSYS, y al mismo tiempo, que las soluciones tengan sentido físico.

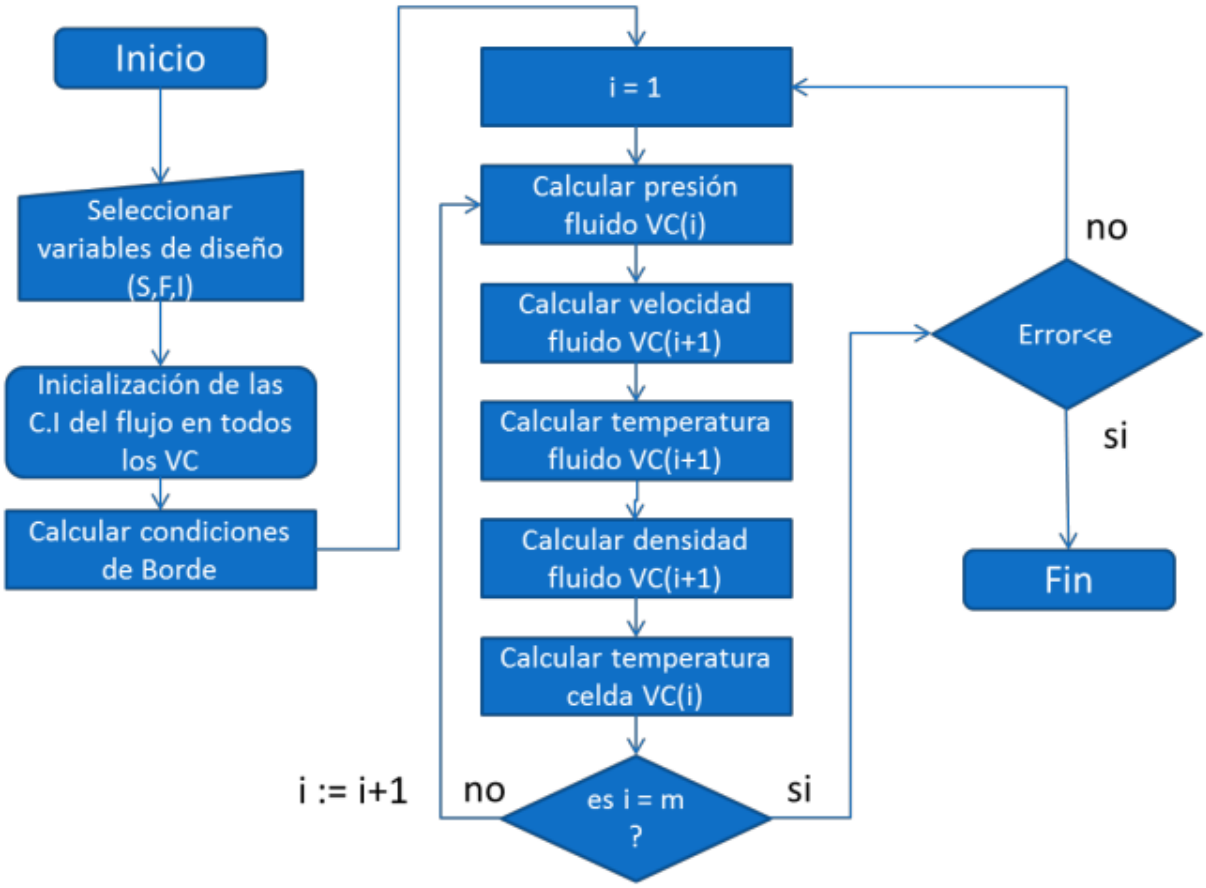


Figura 2.9: Esquema del algoritmo del modelo paramétrico. Extraído de [2].

En la Figura 2.9 se tiene una vista general del algoritmo del modelo paramétrico. Primero se seleccionan las variables que definen el diseño, se inicializan las condiciones iniciales del flujo en los volúmenes de control y se calculan las condiciones de borde. Teniendo los valores de las variables en la columna i, se puede obtener los valores de la siguiente columna $\mathrm{i}+1$ o la anterior $\mathrm{i}-1$. De esta manera con las condiciones de borde y condiciones iniciales que se muestran en la Figura 2.8 se puede llegar a tener el valor de las variables de interés para todas las m columnas del modelo. Luego se debe realizar nuevamente el algoritmo desde $\mathrm{i}=1$, pero esta vez las condiciones iniciales son los valores obtenidos en la iteración anterior. Este procedimiento se debe iterar hasta que la diferencia de los valores entre dos iteraciones seguidas sea menor a un error e, con ello se finaliza el algoritmo y se retornan los valores de presión, temperatura y velocidad para cada una de las columnas. También puede ocurrir que luego de un número determinado de iteraciones, los valores de las salidas en las columnas no converjan, en ese caso se detiene el algoritmo y se retorna error.

Hay que destacar que durante el informe se llama **modelo de Reyes** al modelo fenomenológico con las expresiones teóricas para el coeficiente de arrastre, coeficiente de fricción y número de Nusselt diseñado en [2]. De la misma manera, se llama **modelo de Villa** al modelo fenomenológico utilizando las expresiones de los parámetros encontradas por Villa en [3], lo mismo ocurre para el **modelo de García-Tapia** [5], **modelo de Aguilar** [7] y **modelo de Born** que se desarrolla en esta memoria de título. Esto se tiene debido a que cada modelo tiene su propio comportamiento según las expresiones encontradas.

2.3. Algoritmos Evolutivos (AE)

Esta sección está basada en [9] y explica de manera general cómo funcionan los algoritmos evolutivos para encontrar las soluciones a un problema, más adelante se explica en detalle dos tipos en particular: programación genética que se utilizó en trabajos previos y evolución gramatical que se utiliza en este trabajo.

Los algoritmos evolutivos están inspirados en la evolución natural de Darwin, basándose en mecanismos tales como: selección, reproducción, mutación y recombinación. Se utilizan para explorar soluciones de problemas, que por lo general son difíciles de abordar analíticamente. En la Figura 2.10 se muestra un diagrama de cómo funcionan los algoritmos evolutivos.

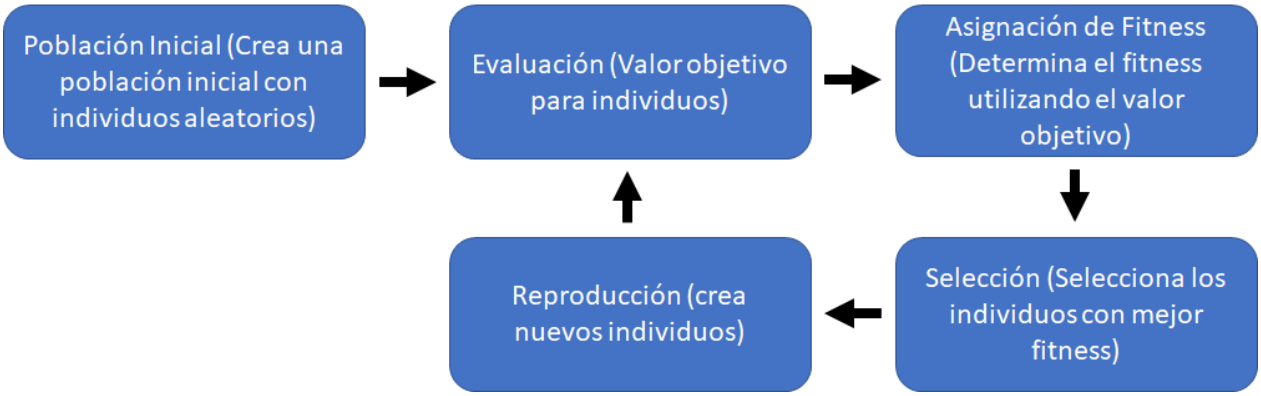


Figura 2.10: Vista general del funcionamiento de un algoritmo evolutivo. Traducción de la imagen extraída de [9].

En la Figura 2.10 se observa que de manera general se debe crear una población inicial de individuos que representan una solución al problema. Los individuos son evaluados según una métrica de desempeño que indica que tan cercanos son a la solución deseada y según el resultado de la evaluación se realizan operadores para generar una nueva generación de individuos que se acerquen más a la solución. La evaluación de la generación y creación de una nueva, es un proceso iterativo hasta que se llegue a un criterio de término. Se espera que cada generación se encuentre más adaptada al problema que su predecesora y se desempeñe de mejor manera.

A modo de ejemplo se presenta el siguiente problema de regresión, el cual tiene semejanzas con el que se soluciona en la memoria. En la Figura 2.11 se muestran 6 puntos de la base de datos utilizada en este problema, la cuál se generó con la fórmula $Y(x) = x^2$. Luego, a partir únicamente de los puntos de la base de datos y utilizando un algoritmo evolutivo para ajustar esos puntos, se encuentra una solución $y(x)$. Debido a que se conoce la fórmula utilizada para generar la base datos, se espera encontrar $y(x) = Y(x)$.

x	1	2	3	4	5	6
$Y(x)$	1	4	9	16	25	36

Figura 2.11: Relación de ejemplo entre x e Y. Obtenidos con $Y(x) = x^2$.

2.3.1. Etapas de un algoritmo evolutivo

Según [10], para resolver un problema con AE se deben seguir ciertos procedimientos. Los cuáles se dividen en representación, función de evaluación, población, mecanismo de selección de padres, operadores de variación y mecanismo de selección de sobrevivientes. En cada uno se tienen diferentes formas de realizarlo, en este caso para ejemplificar se utiliza el problema de regresión de la Sección 2.3 y el algoritmo evolutivo llamado **expresión de genes** [11] que es detallado junto a cada paso.

- **Representación (definición de individuos):** se define una relación entre soluciones del problema a abordar en el "mundo real" con codificaciones que el computador pueda manipular. Los objetos que corresponden a una posible solución al problema original se denominan **fenotipos**, mientras que su codificación para el algoritmo se denomina **genotipo**. La representación consiste en el mapeo del **espacio de fenotipos** al **espacio de genotipos**, los algoritmos evolutivos buscan dentro del último espacio al individuo que representa la mejor solución en el primero.

  Para definir los individuos en expresión de genes se debe elegir un conjunto de terminales y un conjunto de operadores que puedan relacionarse para generar una solución al problema. Como las soluciones del problema de regresión son fórmulas $y(x)$ en función de x, se utilizó:
  - **Conjunto de terminales:** [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, x], que corresponde a los dígitos y a la variable dependiente x.
  - **Conjunto de operadores:** [+, -, /, $\times$], que corresponde a los operadores básicos de suma, resta, división y multiplicación respectivamente.

El genotipo es la unión de dos secuencias de elementos: la cabeza y la cola. La cabeza del genotipo es una secuencia de largo h que está compuesta por elementos del conjunto de terminales y del conjunto de operadores. La cola del genotipo es de largo t y está compuesta por elementos del conjunto de terminales. Una vez definido el largo h, se tiene que $t = h(n-1)+1$, donde n es el número de argumentos del operador con mayor número de argumentos. Para el problema de regresión del ejemplo se decidió utilizar $h = 3$ y como $n = 2$ en este caso, se tiene que $t = 4$. Esto implica que el genotipo utilizado es de largo 7, tal cómo se muestra en el rectángulo azul de la Figura 2.12.

El fenotipo en expresión de genes corresponde a una fórmula matemática y es una posible solución al problema. En la Figura 2.12 se muestra un ejemplo de decodificación del genotipo (rectángulo azul) al fenotipo (rectángulo rojo) a través de un árbol de expresiones. Para la decodificación se genera el árbol de expresiones utilizando los elementos del genotipo uno a la vez y de izquierda a derecha. Según el ejemplo de la Figura 2.12, el primer elemento del genotipo (+) corresponde a la raíz del árbol y como pertenece al conjunto de operadores se abren tantas ramas con nodos sin elementos como argumentos tenga ese operador (en este caso se generan 2 nodos vacíos). Los nodos vacíos se van completando con los siguientes elementos del genotipo (2 y /). Como "/" es un operador con 2 argumentos, se abren dos ramas más con dos nodos vacíos. De la misma forma, se completan esos nodos vacíos con los siguientes elementos del genotipo (4 y 3). Este proceso se realiza de manera iterativa hasta que no queden nodos vacíos. Se destaca que este proceso puede terminar antes de utilizar todos los elementos del genotipo (en este caso no se utilizaron x y 1).

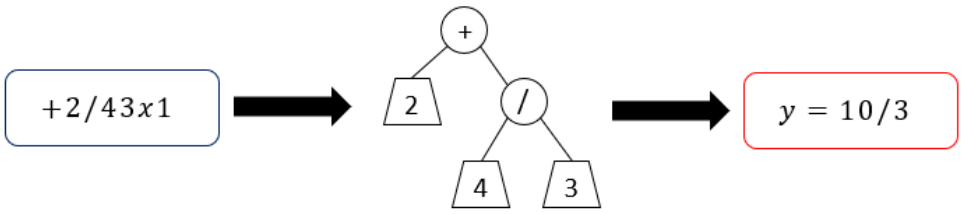


Figura 2.12: Se muestra el proceso de decodificación de genotipo (rectángulo azul) al fenotipo (rectángulo rojo), pasando a través de un árbol de expresión.

- **Función de evaluación:** cada individuo se evalúa a través de una función de aptitud o *fitness*, para obtener una medida de que tan buena es la solución que representa. Durante el trabajo se suele utilizar la palabra en inglés *fitness* para que no haya duda. Generalmente, se debe decodificar el genotipo de un individuo para luego aplicar la función de *fitness* a su fenotipo.

  Para este ejemplo se definió la función de *fitness* mostrada en la Figura 2.13 (rectángulo negro), donde $y(x)$ es la función a la cuál se le está calculando el *fitness*, $Y(x)$ es el valor de la función teórica con la que se generaron los datos y la sumatoria se calcula considerando todos los valores de x de la base de datos. Esta función va entre 0 y 1, dependiendo de si el error de aproximación de los datos es grande o pequeño, respectivamente. Junto a la definición se muestran 2 ejemplos de soluciones enmarcadas en rectángulos rojos: $y(x) = 10/3$ e $y(x) = 4x$ a las cuales se les calcula su *fitness* utilizando la función definida. En la

Figura 2.14 se muestran: en azul los valores de x e $Y(x)$ de la base de datos; y en verde los valores de $y(x)$ para las soluciones de ejemplo.

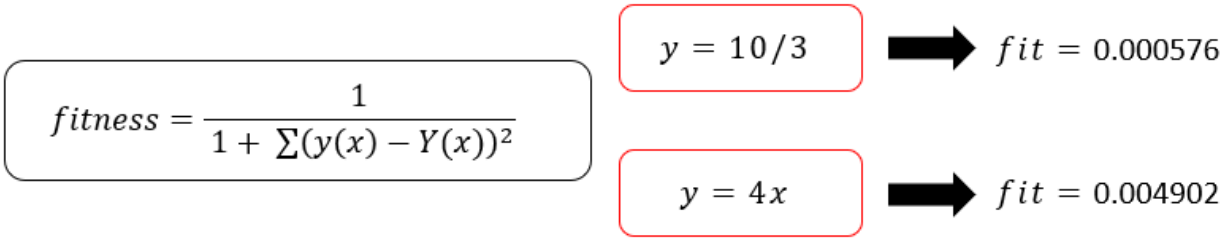


Figura 2.13: Definición de la función de *fitness* (rectángulo negro) y su valor para dos soluciones enmarcadas en rectángulos rojos: $y = 10/3$ e $y = 4x$.

x	1	2	3	4	5	6
$Y(x)$	1	4	9	16	25	36
$y(x) = 10/3$	10/3	10/3	10/3	10/3	10/3	10/3
$y(x) = 4x$	4	8	12	16	20	24

Figura 2.14: En azul se muestran los datos del problema original y en verde los resultados que entregan dos soluciones ($y = 10/3$ e $y = 4x$) al reemplazar el valor respectivo de x.

- **Población:** es un conjunto de genotipos, que pueden estar repetidos. El número de elementos corresponde al tamaño de la población, que generalmente se mantiene constante durante el algoritmo. Mientras que la diversidad corresponde al número de genotipos diferentes que se encuentran. En la Figura 2.15 se muestra una población compuesta por 3 individuos enmarcados en rectángulos azules.

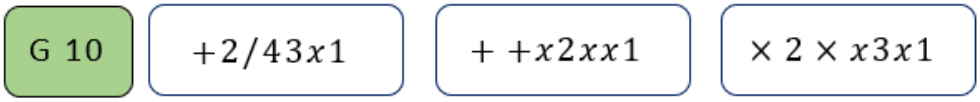


Figura 2.15: Ejemplo de población de tamaño 3. Los genotipos de los individuos se encuentran enmarcados en un rectángulo azul. En verde se indica que la población corresponde a la generación número 10 de individuos que evolucionaron durante el ejemplo de regresión.

- **Mecanismo de selección de padres:** se seleccionan los individuos de la población para que tengan descendencia según el valor de su función de evaluación. Según este criterio los más aptos tienen mayor probabilidad de ser elegidos, sin embargo hay que destacar que incluso los menos aptos tienen una probabilidad de generar descendencia. Esto último se hace para que la búsqueda no quede estancada en óptimos locales. Los más conocidos son la selección por torneo y por ruleta, ver Figura 2.16.

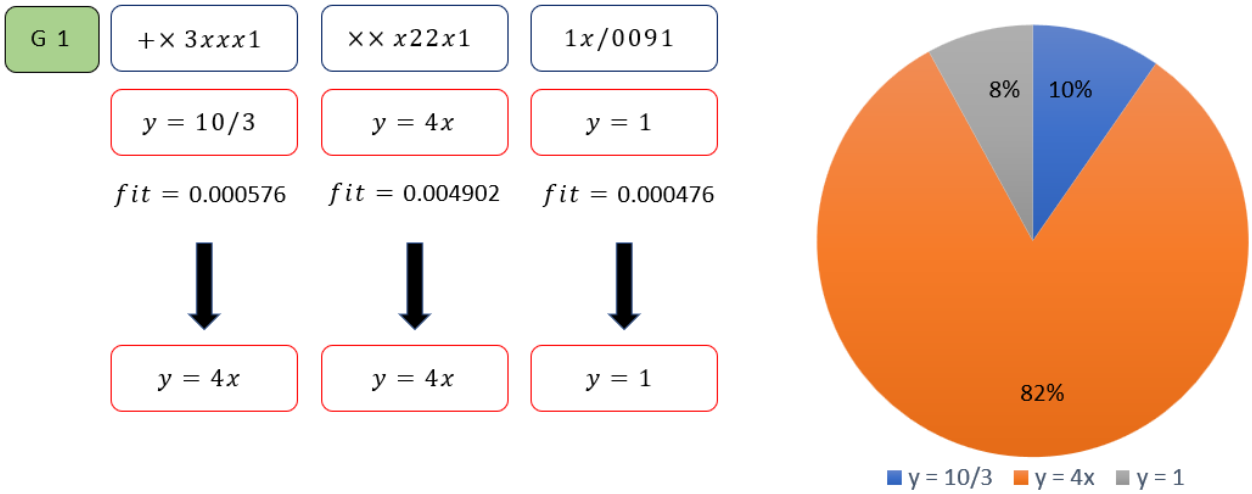


Figura 2.16: Selección por Ruleta de padres. Se divide la ruleta en partes de tamaño proporcional a los valores de la función de *fitness* de los padres, en este caso se divide en 3 partes debido que existen únicamente 3 individuos (mostrados a la izquierda: $y = 10/3$, $y = 4x$ e $y = 1$). Luego se elige un punto en la ruleta de forma aleatoria y se selecciona como padre al individuo asociado a esa división de la ruleta. Este proceso se itera hasta seleccionar la totalidad de padres. En este caso en particular, dos de los puntos pertenecían a la parte naranja y el tercero a la parte gris. Por lo que los 3 padres seleccionados son: $y = 4x$, $y = 4x$ e $y = 1$.

- **Operadores de variación (Mutación y Recombinación):** operan sobre los individuos elegidos como padres y generan nuevos genotipos. La elección de qué padres serán afectados es aleatoria. Se le dice mutación a aquellos operadores que necesitan un sólo padre para actuar mientras que recombinación son los operadores que actúan sobre dos individuos.
  - **Mutación:** opera sobre el genotipo de un padre y genera un nuevo individuo o hijo. Es un cambio en una o más partes del genotipo elegidas, tanto el cambio como el lugar son elegidos aleatoriamente. Generalmente, el genotipo del individuo antiguo y el nuevo son similares; sin embargo, el fenotipo puede cambiar drásticamente dependiendo de la representación utilizada. La mutación puede afectar más de una vez a un individuo o ninguna, tal como se muestra en la Figura 2.17. En el ejemplo utilizando expresión de genes, los elementos de la cabeza del genotipo pueden cambiar a elementos del conjunto de terminales y del conjunto de operadores; mientras que los elementos de la cola pueden cambiar a elementos del conjunto de terminales.

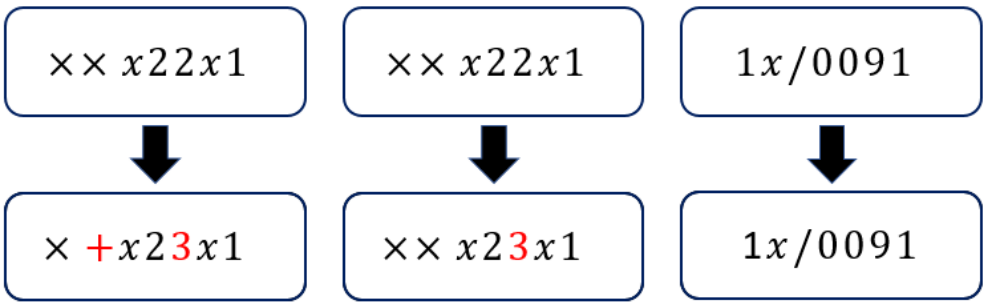


Figura 2.17: Ejemplo de mutación para 3 genotipos, en rojo se observan los cambios producidos. En este ejemplo se realizaron 2, 1 y 0 mutaciones a cada uno de los individuos respectivamente.

– **Recombinación:** usualmente llamado ***crossover*** realiza una combinación de dos padres, para generar dos hijos. Según la representación se tienen distintos tipos de *crossover*. Usualmente un individuo sólo puede ser modificado una vez por cada operador de recombinación, lo cuál se elige aleatoriamente.

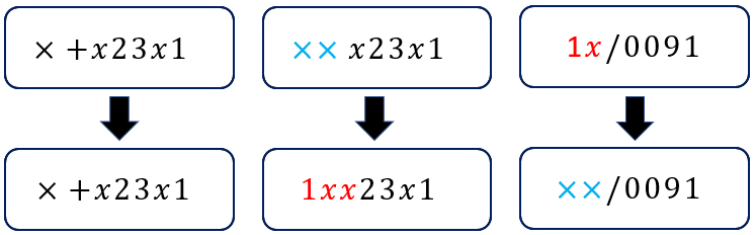


Figura 2.18: Ejemplo de *crossover* entre los individuos resultantes de la mutación de la Figura 2.17. En este caso, hubo recombinación entre el segundo y tercer individuo, mientras que el primer individuo se mantuvo sin cambios.

- **Mecanismo de selección de sobrevivientes (Reemplazo):** se realiza una selección sobre que individuos pasan a la siguiente generación. Al contrario de la selección de padres, en este caso no se realiza en forma aleatoria. Generalmente está la opción de elegir entre los padres e hijos según quién tenga mayor aptitud; y la otra opción es elegir siempre los hijos. En la Figura 2.19 se muestra un ejemplo de la selección de sobrevivientes luego de haber pasado por los operadores anteriores.

G 1	+2/43x1	×× x22x1	1x/0091	y = 10/3	y = 4x	y = 1
Padres	×× x22x1	×× x22x1	1x/0091	y = 4x	y = 4x	y = 1
Hijos	× +x23x1	1xx23x1	××/0091	y = 5x	y = 1	y = 0
G 2	× +x23x1	1xx23x1	××/0091	y = 5x	y = 1	y = 0

Figura 2.19: Ejemplo de selección de los sobrevivientes que pasan a la siguiente generación. En este caso se utilizó la elección de hijos como sobrevivientes.

- **Inicialización:** corresponde a la creación de la primera generación de individuos. Se puede realizar de forma totalmente aleatoria o utilizar un método para asegurar que los individuos iniciales sean distintos y ampliar la búsqueda, ayudando a escapar de óptimos locales.
- **Condición de Término:** debido al funcionamiento heurístico de los algoritmos evolutivos es improbable que un individuo llegue a la solución óptima (con error igual a 0). Debido a esto, generalmente se utilizan los siguientes criterios en caso de que no se llegue al óptimo:
  1. Un tiempo máximo de uso de la CPU.
  2. Un número máximo de evaluaciones del *fitness* de individuos.
  3. El mejoramiento del *fitness* no cambia, respecto a un umbral, por un cierto número de generaciones.
  4. La diversidad de la población baja de un umbral.

En la Figura 2.20 se muestran 4 generaciones de soluciones: $G1$, $G2$, $G10$ y $G30$ que aparecieron al utilizar el algoritmo de expresión de genes para resolver el problema de ejemplo de esta sección. Se espera que mientras las generaciones avancen, las soluciones se acerquen más a $y(x) = x^2$. A la izquierda se observan las soluciones codificadas (genotipos) para que el computador las procese, mientras que a la derecha están las propuestas de solución (fenotipos) al problema de regresión. En la figura se observa que en la primera generación (población inicial) las soluciones propuestas son $y = 10/3$, $y = 4x$, $y = 1$; mientras que en la generación número 30, las soluciones ya se acercan a lo esperado e incluso se encuentra la solución óptima $y(x) = x^2$. En este ejemplo las soluciones mejoran en el transcurso del algoritmo.

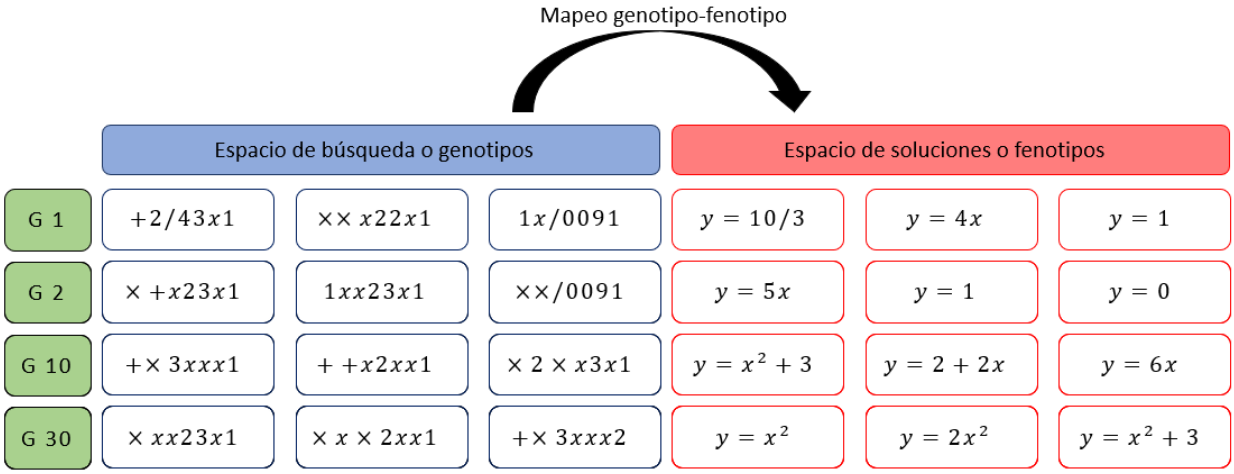


Figura 2.20: Ejemplo de la evolución de una población de 3 individuos. Se muestran solamente las generaciones 1, 2, 10 y 30. Se muestra tanto el genotipo como el fenotipo de cada uno de los individuos, en la izquierda y derecha de la imagen respectivamente.

2.4. Programación Genética (PG)

Esta sección está basada en [12] y explica de manera básica como funciona el algoritmo de programación genética que se utilizó en los trabajos previos. Se describe programación genética para comprender mejor la contribución que genera utilizar el algoritmo de evolución gramatical.

2.4.1. Estructura PG

El **genotipo** es un árbol de tamaño variable que tiene hojas pertenecientes al conjunto de terminales T y nodos intermedios pertenecientes al conjunto de funciones N, donde los elementos de N se eligen teniendo en cuenta los problemas de clausura y suficiencia.

El problema de clausura requiere que todos los elementos de N deben tener la capacidad de manejar cualquier entrada durante el proceso; por lo consiguiente se debe poder evaluar cualquier individuo. Si hay algún caso excepcional que genere una expresión no válida, se puede manejar manualmente entregando valores bajos de aptitud a estos individuos inviables. El problema de

suficiencia indica que el conjunto de terminales y funciones debe ser suficiente para generar una solución posible al problema.

El **fenotipo** es el mismo árbol del genotipo escrito como fórmula, por lo que en este caso no hay un **mapeo genotipo-fenotipo** como tal.

2.4.2. Ejemplo PG

En la Figura 2.21 se muestran dos generaciones consecutivas de una población de 2 individuos. A la izquierda se muestran los genotipos, los cuales fueron modificados por una operación de crossover y posteriormente sufrieron mutaciones mostradas en rojo. A la derecha se observan los fenotipos correspondientes.

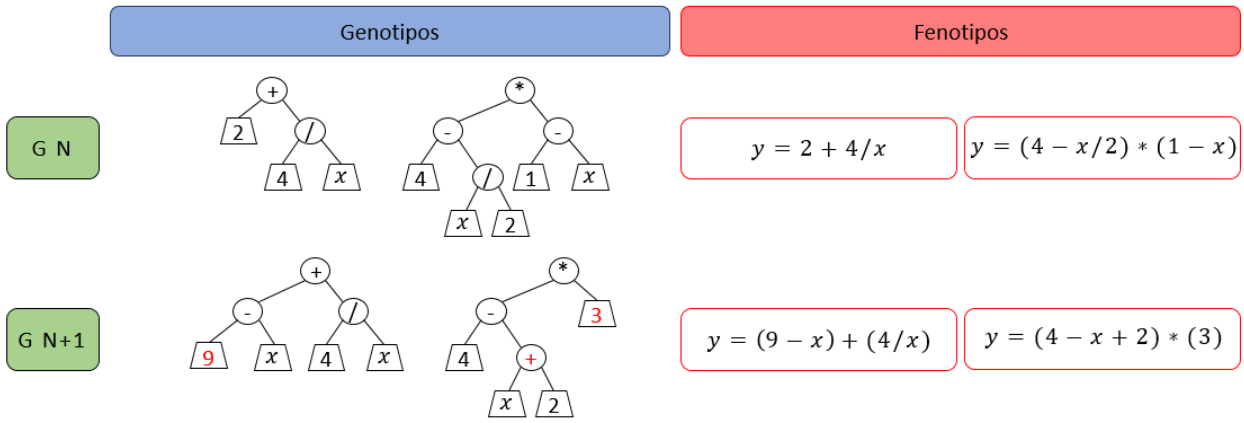


Figura 2.21: Ejemplo de programación genética. En rojo se observan los caracteres que sufrieron mutaciones luego del *crossover*.

2.4.3. Operadores PG

- Reproducción

Se elige un porcentaje de los padres seleccionados y se copian directamente a la siguiente generación, sin pasar por mutaciones o *crossover*.

- Crossover

Dos padres generan 2 hijos. A cada padre se le elige aleatoriamente un punto para realizar la combinación y se intercambian los subárboles bajo esos puntos. Hay que destacar que esto puede aumentar considerablemente la profundidad de los individuos. Y como los dos puntos son distintos, aún cuando los padres sean el mismo individuo se pueden generar distintos hijos, lo que disminuye la convergencia prematura.

Si se elige un tamaño máximo para la profundidad y el hijo fuera a exceder ese largo, se aborta la operación para ese hijo y el padre que comparte la raíz es elegido para su reproducción idéntica

a la siguiente generación.

- Mutación

Un padre genera un hijo. Se elige aleatoriamente un punto en el árbol, se elimina ese punto y todo su subárbol. Luego se inserta un subárbol generado aleatoriamente. Se da un máximo de profundidad al subárbol generado para que no exceda el límite de profundidad a los árboles en la inicialización.

- Permutación

Un padre genera un hijo. Primero se selecciona un elemento que pertenezca a N. Si esa función tiene k argumentos, se elige una permutación entre las $k!$ posibilidades y se ordenan de acuerdo a eso. Esto genera un cambio para las funciones que no son conmutativas.

- Edición

Cuando se utiliza, opera sobre todos los individuos, tiene el fin de simplificar las expresiones. Reduce el número de nodos de los subárboles. Se puede utilizar de dos maneras: para la apariencia, al mostrar el resultado se simplifica las expresiones; o durante el desarrollo para tratar de simplificar el resultado y mejorar el rendimiento. Debido a que es un proceso iterativo sobre cada individuo, consume mucha capacidad computacional. Por lo que se suele utilizar esta operación sólo en algunas generaciones.

- Encapsulación

Consiste en elegir algunos padres aleatoriamente. Y elegir un subárbol de cada padre para encapsularlo y definirlo nuevamente como un nodo terminal. Esto nodo puede ser utilizado en las demás operaciones.

Inicialización

Hay varias formas de generar la primera población, dos formas básicas son: *full method* y *growth method*. El *full* corresponde a generar árboles aleatorios donde todos los caminos entre una hoja y la raíz tienen distancia igual a la profundidad del árbol. Para ello se eligen nodos de función en las cercanías a la raíz y se eligen terminales en el final del árbol. Mientras que en *growth* se determina la profundidad del árbol y luego todos los caminos raíz-hoja son menores o iguales a la profundidad. Para ello se pueden elegir tanto terminales como funciones en las cercanías de la raíz y sólo terminales en las distancias cercanas a la profundidad.

Otro método se llama *ramped half-and-half*, el cual es una combinación de los dos anteriores. Se generan igual cantidad de árboles para distintas profundidades que van entre 2 y la máxima profundidad. Para cada profundidad se genera la mitad de los árboles con *growth* y la otra mitad con *full*.

2.5. Evolución Gramatical (EG)

Esta sección está basada en [13] y se explica de manera básica como funciona el algoritmo que es utilizado en este trabajo para encontrar las soluciones. Cabe destacar que gran parte del trabajo de título consistió en la correcta implementación de este algoritmo y el estudio de su funcionamiento. En la Sección 3 se describe el algoritmo ajustado para la problemática particular de este trabajo.

2.5.1. Estructura EG

El **genotipo** corresponde a una secuencia de números y puede tener largo variable. La secuencia se divide en "codones" que son grupos consecutivos generalmente de 8 bits y representan un número entero entre 0 y $2^{\text{numero bits}}$-1. El **fenotipo** es un programa que en esta aplicación se puede representar con un árbol de expresión de tamaño variable.

Para el **mapeo genotipo-fenotipo** se debe definir una gramática *Backus-Naur Form* (BNF) de un lenguaje, la cuál consiste en:

- **Terminales:** son caracteres que pueden aparecer en el lenguaje.
- **No-terminales:** son caracteres que no pueden aparecer en el lenguaje, se pueden expandir en otros no-terminales y/o terminales a través de ciertas reglas.

Así, se puede representar una gramática con una tupla $\{N,T,P,S\}$. Donde N es el conjunto de no-terminales, T el conjunto de terminales, P es un conjunto de reglas que mapean los elementos de N a T y S es el símbolo inicial que se encuentra en el conjunto N. En la Figura 2.22 se ilustra un ejemplo de gramática.

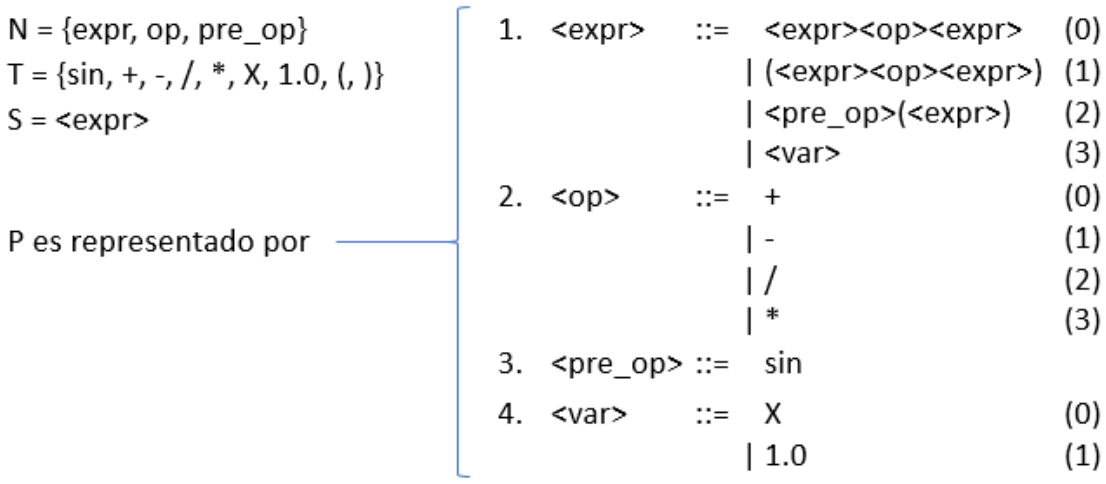


Figura 2.22: A la izquierda se encuentra el conjunto de terminales (T), no-terminales (N) y el símbolo inicial (S). A la derecha las reglas P que indican las transformaciones que puede tener cada elemento de N, cada transformación se separa por | y están enumeradas desde 0 hasta el número de opciones menos uno. Ejemplo rediseñado, extraído de [13].

En la Tabla 2.1 se muestra el número de opciones en las que se puede expandir cada no-terminal del ejemplo de la Figura 2.22.

Tabla 2.1: Número de transformaciones que puede tener cada elemento de N. Ejemplo rediseñado, extraído de [13].

Regla número	Número de opciones
1	4
2	4
3	1
4	2

Una vez que se tiene definida la gramática a utilizar, se realiza el procedimiento para mapear el símbolo S a un árbol de expresión a través de reglas decididas por el genotipo. Como se mencionó anteriormente, el genotipo se divide en codones que representan números enteros. Luego se realiza de manera iterativa lo siguiente: de la expresión que se tiene (en un comienzo es igual a S), se observa el número de opciones en las que se puede transformar el elemento de N que se encuentre más a la izquierda de la expresión. Si el número de opciones O es mayor a 1, entonces se toma el valor C del siguiente codón (partiendo por la izquierda) y se elige la regla tomando el resto de la división C/O.

En este proceso puede ocurrir que se acaben los codones y la expresión aún tenga símbolos de N. En ese caso se reutilizan los codones desde el comienzo, lo que se denomina **encapsulamiento**. Si luego de ciertas iteraciones aún hay un mapeo incompleto, se le otorga el mínimo valor posible de aptitud a ese individuo y se le cataloga como inviable. También se destaca que no siempre se utilizan todos los codones del genotipo, lo que genera que haya una región no codificada.

2.5.2. Ejemplo EG

En la Figura 2.23 se observan dos generaciones consecutivas de una población de 2 individuos. En el lado izquierdo se muestra el genotipo de cada individuo, los cuales fueron afectados por una operación de *crossover* seguida de mutaciones (mostradas en rojo). Y en el lado derecho se muestran los fenotipos correspondientes.

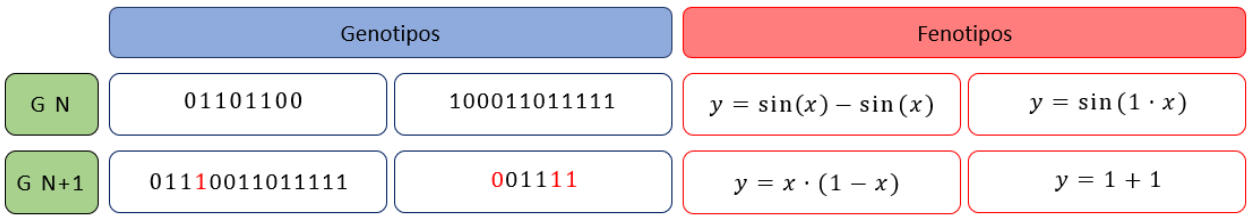


Figura 2.23: Ejemplo de evolución gramatical. En rojo se muestran las mutaciones que ocurren en el genotipo luego de una operación de *crossover*.

2.5.3. Operadores EG

Debido a que el genotipo es una secuencia de caracteres, se pueden utilizar los operadores de mutación y *crossover* estándares.

- Duplicación de codón

Como no importa el largo de la secuencia, se puede agregar un operador que lo modifique. De forma aleatoria se elige el número de codones a duplicar y el codón de inicio. Luego se copian los codones correspondientes y se colocan al final del cromosoma.

Inicialización

Por el tipo de genotipo se pueden utilizar inicializaciones estándares de algoritmos genéticos.

2.6. Resultados de Trabajos Anteriores

En esta sección se describen los trabajos anteriores para el diseño óptimo de empaquetamiento de baterías ión litio. El año 2014 Reyes reportó en [2] el modelo paramétrico de la Sección 2.2 que permite obtener variables del estado de una batería en función de la configuración de sus celdas, de la corriente que circula por ella y del flujo de aire que entra al *pack*. El desempeño de este modelo base fue obtenido comparando el error de las variables de salida: temperatura de las celdas centrales, velocidad del fluido y presión del fluido, respecto a lo entregado por el *software* computacional CFD ANSYS que entrega resultados precisos respecto a la teoría de fluidodinámica.

Con el objetivo de mejorar el desempeño del modelo base, el año 2015 en [3] Villa analizó el reporte paramétrico de Reyes y seleccionó 3 expresiones del modelo para ajustar: el número de Nusselt, el coeficiente de fricción y el coeficiente de arrastre. Villa eligió estos 3 parámetros debido a que Reyes se basó en datos experimentales de otras situaciones que podrían no ajustarse bien a las condiciones de un banco de baterías, existiendo la posibilidad de que tengan un margen de error que mejorar. En el modelo base realizado por Reyes [2] se utilizaron las siguientes expresiones:

$$\text{Cdrag}^R = 3.3481\text{Rem}^{0.179} \tag{2.12}$$

$$\text{Friction}^R = 0.7 \cdot A(S)\text{Rem}^{B(S)} \tag{2.13}$$

$$\text{Nu}^R = C'(\text{Rem}, \text{indCol})\text{Rem}^{0.653}, \tag{2.14}$$

donde Cdrag^R, Friction^R y Nu^R son los coeficientes de arrastre, fricción y número de Nusselt respectivamente; y el superíndice R es la notación que indica que son las expresiones utilizadas por Reyes. Rem es el número de Reynolds, $A(S)$ y $B(S)$ son funciones lineales por tramo en función de la separación entre celdas, dadas por la Tabla 2.2; y C' varía en función del tramo en que se encuentre el número de Reynolds y de la columna (indCol) en el banco en que se evalúa, según lo descrito en la Tabla 2.3.

Tabla 2.2: $A(S)$ y $B(S)$ en función de la separación entre celdas para el modelo fenomenológico original.

Separación	0,25	0,5	1	1,5
A	82,188	38,446	11,728	1,2095
B	-0,605	-0,54	-0,402	-0,211

Tabla 2.3: Factor de Corrección del número de Nusselt en función de la columna de fluido evaluada.

Columna	1	2	3	4	5	7	10	13	16	$\geq$20
C'	0,64	0,76	0,84	0,89	0,92	0,95	0,97	0,98	0,99	1

Para ajustar las expresiones se utilizó programación genética, descrita en la Sección 2.4, y el *toolbox* de MATLAB GPlab [14]. Debido a que las tres expresiones se desean optimizar simultáneamente y todas dependen de la configuración, se tiene que en un principio el problema a resolver es multivariable y multiobjetivo. Sin embargo, a causa del costo computacional y limitaciones del *software* utilizado, Villa acotó el problema estudiando únicamente el caso para una configuración de 5 celdas y tratándolo como si fuera univariable y mono-objetivo de la siguiente manera:

- Para obtener la expresión del coeficiente de arrastre, minimizó el error de la velocidad del fluido a la salida, manteniendo las expresiones para el coeficiente de fricción y el número de Nusselt del modelo base.
- Para obtener la expresión del coeficiente de fricción, minimizó el error de la presión del fluido a la salida, manteniendo las expresiones para el coeficiente de arrastre y el número de Nusselt del modelo base.
- Para obtener la expresión del número de Nusselt, minimizó el error de la temperatura de la celda central a la salida, manteniendo las expresiones para el coeficiente de arrastre y el coeficiente de fricción del modelo base.

Finalmente, Villa encontró las siguientes expresiones:

$$\mathrm{Cdrag}^{\mathrm{FV}} = 4.78\Big(0.636\big(4.81\mathrm{Rem}^{2.3\cdot 0.49^{\mathrm{An}}}\big)^{\mathrm{Dfn}^{\mathrm{An}/\mathrm{Rem}}}\Big)^{0.0148} \tag{2.15}$$

$$\mathrm{Friction}^{\mathrm{FV}} = 4\Big(1.021\cdot(9\mathrm{S})^{(2.81\mathrm{S})^{-2}} + \big(\frac{0.3021}{\mathrm{S}^2}\big)^2 - 0.6\Big)\mathrm{Rem}^{-0.2655} \tag{2.16}$$

$$\mathrm{Nu}^{\mathrm{FV}} = \mathrm{Rem}^{0.677}, \tag{2.17}$$

donde FV denota el modelo de Villa, Cdrag es el coeficiente de arrastre, Friction es el coeficiente de fricción, Nu es el número de Nusselt, Rem es el número de Reynolds, S es la separación de

celdas, Dfn es la densidad del aire normalizada por una densidad de referencia y An es la razón entre el área superficial de la celda y el área por donde pasa el fluido. Si bien las expresiones encontradas por Villa tienen un mejor desempeño que el modelo base, sus resultados aún difieren significativamente de lo proporcionado por ANSYS, por ello se continuaron los trabajos en busca de una mejor aproximación.

El 2016 Yon [4] mejoró la eficiencia computacional migrando el código desde MATLAB a Java utilizando el *toolkit* ECJ [15]. Y el 2017 en [6] se implementó un algoritmo evolutivo multiobjetivo NSGA-II [16] por sobre el esquema de programación genética. Esto permitió optimizar las tres expresiones de interés simultáneamente. Con los aportes anteriores, el año 2018 en [5] García y Tapia abordaron el problema de manera multi-objetivo y con el código de Java. Primero analizaron los resultados de Villa y decidieron modificar los rangos de estudio para la separación de las celdas de [0, 1.5] a [0.3, 1.5], y el rango de flujo de entrada de [0, 200] a [1, 200] para eliminar los casos de mayor error. Finalmente seleccionaron las siguientes expresiones:

$$\text{Cdrag}^{GT} = 0{,}891 \cdot \text{An} - \text{An}^2 + \text{Rem}^{0{,}1452} \tag{2.18}$$

$$\text{Friction}^{GT} = \text{Dfn} \cdot \text{S}^{(24{,}5261\text{Rem}^{-0{,}3891}-2)} \tag{2.19}$$

$$\text{Nu}^{GT} = 2{,}0232\text{Rem}^{0{,}5528}, \tag{2.20}$$

donde GT denota el modelo de García-Tapia [5]. Rem es el número de Reynolds, S es la separación de celdas, Dfn es la densidad del aire normalizada por una densidad de referencia y An es la razón entre el área superficial de la celda y el área por donde pasa el fluido.

En el trabajo inmediatamente anterior a esta memoria, en el año 2019 Aguilar [7] abordó el problema manteniendo la metodología de García-Tapia pero aumentando el estudio a distintos números de celdas. Anteriormente se había ajustado el modelo únicamente a un banco de baterías de 5 celdas y Aguilar realizó ajustes para 5, 53, 102 y 151 celdas. Para ello, se tuvo que simular los modelos de esas configuraciones en el *software* ANSYS y obtener los datos según distintas variables de entrada, aumentando la base de datos. Así se logró estudiar la extrapolación e interpolación a distinto número de celdas de los modelos ajustados por los trabajos anteriores. Se encontró que los modelos ajustados no se desempeñaban de igual manera al cambiar el número de celdas, por este motivo en [7] se agregó como variable de entrada al modelo el número de columna donde se evalúan las expresiones. Se encontraron distintas expresiones dependiendo de la función de *fitness* u objetivo elegido. Los objetivos estudiados por Aguilar fueron: minimizar el error de la velocidad del fluido, minimizar el error de la presión del fluido, minimizar el error de la temperatura de la celda central y minimizar el producto de los 3 errores anteriores. A continuación, se muestran las expresiones obtenidas para el último objetivo mencionado en el modelo de 102 celdas:

$$\text{Cdrag}^{\text{NA}} = 1.4052 \tag{2.21}$$

$$\begin{aligned}\text{Friction}^{\text{NA}} = &\left(\frac{0.0762}{\text{Vmfn}^2}\right)^{\text{Vmfn}^{S^2}} \\ &+ \left(\text{Dfn} + \left(\frac{0.0199}{\text{Vmfn}^2} - (0.2609\text{Vmfn}^2)\right)^4 - \left((\text{S}^2)^{(0.0422/\text{Vmfn})} - \text{S}^2\right)^2\right)^2 \\ &+ \frac{0.0762}{\text{Vmfn}^2} - 0.0269\end{aligned} \tag{2.22}$$

$$\text{Nu}^{\text{NA}} = \text{Rem}^{\left(\left(0.5702\cdot\left(\text{indCol}+0.5702\cdot\text{Rem}^{(0.9403^{\text{indCol}})}\right)^{((3.069\cdot\text{indCol})/\text{Rem}+0.8519)}\right)/\text{Rem}+0.4857\right)}, \tag{2.23}$$

donde NA denota el modelo de Aguilar [7]. Rem es el número de Reynolds, S es la separación de celdas, Dfn es la densidad del aire normalizada por una densidad de referencia y Vmfn es la velocidad media del fluido normalizada por una velocidad de referencia.

Con lo anterior se concluye el resumen de los trabajos previos. En las secciones siguientes se mostrará como se encuentran nuevas expresiones para estos parámetros y se analizará su desempeño.

Capítulo 3

Metodología e Implementación

El objetivo general de esta memoria es ajustar y analizar un modelo fenomenológico de empaquetamiento de baterías usando evolución gramatical. Como se mencionó, este tema ha sido abordado en trabajos anteriores con distintos enfoques y en cada uno se ha encontrado una solución posible al problema. En este capítulo se describe cómo se obtuvieron nuevas expresiones utilizando evolución gramatical manteniendo su baja complejidad matemática. Luego se define un método para validar los modelos según su interpretabilidad física. Para hacer referencia a cada individuo, se utiliza la siguiente nomenclatura:

$$\text{Ind\_}X = \begin{pmatrix} \text{Cdrag}^X \\ \text{Friction}^X \\ \text{Nu}^X \end{pmatrix}, \tag{3.1}$$

donde Ind_X es el individuo en el modelo de X conformado por los 3 parámetros de interés Cdrag^X, Friction^X y Nu^X. Cdrag, Friction y Nu corresponden al coeficiente de arrastre, coeficiente de fricción y número de Nusselt respectivamente. X puede ser {R, FV, GT, NA o FB} dependiendo si fue encontrado en el trabajo de Jorge Reyes, Francisco Villa, García y Tapia, Nicolás Aguilar o Francisco Born, respectivamente. En este capítulo se describe cómo se obtuvo Ind_FB.

3.1. Obtención de expresiones con evolución gramatical

Al inicio del informe se propuso el uso de evolución gramatical como algoritmo para encontrar nuevas soluciones. Si bien los últimos trabajos están programados completamente en el lenguaje Java, se decidió utilizar el lenguaje Python debido a su simplicidad. Sin embargo, como el modelo hecho en Java ya está optimizado para su velocidad de ejecución y disminuye considerablemente el tiempo en comparación a Python, estos se combinaron para obtener un código más entendible sin que aumente el tiempo de ejecución. Así el algoritmo evolutivo está programado en Python y hace un llamado al programa de Java cada vez que se requiera evaluar un individuo con el modelo fenomenológico.

En [17] se menciona la librería PonyGE para Python que se encuentra en su segunda versión: PonyGE2 [18] la cual se utilizó en este trabajo como base para el algoritmo de evolución gramatical. Cualquier duda al respecto, en la página de la librería `https://github.com/PonyGE/PonyGE2/wiki` se explica su uso.

Para encontrar Ind_FB utilizando la librería anterior, se dividió el proceso en etapas de distinta complejidad para observar de mejor manera el comportamiento del algoritmo. Esto permitió asegurar su funcionamiento antes de utilizar el algoritmo para resolver el problema original. La primera etapa fue una prueba preliminar con un ejemplo que viene incluido en la librería utilizada, luego se realizaron dos etapas que consideran una aproximación al problema original utilizando una base de datos generada con funciones conocidas y la última etapa es la obtención de Ind_FB.

3.1.1. Prueba preliminar

La primera etapa tiene por objetivo estudiar el funcionamiento de la librería y familiarizarse con su uso. Se realizó el ejemplo de regresión que aparece en la página de la librería ya que es el que más se asemeja al problema real de este trabajo. En el ejemplo se tiene una base de datos con 1024 puntos, cada uno compuesto por 5 variables de entrada ($x[0]$, $x[1]$, $x[2]$, $x[3]$ y $x[4]$) y una variable de salida ($y(x)$), así el algoritmo busca la función de 5 variables que pueda obtener la variable de salida. La métrica utilizada en este ejemplo es el error cuadrático medio entre la salida teórica del conjunto de datos y la salida obtenida con las soluciones encontradas. Debido a eso, en este caso entre más cercana esté la métrica a 0, mejor es la solución. Al correr el programa con los parámetros que vienen por defecto para este ejemplo, se encuentra que el mejor individuo corresponde a la fórmula:

$$y(x) = tanh\left(sen(\sqrt{x[3]}) \cdot sen\left(\sqrt{sen(\sqrt{x[2]}) \cdot sen\big(sen(\sqrt{x[0]})\big)}\right)\right), \tag{3.2}$$

la cuál al medir su desempeño tiene un *fitness* de 0.027065. Como la fórmula obtenida no es fácil de interpretar se procedió a graficar la salida teórica en color verde con la salida obtenida con la ecuación anterior en color azul. Así se obtuvo la Figura 3.1 donde se observa como la función trata de ajustarse a los datos. No se puede realizar un estudio más exhaustivo debido a que no se dispone de la fórmula real con la que se generaron los datos de salida, por lo que puede que la diferencia entre las curvas sea debido a que el conjunto de datos de ejemplo tenga ruido o que los parámetros de la evolución no eran los adecuados, por lo que sería necesario realizar pruebas extras con un conjunto de datos controlado sin ruido y con conocimiento de la expresión original.

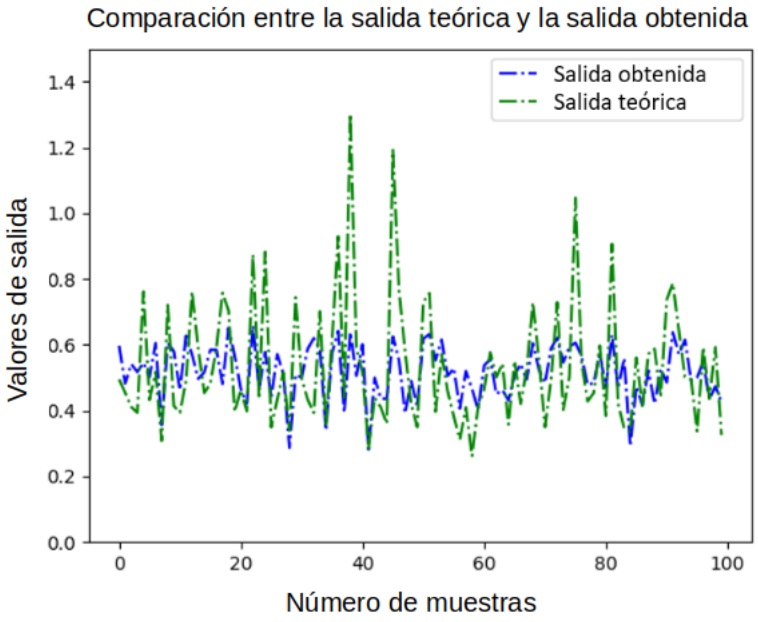


Figura 3.1: Comparación de la curva teórica con la curva obtenida en el ejemplo de regresión de la librería PonyGE2.

3.1.2. Primera aproximación

La segunda etapa considera una versión simplificada del problema original. El objetivo es estudiar el alcance del algoritmo y realizar ajustes a la implementación de este hasta que sea capaz de resolver el problema de una manera adecuada. Para esta aproximación se generó una base de datos con ecuaciones conocidas, de manera de comprobar si el algoritmo logra encontrar estas ecuaciones.

Para simplificar el problema se eligió al individuo Ind_GT (que contiene las ecuaciones 2.18-2.20) como objetivo de búsqueda del algoritmo, esto debido a que sus expresiones son más simples que las de otros trabajos y porque fue entrenado en un modelo compuesto por un bloque de 5 celdas, por lo que su evaluación es más rápida. Luego se escogió aleatoriamente un conjunto de 100 muestras que sirven de entrada al modelo fenomenológico, se evaluó utilizando el modelo de García-Tapia y se guardaron las salidas respectivas. Así se generó un conjunto de datos sin ruido. Por esta razón se espera poder obtener expresiones similares con EG. En la primera aproximación se consideraron conocidos Friction^{GT} y Nu^{GT}, por lo que el algoritmo debía obtener la expresión para Cdrag^{GT}. Se ajustó la implementación del algoritmo de la librería PonyGE2 para que sea capaz de evaluar el modelo. El algoritmo utilizado se describe en la Sección 3.1.4 y la expresión para el coeficiente de arrastre encontrada en esta aproximación se muestra en la Sección 4.3.

3.1.3. Segunda aproximación

Esta etapa es similar a la anterior y se utilizó la misma base de datos. La diferencia es que en esta aproximación no se consideran conocidos el coeficiente de fricción y el número de Nusselt. El

algoritmo debe encontrar $Cdrag^{GT}$, $Friction^{GT}$ y Nu^{GT} simultáneamente. El algoritmo utilizado se describe en la Sección 3.1.4 y las expresiones encontradas se muestran en la Sección 4.4.

3.1.4. Implementación de evolución gramatical utilizada

Para resolver la primera y segunda aproximación se ajustó el algoritmo de evolución gramatical que viene por defecto en la librería PonyGE2. Este algoritmo también se utilizó para obtener Ind_FB a excepción de modificaciones en la gramática que se describen en la sección 3.1.6. Como cada individuo debe representar 3 funciones que evolucionan simultáneamente, lo primero que se realizó fue ajustar la representación de individuos a un enfoque de multigen con 1 gen por cada función. En la Figura 3.2 se muestra un ejemplo del genotipo de un individuo con 3 genes, donde cada gen es una secuencia de codones de números enteros.

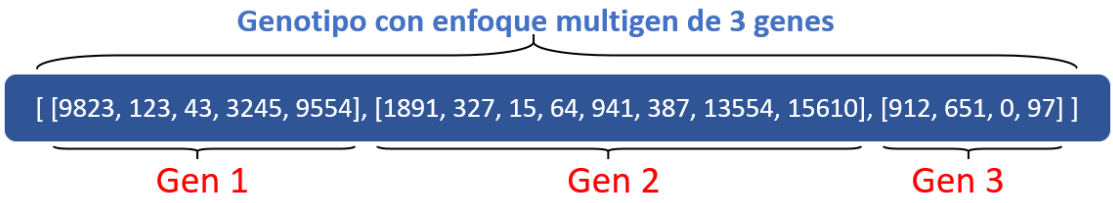


Figura 3.2: Ejemplo de individuo con un genotipo de 3 genes. El primer gen está asociado al coeficiente de arrastre, el segundo gen al coeficiente de fricción y el tercer gen al número de Nusselt.

Para el mapeo genotipo-fenotipo se utilizaron 3 gramáticas distintas: para el coeficiente de arrastre, coeficiente de fricción y el número de Nusselt, respectivamente. Cada uno de los genes del genotipo indica las reglas que se utilizan en su respectiva gramática para generar el fenotipo. Las gramáticas se diseñaron según las variables que aparecen en las expresiones objetivo. El diseño de la gramática es de gran importancia al utilizar este algoritmo, por ello se describe separadamente en la Sección 3.1.5.

Una vez definida la representación multigen de los individuos y diseñadas las gramáticas, se ajustaron los operadores del algoritmo como se describe posteriormente. De manera general el algoritmo utilizado en esta memoria de título sigue los siguientes pasos:

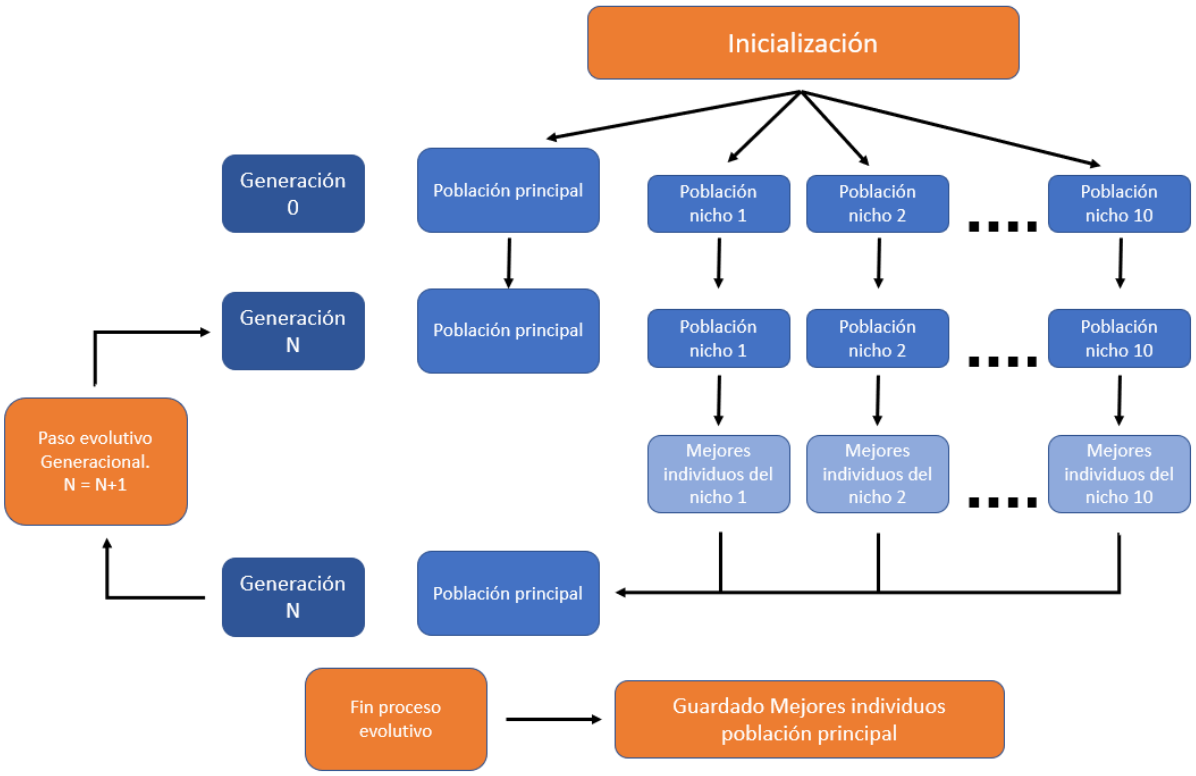


Figura 3.3: Interacción entre las poblaciones nicho y la población principal. En cada generación se seleccionan los mejores individuos de cada población nicho para agregarlos a la población principal y aumentar su diversidad.

- **Inicialización de la población principal:** es una población de individuos que sigue un proceso evolutivo propio y además recibe una copia de los mejores individuos de las poblaciones nicho. En la Figura 3.3 se muestra como evolucionan las distintas poblaciones en paralelo y cómo al final de cada paso evolutivo se combinan las poblaciones nicho en la principal. Debido a la combinación de poblaciones se obtiene un aumento de la diversidad en la población principal y se recorre de mejor manera el espacio de soluciones. Para inicializar un conjunto de individuos se itera el siguiente proceso hasta completar el tamaño de la población: se genera un genotipo válido con codones totalmente aleatorios, luego se obtiene el fenotipo correspondiente y se evalúa su *fitness*. Si este es un valor válido se agrega el individuo al conjunto inicial, esto se realiza ya que al generar un genotipo de manera aleatoria puede ocurrir que el *fitness* de un individuo sea infinito. Si bien los individuos con *fitness* inválido son removidos rápidamente debido al proceso evolutivo, se pierde diversidad en las primeras generaciones y puede causar estancamiento prematuro en soluciones locales.
- **Inicialización de las poblaciones utilizadas como nichos:** son 10 poblaciones de menor tamaño que la principal, las cuáles tienen su propio proceso evolutivo que no es afectado por ninguna otra población. Debido a ello se espera que cada población nicho evolucione hacia un óptimo distinto. El método de inicialización es el mismo que para la población principal, sólo varía en el número de individuos. La Figura 3.3 muestra un diagrama con la interacción entre las poblaciones nicho y la población principal durante el proceso evolutivo. En el diagrama se puede ver que al final de cada generación, se realiza una copia de los mejores individuos

de cada población nicho y se agregan a la población principal. Para mantener el tamaño de la población principal, se eliminan los individuos que tienen los valores más altos de *fitness*.

- **Evaluación de todos los individuos:** los individuos de todas las poblaciones anteriores son evaluados de acuerdo a la métrica de desempeño para poder iniciar el proceso evolutivo generacional. El fenotipo de un individuo consiste en las tres expresiones en forma de fórmula matemática: el coeficiente de arrastre, coeficiente de fricción y el número de Nusselt. Junto a estas expresiones se entregan las entradas de la base de datos al modelo fenomenológico y se evalúan las salidas correspondientes. Una vez hecho esto, se comparan las salidas teóricas con las salidas obtenidas, utilizando la métrica de desempeño. Se definió una métrica de desempeño que cuantifica la diferencia entre las salidas del modelo utilizando las expresiones evolucionadas *versus* las generadas con Ind_GT. Como se señaló en la sección 2.1 las salidas del modelo son la velocidad del fluido, presión del fluido y temperatura de celda. La métrica utilizada considera el error cuadrático medio para cada variable:

$$MSE\_Y(\mathrm{eval}) = \frac{1}{N}\sum_{k=1}^{N}\left(\frac{Y_k - Y_{eval,k}}{mean\_Y}\right)^2, \tag{3.3}$$

donde $MSE\_Y(eval)$ es el error cuadrático medio entre la salida deseada y la obtenida con la expresión que está siendo evaluada (ambas normalizadas por $mean\_Y$), $mean\_Y$ es el promedio de los valores de salida teóricos respectivos, $eval$ es el individuo o solución a evaluar, Y es la variable de salida (V_f, P_f o T_c), Y_k es la k-ésima salida teórica de la base de datos, $Y_{eval,k}$ es la k-ésima salida obtenida con las expresiones evaluadas, k es el número de muestra y N es el total de muestras. Finalmente, la función de *fitness* considera las 3 variables:

$$sum\_MSE(\mathrm{eval}) = MSE\_V_f(\mathrm{eval}) + MSE\_P_f(\mathrm{eval}) + MSE\_T_c(\mathrm{eval}), \tag{3.4}$$

$$max\_MSE(\mathrm{eval}) = max(MSE\_V_f(\mathrm{eval}), MSE\_P_f(\mathrm{eval}), MSE\_T_c(\mathrm{eval})), \tag{3.5}$$

$$fit(\mathrm{eval}) = 0{,}5 * sum\_MSE(\mathrm{eval}) + 0{,}5 * max\_MSE(\mathrm{eval}), \tag{3.6}$$

donde $fit(eval)$ es el valor de la métrica de desempeño del individuo $eval$, está compuesta por dos partes: $sum\_MSE(eval)$ y $max\_MSE(\mathrm{eval})$. La primera mide el desempeño general del individuo en ajustar las curvas de velocidad, presión y temperatura; mientras que la segunda favorece soluciones que tienen buen desempeño en todas las curvas al mismo tiempo. Mientras más cercano sea el *fitness* a 0, se tiene un mejor individuo, así un valor de 0 implica que se encontraron exactamente las mismas expresiones.

- **Paso evolutivo generacional:** este es un proceso iterativo que obtiene nuevas generaciones de cada población. A partir de una generación de individuos N, se utilizan los operadores de *crossover* y mutación para obtener una generación de individuos $N + 1$ diferente. Se espera que a medida que aumente N el desempeño de los individuos mejore. Se detiene el proceso iterativo cuando no se encuentra mayor mejora en el mejor individuo por un tiempo determinado. Como se mencionó anteriormente las poblaciones nicho y la población principal siguen este proceso. En la Figura 3.4 se muestra un diagrama del paso evolutivo y se puede observar cómo al final de cada iteración se copian los mejores individuos de las poblaciones nicho a la población principal para mantener una alta diversidad de fenotipos. El paso evolutivo consiste en las siguientes etapas:

1. **Elitismo:** se ordenan los individuos según su *fitness* y se guardan los primeros individuos que tengan fenotipo distinto. El tamaño del elitismo usado es de 1 % del tamaño de la población principal.

2. **Selección de padres:** para elegir los padres se utilizó la selección por torneo de tamaño 2, donde del par que se elige, se selecciona el individuo con mayor *fitness* con una probabilidad de 0.9 y al individuo de menor *fitness* con 0.1.

3. ***Crossover*:** se utilizaron 2 tipos de *crossover* mutuamente excluyentes. Con 0.3 de probabilidad se utiliza un *crossover* de genes completos el cual intercambia uno de los tres genes entre ambos padres. Mientras que con 0.7 se utiliza un *crossover* de dos puntos fijo entre cada uno de los 3 genes. Dentro de este último hay una probabilidad de 0.8 a que ocurra el *crossover* y 0.2 que se retorne el individuo sin cambios.

4. **Mutación:** 90 % de los individuos retornados de la operación del *crossover* pasan por el proceso de mutación, mientras que al 10 % restante no se le aplica. Dentro de los individuos a los que se le aplica la mutación se tiene que cada codón tiene una probabilidad de 0.05 para cambiar por un valor aleatorio entre todos los posibles.

5. **Selección de descendencia:** siempre se eligen a los hijos como la descendencia para pasar a la siguiente generación, sin importar si el valor de su *fitness* mejora o no.

6. **Evaluación de los individuos:** luego de realizar cambios al genotipo de los individuos con la mutación y *crossover*, se debe realizar un mapeo genotipo-fenotipo. El fenotipo obtenido se evalúa para obtener su *fitness* según la métrica de desempeño definida anteriormente. Cada 5 generaciones se realizó un proceso adicional de **optimización de constantes** que se describe al final de esta sección.

7. **Revisión elitismo:** se revisa dentro de los individuos retornados por el paso anterior si se encuentran todos los guardados al inicio del paso generacional. En caso que no se encuentren se agregan forzadamente quitando a los individuos de peor *fitness* para mantener el tamaño de la población.

8. **Copia de individuos:** es distinto para las poblaciones nicho y la principal. Se copian los mejores individuos de cada población nicho y se agregan a la población principal. Para ello se quitan los individuos con peor desempeño de la principal, manteniendo su tamaño.

- **Guardado de los mejores individuos:** basta con guardar la población principal debido a que esta ya contiene a los mejores individuos de las poblaciones nichos. Los individuos guardados son analizados posteriormente según su métrica de desempeño y su complejidad.

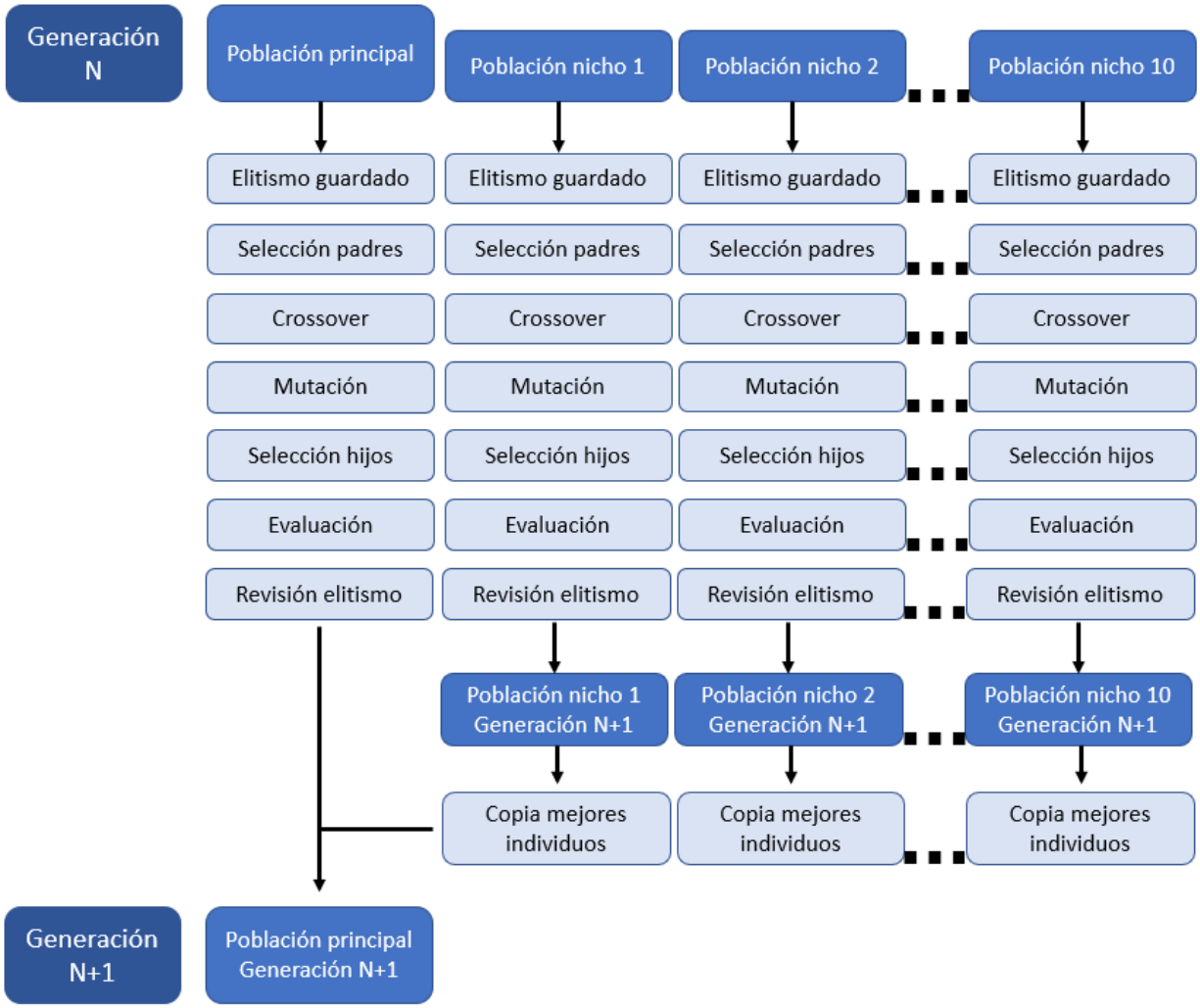


Figura 3.4: Paso evolutivo que siguen las poblaciones de individuos. Tanto las poblaciones nicho como la población principal tienen su propio proceso de elitismo, *crossover* y mutación. Al final de ese proceso se agregan copias de los mejores individuos de cada población nicho a la población principal.

Como se mencionó durante la descripción del paso evolutivo, cada 5 generaciones se realizó una optimización de las constantes a cada uno de los fenotipos de esa generación. Se utilizó la función *minimize* de la librería *scipy.optimize*, el detalle de la función se encuentra disponible en `docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html`. Dentro de la función se eligió el método de optimización "L-BFGS-B", el cuál busca las mejores constantes en la vecindad de un punto inicial que mejoran la métrica de desempeño. Para ello, desde el punto inicial se modifican las constantes con pasos pequeños y una constante a la vez. Cada vez que se modifica una constante se calcula su métrica de desempeño, de esta forma el optimizador calcula un gradiente de descenso para la función de *fitness* y elige nuevos pasos según ello. El método permite colocar límites a las constantes para que se mantengan dentro de cierto rango. La información para el punto inicial y los límites de las constantes está dada en el fenotipo del individuo y se detalla en la Sección 3.1.5. El proceso de optimización tiene varios cri-

terios de término, en particular se termina si el gradiente calculado es 0 por un número definido de evaluaciones, lo que se detecta como un mínimo local y se retornan los valores de las constantes.

En este último punto es importante destacar que la función *scipy.optimize.minimize* no tiene memoria de todas las evaluaciones que realiza durante la optimización y retorna el valor de las constantes que hay en el momento del criterio de término. Debido a la dependencia fuerte de las constantes en el modelo fenomenológico, un pequeño cambio en ellas puede producir la divergencia de la función de *fitness*. Esto puede causar que el optimizador finalice y retorne constantes que empeoran el *fitness* del individuo lo cual es opuesto a lo que se busca. Por este motivo se modificó la implementación del minimizador para que guarde en todo momento las constantes con mejor resultado. Por lo que una vez que se activa el criterio de término del optimizador, se retornan las constantes que se asocian al mínimo *fitness* calculado durante el proceso de optimización.

3.1.5. Diseño de gramática

Esta sección muestra la relevancia de la gramática sobre los resultados y cómo fue diseñada para cada uno de los genes. Para su diseño se tomó en cuenta lo siguiente:

1. **Variables dependientes:** para cada parámetro de interés se utilizaron distintas variables dependientes. Para la primera y segunda aproximación se utilizaron {An, Rem} como variables para el coeficiente de arrastre, {Dfn, S, Rem} para el coeficiente de fricción y {Rem} para el número de Nusselt. Para la resolución del problema original descrito en la Sección 3.1.6, se utilizaron {An, Rem, S} para el coeficiente de arrastre, {S, Rem, Vmfn} para el coeficiente de fricción y {Rem} para el número de Nusselt.

   Donde **An** es el área normalizada del manto de la celda que interactúa con el fluido, **Rem** es el número de Reynolds medio en el lugar de medición, **Dfn** es la densidad del fluido normalizada, **S** es la separación entre las celdas normalizada por el diámetro de celda y **Vmfn** es la velocidad media del fluido normalizada por una velocidad de referencia. Se destaca que todas las variables son adimensionales para que no haya problemas dimensionales en la interacción entre ellas al generar la función.
2. **Complejidad:** se restringió el número de sumandos a 3, donde cada sumando puede ser una multiplicación de algunas variables dependientes o constante. Para el problema original se restringió la posibilidad de que un exponente se encuentre elevado nuevamente, evitando casos del tipo $\mathrm{An}^{\mathrm{Rem}^4}$. Este tipo de casos se permitió para el coeficiente de fricción en la segunda aproximación debido a que $\mathrm{Friction}^{\mathrm{GT}}$ pertenece a este conjunto.
3. **Sentido físico:** cada variable dependiente puede estar elevada a un exponente que difiere según la variable. Esto último impide que hayan casos del tipo $\mathrm{Rem}^{\mathrm{Rem}}$ o $\mathrm{An}^{\mathrm{An}}$ que carecen de sentido.

Para ejemplificar, en la Figura 3.5 se muestra la gramática utilizada para el coeficiente de arrastre en la primera y segunda aproximación. Mientras que la totalidad de gramáticas diseñadas se encuentran en el **Anexo A**.

```
<start> ::=    <expr> + <expr> + <expr>|
               <expr> + <expr> + c[<idx>]*(<expr>)|
               <expr> + c[<idx>]*(<expr>) + <expr>|
               <expr> + c[<idx>]*(<expr>) + c[<idx>]*(<expr>)|
               c[<idx>]*(<expr>) + <expr> + <expr>|
               c[<idx>]*(<expr>) + <expr> + c[<idx>]*(<expr>)|
               c[<idx>]*(<expr>) + c[<idx>]*(<expr>) + <expr>|
               c[<idx>]*(<expr>) + c[<idx>]*(<expr>) + c[<idx>]*(<expr>)|
               <expr> + <expr>|
               <expr> + c[<idx>]*(<expr>)|
               c[<idx>]*(<expr>) + <expr>|
               c[<idx>]*(<expr>) + c[<idx>]*(<expr>)|
               <expr>|
               c[<idx>]*(<expr>)|

<expr>    ::=  An | Rem | An^c[<idx>] | Rem^c[<idx2>] |
               -An | -Rem | -An^c[<idx>] | -Rem^c[<idx2>] |
               Rem*An | An*An |
               -Rem*An | -An*An |
               (Rem^c[<idx2>])*An | (An^c[<idx>])*Rem | (Rem^c[<idx2>])*(An^c[<idx>]) |
               (-Rem^c[<idx2>])*An | (-An^c[<idx>])*Rem | (-Rem^c[<idx2>])*(An^c[<idx>]) |

<idx> ::= GE_constants:10
<idx2> ::= GE_constants:1
```

Figura 3.5: Gramática utilizada para el coeficiente de arrastre en la primera y segunda aproximación.

En la Figura 3.5 se observa que la regla de partida <*start*> indica si el coeficiente de arrastre tendrá 1, 2 o 3 sumandos y si esos sumandos están siendo multiplicados por una constante o no. Los sumandos son la regla <*expr*> y las constantes son c[<*idx*>]. Para este caso los sumandos pueden ser la variable An, la variable Rem o una multiplicación de ambas. Para cada una de las combinaciones posibles se tiene que los factores del sumando pueden estar elevados a una constante, An puede estar elevado a la constante c[<*idx*>] y Rem puede estar elevado a c[<*idx2*>]. Se dividieron las constantes en dos conjuntos, la regla <*idx*> y la regla <*idx2*> debido a la diferencia en los rangos de valores de An y Rem. Mientras que An es menor a 10, Rem puede llegar a 15000. Por lo que carece de sentido que Rem se encuentre elevado a un número grande.

La regla <*idx*> ::= GE_constants:N es equivalente a <*idx*> ::= N_0.0000 | N_0.0001 | N_0.0002 | ... | N_N.0000. Se utilizó esta notación con la palabra clave GE_constants para que las reglas puedan ser leídas y completadas en el inicio del programa. Un ejemplo de fenotipo generado con esta gramática es la siguiente ecuación:

$$\text{Rem*An + c[10\_7.3821]*(Rem\^{}c[1\_0.9638]).} \tag{3.7}$$

La notación c[N_X.XXXX] para las constantes se utiliza al momento de evaluar el fenotipo e indica que el valor de la constante es X.XXXX y que el límite superior de la constante es N. El punto en X.XXXX se utiliza para los decimales, por lo que se tiene una presición de 4 decimales. Esto implica que la constante <*idx*> está entre el rango [0.0000, 10.0000] y la constante <*idx2*> en el rango [0.0000, 1.0000]. Al evaluar el fenotipo se tienen 2 casos:

- **Evaluación sin optimización:** cuando no se desea optimizar el valor de la constante se reemplaza c[N_X.XXXX] por X.XXXX y se evalúa la ecuación obtenida con la métrica de desempeño.
- **Evaluación con optimización:** cuando se desea optimizar el valor de las constantes, se utiliza el algoritmo de optimización descrito al final de la Sección 3.1.4. En el caso de la constante c[N_X.XXXX] el valor inicial utilizado para la optimización es X.XXXX y las condiciones de borde para la constante están entre 0 y N. Luego se reemplaza c[N_X.XXXX] por Y.YYYY donde Y.YYYY es el valor para la constante entregado por el optimizador. En el ejemplo de la ecuación 3.7 se optimizarían las constantes buscando en la vecindad de 7.3821 y 0.9638; y los valores entregados por el optimizador estarían en los rangos [0, 10] y [0, 1] respectivamente.

3.1.6. Resolución del problema original

En esta sección se describe cómo se obtuvo Ind_FB. Una vez resueltas las dos aproximaciones anteriores, se procedió a correr el algoritmo con el conjunto de datos reales obtenidos de ANSYS. Estos datos fueron obtenidos en trabajos anteriores y en [5] se revisaron y se dividieron en subconjuntos de entrenamiento, validación y prueba; en este trabajo se mantuvieron estas divisiones para que los resultados puedan ser comparables. Al igual que en las aproximaciones, el trabajo se enfocó en un modelo de empaquetamiento de batería de 5 celdas. En este caso se desea ajustar el modelo fenomenológico a una base de datos obtenida con ANSYS que utiliza ecuaciones diferenciales parciales, por lo que no se tiene conocimiento sobre las ecuaciones de coeficiente de arrastre, coeficiente de fricción o número de Nusselt y tampoco se tiene conocimiento del mínimo error posible entre la salida teórica y la obtenida con el modelo ajustado de esta manera. Por lo anterior, se diseñó una gramática que abarque lo más posible pero no aumente drásticamente la complejidad de las soluciones. Las gramáticas utilizadas para cada parámetro se encuentran en el **Anexo A** y se diseñaron según lo descrito en la Sección 3.1.5.

Como se quiere que las curvas obtenidas con el modelo de Born tengan sentido físico y las expresiones sean de baja complejidad, la métrica de desempeño no es suficiente para elegir al mejor individuo. Por este motivo se guardaron varias soluciones de buen desempeño que tengan distintos fenotipos para analizar individualmente su complejidad y sentido físico. Para la complejidad se observó el tamaño de las expresiones y para el sentido físico se realizó el estudio que se describe en la Sección 3.2.

3.2. Estudio de interpretabilidad física de los modelos

Un punto importante para realizar un diseño óptimo de empaquetamiento, es poder predecir lo que ocurre al cambiar alguna variable de entrada. En esta sección se describe el método para validar si un modelo fenomenológico mantiene el sentido físico de la teoría. En [5] se acotó la base de datos según las variables de entrada:

- **Corriente que circula por las celdas (I):** puede tomar valores continuos entre 0 y 15 [A].

- **Separación entre las celdas (S):** puede tomar valores continuos entre 0.3 y 1.5 [% diámetro].
- **Diámetro de las celdas (D):** se utilizaron valores discretos de 18, 22, 26 y 32 [mm].
- **Flujo de aire a la entrada del pack (Fin):** puede tomar valores continuos entre 1 y 200 [CFM].
- **Temperatura del aire de entrada (Tin):** puede tomar valores continuos entre 10 y 30 [°C].

Para observar el comportamiento al modificar las variables de entrada de una a la vez, se definieron curvas según los rangos señalados anteriormente para todas las variables de entrada a excepción del diámetro de las celdas. Esto debido a que el diámetro es una variable que toma pocos valores discretos, por lo que el comportamiento se puede observar dentro de las demás curvas. En total son 15 curvas para cada variable de entrada que se estudia: 5 para la velocidad del fluido, 5 para la presión del fluido y 5 para la temperatura de celda.

1. **Curvas en función de I:** se hace variar la corriente entre 0 y 15 [A]. Se realizan 5 combinaciones con las demás variables. Para cada combinación mostrada a continuación se genera una curva de velocidad del fluido, una de presión del fluido y una de temperatura de celda.
    - S=0.48, Fin=30.85, Tin=13, Diam=22
    - S=1.32, Fin=30.85, Tin=13, Diam=22
    - S=1.32, Fin=170.15, Tin=13, Diam=22
    - S=1.32, Fin=170.15, Tin=27, Diam=22
    - S=1.32, Fin=170.15, Tin=27, Diam=26
2. **Curvas en función de S:** se hace variar la separación entre 0.3 y 1.5 [% diámetro]. Se realizan 5 combinaciones con las demás variables. Para cada combinación mostrada a continuación se genera una curva de velocidad del fluido, una de presión del fluido y una de temperatura de celda.
    - I=2.25, Fin=30.85, Tin=13, Diam=22
    - I=12.75, Fin=30.85, Tin=13, Diam=22
    - I=12.75, Fin=170.15, Tin=13, Diam=22
    - I=12.75, Fin=170.15, Tin=27, Diam=22
    - I=12.75, Fin=170.15, Tin=27, Diam=26
3. **Curvas en función de Fin:** se hace variar el flujo de aire entre 1 y 200 [CFM]. Se realizan 5 combinaciones con las demás variables. Para cada combinación mostrada a continuación se genera una curva de velocidad del fluido, una de presión del fluido y una de temperatura de celda.
    - I=2.25, S=0.48, Tin=13, Diam=22
    - I=12.75, S=0.48, Tin=13, Diam=22
    - I=12.75, S=1.32, Tin=13, Diam=22
    - I=12.75, S=1.32, Tin=27, Diam=22
    - I=12.75, S=1.32, Tin=27, Diam=26

4. **Curvas en función de Tin:** se hace variar la temperatura de entrada entre 10 y 30 [°C]. Se realizan 5 combinaciones con las demás variables. Para cada combinación mostrada a continuación se genera una curva de velocidad del fluido, una de presión del fluido y una de temperatura de celda.
    - I=2.25, S=0.48, Fin=30.85, Diam=22
    - I=12.75, S=0.48, Fin=30.85, Diam=22
    - I=12.75, S=1.32, Fin=30.85, Diam=22
    - I=12.75, S=1.32, Fin=170.15, Diam=22
    - I=12.75, S=1.32, Fin=170.15, Diam=26

Para poder validar si las curvas de un modelo siguen la teoría, se deben generar estas mismas con el programa ANSYS y compararlas. Para generar estas curvas en ANSYS se debe discretizar las variables independientes de cada curva. Debido a problemas con el *software*, no se alcanzaron a generar las curvas teóricas, sin embargo se propone realizar una discretización en 100 partes iguales. Y luego para cuantificar el desempeño según la interpretación física se utilizó el error cuadrático medio en cada curva.

Si bien no se dispone de las curvas teóricas para validar los modelos. Se realizó un análisis del sentido físico sobre el modelo de Born, el modelo de García-Tapia y el modelo de Aguilar. Esto con el objetivo de tener precedentes para cuando se generen las curvas teóricas. En la Sección 4.2 se observan las curvas obtenidas. Para las curvas con el modelo de Born y el modelo de García-Tapia se utilizaron las combinaciones descritas anteriormente en esta misma sección. Mientras que para las curvas con el modelo de Aguilar, se utilizaron menos combinaciones pero se graficaron para distintas columnas del modelo (debido a que este modelo es de 102 celdas y tiene las variables de salida velocidad, presión y temperatura para las 14 columnas).

Para las curvas del modelo de Aguilar se utilizaron las salidas de las columnas de celdas 2, 4, 6, 8, 10 y 12. Las combinaciones utilizadas fueron:

1. **Curvas en función de I:** se hace variar la corriente entre 0 y 15 [A]. Se realizan 2 combinaciones con las demás variables. Para cada combinación mostrada a continuación se generan 6 curvas de velocidad del fluido, 6 de presión del fluido y 6 de temperatura de celda. Esto último debido a las 6 columnas de celda donde se obtuvieron los valores.
    - S=0.48, Fin=30.85, Tin=13, Diam=22
    - S=1.32, Fin=170.15, Tin=27, Diam=26
2. **Curvas en función de S:** se hace variar la separación entre 0.3 y 1.5 [% diámetro]. Se realizan 2 combinaciones con las demás variables. Para cada combinación mostrada a continuación se generan 6 curvas de velocidad del fluido, 6 de presión del fluido y 6 de temperatura de celda. Esto último debido a las 6 columnas de celda donde se obtuvieron los valores.
    - I=2.25, Fin=30.85, Tin=13, Diam=22
    - I=12.75, Fin=170.15, Tin=27, Diam=26

3. **Curvas en función de Fin:** se hace variar el flujo de aire entrante entre 1 y 200 [CFM]. Se realizan 2 combinaciones con las demás variables. Para cada combinación mostrada a continuación se generan 6 curvas de velocidad del fluido, 6 de presión del fluido y 6 de temperatura de celda. Esto último debido a las 6 columnas de celda donde se obtuvieron los valores.
    - I=2.25, S=0.48, Tin=13, Diam=22
    - I=12.75, S=1.32, Tin=27, Diam=26
4. **Curvas en función de Tin:** se hace variar la temperatura del flujo de aire entrante entre 10 y 30 [°C]. Se realizan 2 combinaciones con las demás variables. Para cada combinación mostrada a continuación se generan 6 curvas de velocidad del fluido, 6 de presión del fluido y 6 de temperatura de celda. Esto último debido a las 6 columnas de celda donde se obtuvieron los valores.
    - I=2.25, S=0.48, Fin=30.85, Diam=22
    - I=12.75, S=1.32, Fin=170.15, Diam=26

Capítulo 4

Análisis de Resultados

A continuación se muestran los resultados obtenidos, ordenados según su relevancia. Junto a ellos se encuentra el análisis realizado y en el Capítulo 5 se encuentra la discusión de los resultados.

4.1. Individuos obtenidos con modelo de 5 celdas: Evolución Gramatical

Como se mencionó en la Sección 3.1.6, se utilizó evolución gramatical para ajustar el modelo fenomenológico de 5 celdas con los datos obtenidos con ANSYS. El individuo Ind_FB propuesto como solución al problema es:

$$\text{Cdrag}^{\text{FB}} = -\text{An} \cdot \text{An} + 0.0525 \cdot \text{An}^{4.1271} + 1.5874 \cdot \text{An}, \tag{4.1}$$

$$\text{Friction}^{\text{FB}} = 49.9984 \cdot \text{Rem}^{-0.427} \cdot \text{S}^{-2.5502}, \tag{4.2}$$

$$\text{Nu}^{\text{FB}} = 5.5163 \cdot \text{Rem}^{0.4549}. \tag{4.3}$$

Donde **An** es el área normalizada del manto de la celda que interactúa con el fluido, **Rem** es el número de Reynolds medio en el lugar de medición, **Dfn** es la densidad del fluido normalizada y **S** es la separación entre las celdas normalizada por el diámetro de celda. Estas expresiones son válidas para los siguientes rangos de las variables de entrada: I de 0 a 15 [A], S de 0.4 a 1.5 [% diámetro], D de 18 a 32 [mm], F_{in} de 1 a 200 [CFM] y T_{in} de 10 a 30 [°C].

El individuo mostrado en las ecuaciones anteriores tiene un error cuadrático medio de 1.93[m/s], 4.68[Pa] y 1.09[°C] en el conjunto de prueba para las salidas de velocidad, presión y temperatura

respectivamente. El mejor individuo en trabajos previos para esta distribución de celdas fue obtenido por García-Tapia (Ecuaciones 2.18, 2.19 y 2.20) que tiene un error cuadrático medio de 1.09[m/s], 6.13[Pa] y 0.58[°C] para las salidas respectivas. Según la métrica de desempeño definida en la Ecuación 3.6 se tiene un *fitness* de 0.0771 para Ind_FB y 0.0898 para Ind_GT; lo que indica que se obtuvo un individuo con un desempeño 16 % mejor en cuanto a la predicción general de las salidas.

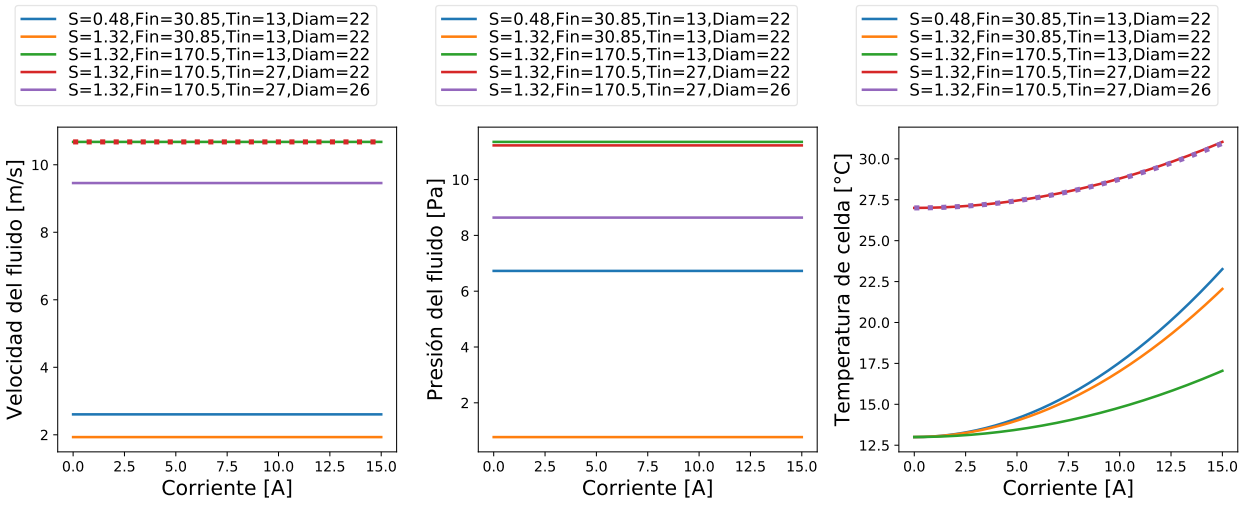
4.2. Estudio del comportamiento físico

Como se mencionó en la Sección 3.2, para observar el efecto de las variables de entrada y las de diseño se generaron diversas curvas modificando una variable a la vez y manteniendo las demás en distintas combinaciones. Para mostrar las curvas se utilizó el siguiente orden: para cada variable de entrada a la cual se le modificó su magnitud, se tienen 3 figuras:

- En la primera figura se tienen tres gráficas para el **modelo de Born**: el efecto de la variable de entrada sobre la velocidad del fluido, la presión del fluido y la temperatura de la celda respectivamente. Y en cada una de las gráficas se tienen las 5 combinaciones de las variables de entrada que se mantuvieron fijas.
- En la segunda figura se tienen tres gráficas para el **modelo de García-Tapia**: el efecto de la variable de entrada sobre la velocidad del fluido, la presión del fluido y la temperatura de la celda respectivamente. Y en cada una de las gráficas se tienen las 5 combinaciones de las variables de entrada que se mantuvieron fijas.
- En la tercera figura se tienen 6 gráficas para el **modelo de Aguilar** donde se hizo variar la variable de entrada. De arriba hacia abajo se tienen: 2 gráficas que muestran el comportamiento de la velocidad del fluido, 2 gráficas que muestran el comportamiento de la presión del fluido y 2 gráficas que muestran el comportamiento de la temperatura de celda. Ambas gráficas para cada variable de salida se diferencian en los valores de las magnitudes que se mantuvieron constantes. En cada una de las gráficas se tienen 6 curvas que hacen referencia a las columnas 2, 4, 6, 8, 10 y 12 respectivamente.

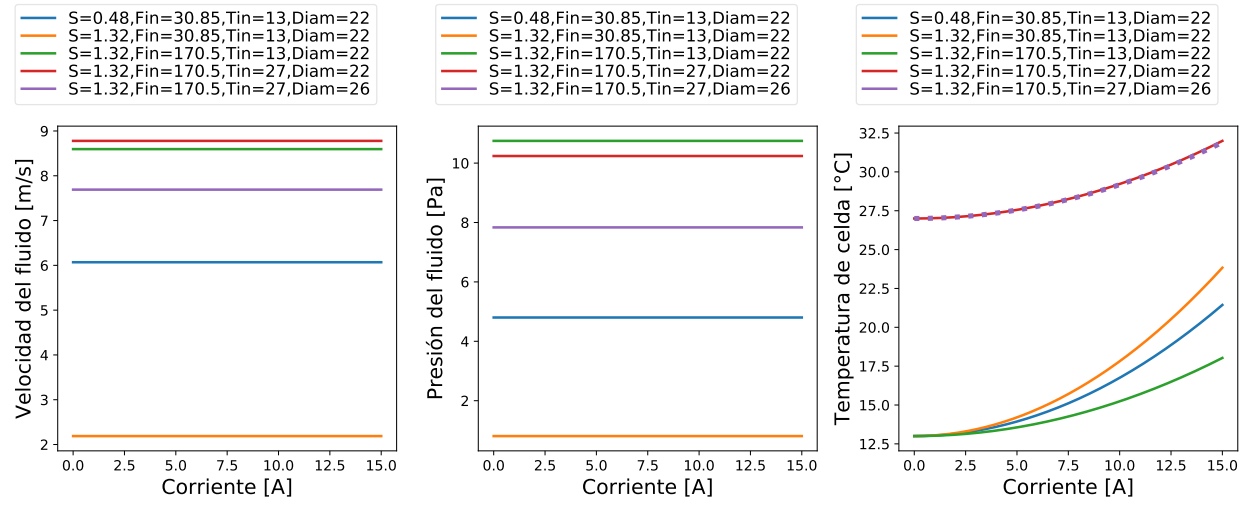
4.2.1. Efecto de la corriente

Se observa el efecto en la velocidad, presión y temperatura al modificar la corriente en el modelo de Born (Figura 4.1), en el modelo de García-Tapia (Figura 4.2) y en el modelo de Aguilar (Figura 4.3).



(a) **(b)** **(c)**

Figura 4.1: Efecto de la corriente en el modelo de Born. Se muestra el efecto sobre: (a) la velocidad del fluido, (b) la presión del fluido y (c) la temperatura de celda. En cada una de las gráficas se tienen 5 combinaciones de las demás variables de entrada que se mantuvieron fijas.



(a) **(b)** **(c)**

Figura 4.2: Efecto de la corriente en el modelo de García-Tapia. Se muestra el efecto sobre: (a) la velocidad del fluido, (b) la presión del fluido y (c) la temperatura de celda. En cada una de las gráficas se tienen 5 combinaciones de las demás variables de entrada que se mantuvieron fijas.

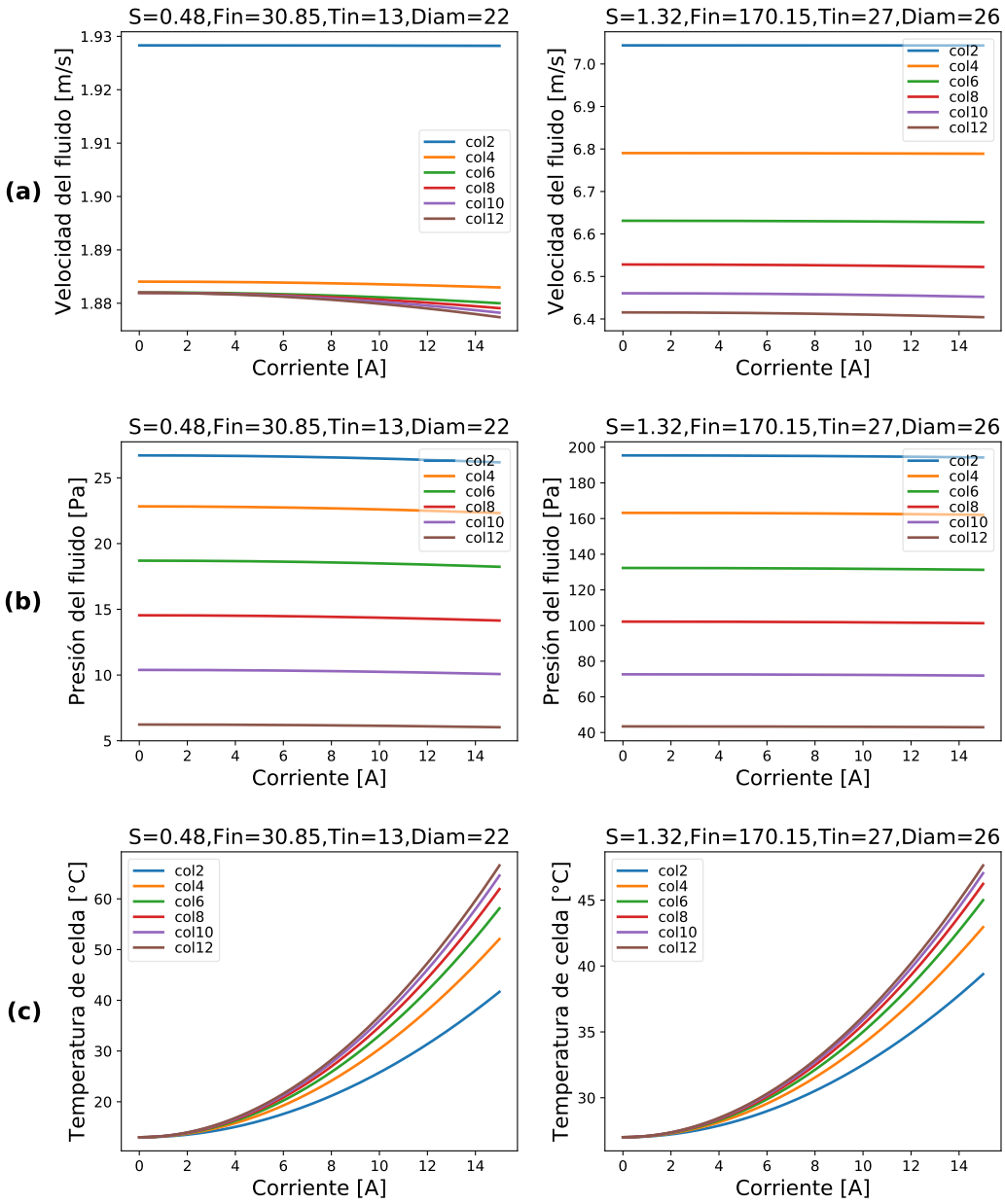


Figura 4.3: Efecto de la corriente en el modelo de Aguilar. Para dos combinaciones distintas de las variables de entrada, se muestra el efecto sobre: (a) la velocidad del fluido, (b) la presión del fluido y (c) la temperatura de celda. En cada una de las gráficas se tienen 6 curvas asociadas a las distintas columnas del modelo.

De las curvas mostradas se desprende que al modificar la corriente: la velocidad (Figuras 4.1 (a), 4.2 (a) y 4.3 (a)) y la presión del fluido (Figuras 4.1 (b), 4.2 (b) y 4.3 (b)) no varían sustancialmente. Si bien en la Figura 4.3 (a)-derecha hay una disminución de la velocidad al aumentar la corriente, el cambio es leve. Por el contrario, al aumentar la corriente, la temperatura de la celda aumenta drásticamente como se observa en las Figuras 4.1 (c), 4.2 (c) y 4.3 (c).

4.2.2. Efecto de la separación entre celdas

Se observa el efecto en la velocidad, presión y temperatura al modificar la separación entre celdas en el modelo de Born (Figura 4.4), en el modelo de García-Tapia (Figura 4.5) y en el modelo de Aguilar (Figura 4.6).

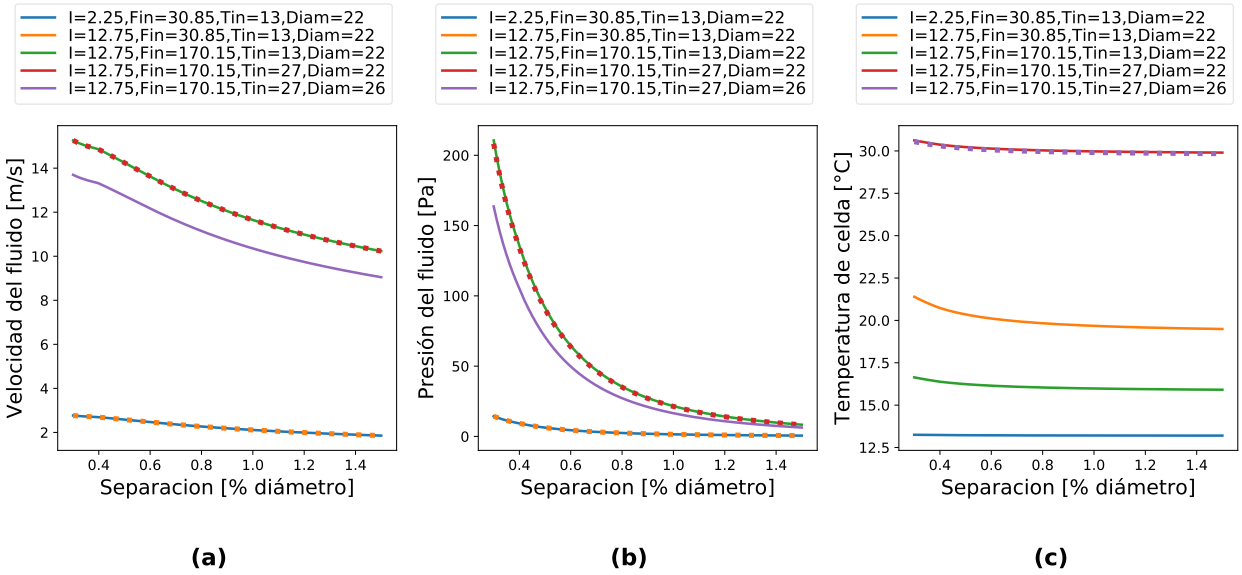


Figura 4.4: Efecto de la separación entre celdas en el modelo de Born. Se muestra el efecto sobre: (a) la velocidad del fluido, (b) la presión del fluido y (c) la temperatura de celda. En cada una de las gráficas se tienen 5 combinaciones de las demás variables de entrada que se mantuvieron fijas.

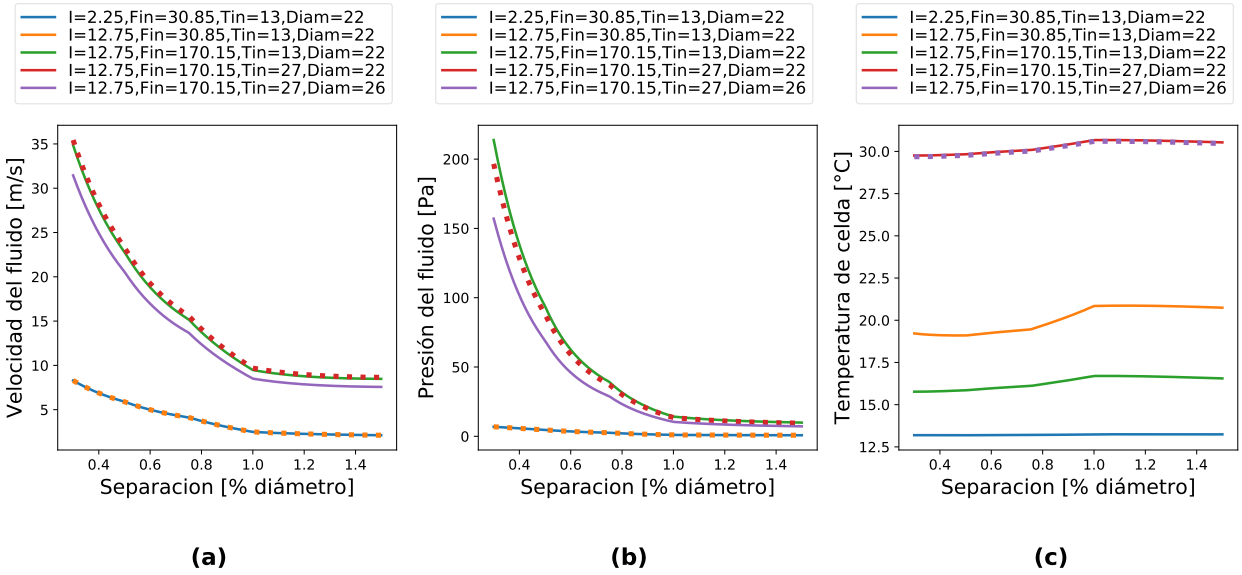


Figura 4.5: Efecto de la separación entre celdas en el modelo de García-Tapia. Se muestra el efecto sobre: (a) la velocidad del fluido, (b) la presión del fluido y (c) la temperatura de celda. En cada una de las gráficas se tienen 5 combinaciones de las demás variables de entrada que se mantuvieron fijas.

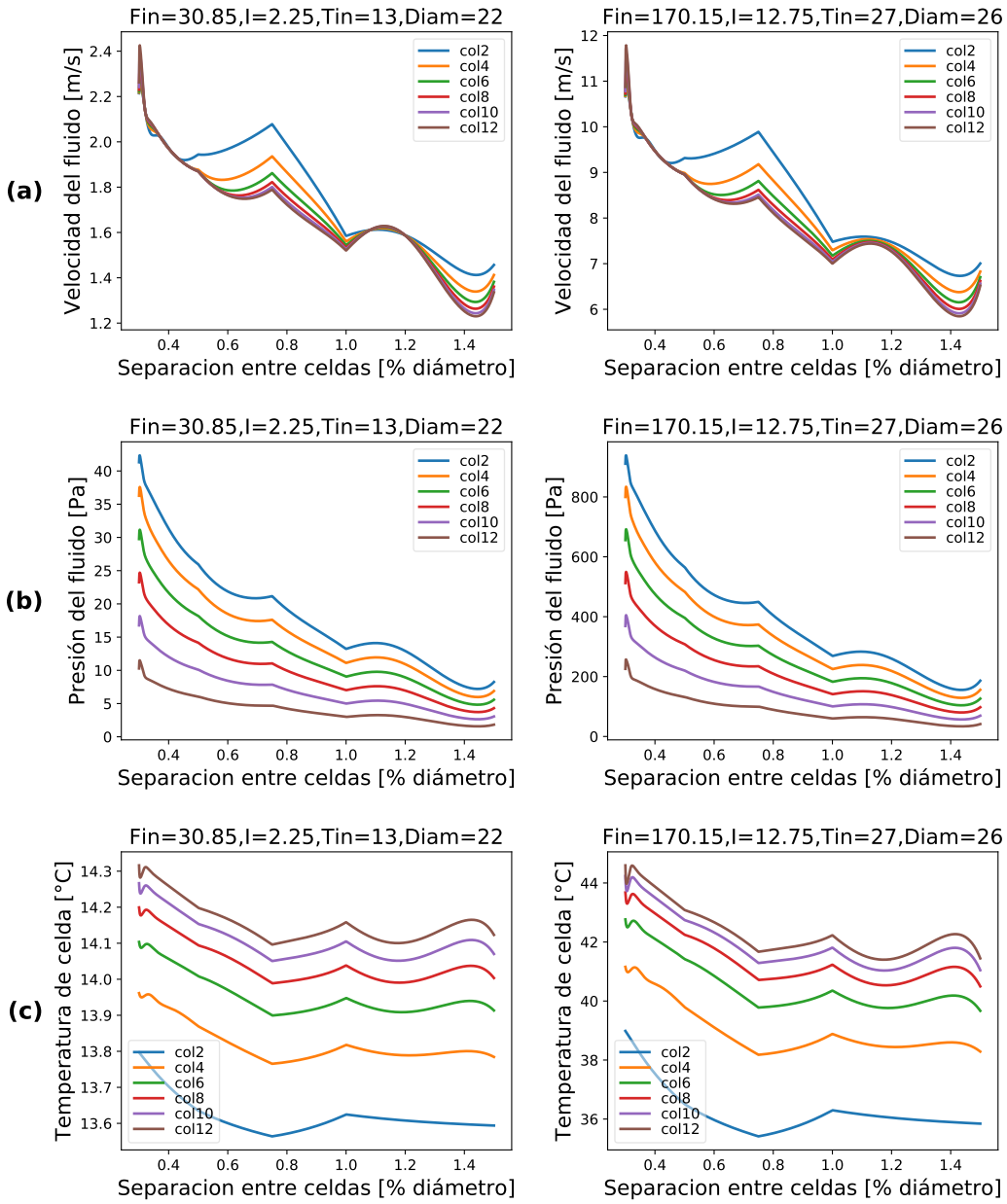


Figura 4.6: Efecto de la separación entre celdas en el modelo de Aguilar. Para dos combinaciones distintas de las variables de entrada, se muestra el efecto sobre: (a) la velocidad del fluido, (b) la presión del fluido y (c) la temperatura de celda. En cada una de las gráficas se tienen 6 curvas asociadas a las distintas columnas del modelo.

De las curvas mostradas se desprende que al aumentar la separación: la velocidad (Figuras 4.4 (a), 4.5 (a) y 4.6 (a)) y la presión del fluido (Figuras 4.4 (b), 4.5 (b) y 4.6 (b)) tienden a disminuir. Mientras que la temperatura no varía en gran medida como se muestra en las Figuras 4.4 (c), 4.5 (c) y 4.6 (c). Además del comportamiento general, en las Figuras 4.5 (c) y 4.6 se pueden observar máximos y mínimos locales de cada magnitud para valores específicos de separación lo que indica un posible punto de optimización para esta variable.

4.2.3. Efecto del flujo de aire entrante

Se observa el efecto en la velocidad, presión y temperatura al modificar la cantidad de flujo de aire entrante en el modelo de Born (Figura 4.7), en el modelo de García-Tapia (Figura 4.8) y en el modelo de Aguilar (Figura 4.9).

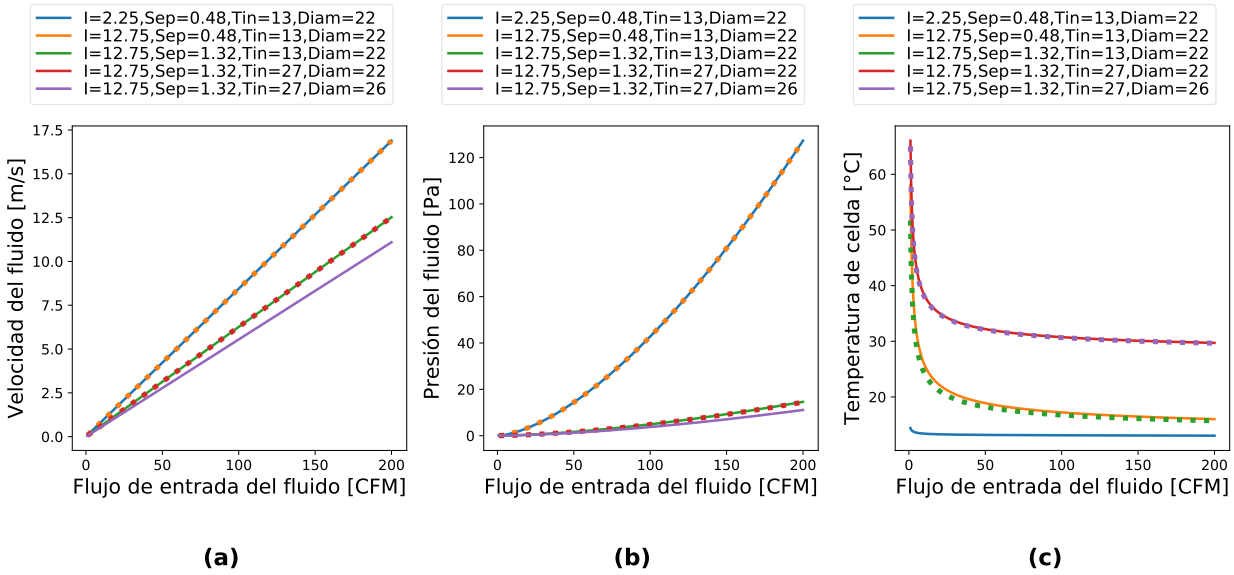


Figura 4.7: Efecto del flujo de aire entrante en el modelo de Born. Se muestra el efecto sobre: (a) la velocidad del fluido, (b) la presión del fluido y (c) la temperatura de celda. En cada una de las gráficas se tienen 5 combinaciones de las demás variables de entrada que se mantuvieron fijas.

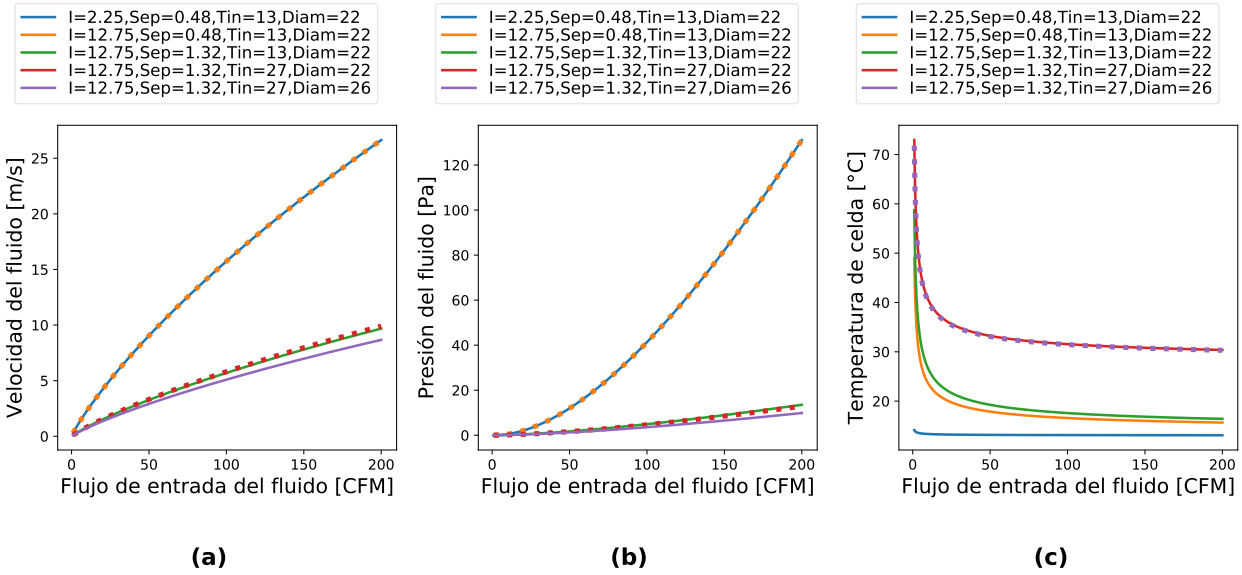


Figura 4.8: Efecto del flujo de aire entrante en el modelo de García-Tapia. Se muestra el efecto sobre: (a) la velocidad del fluido, (b) la presión del fluido y (c) la temperatura de celda. En cada una de las gráficas se tienen 5 combinaciones de las demás variables de entrada que se mantuvieron fijas.

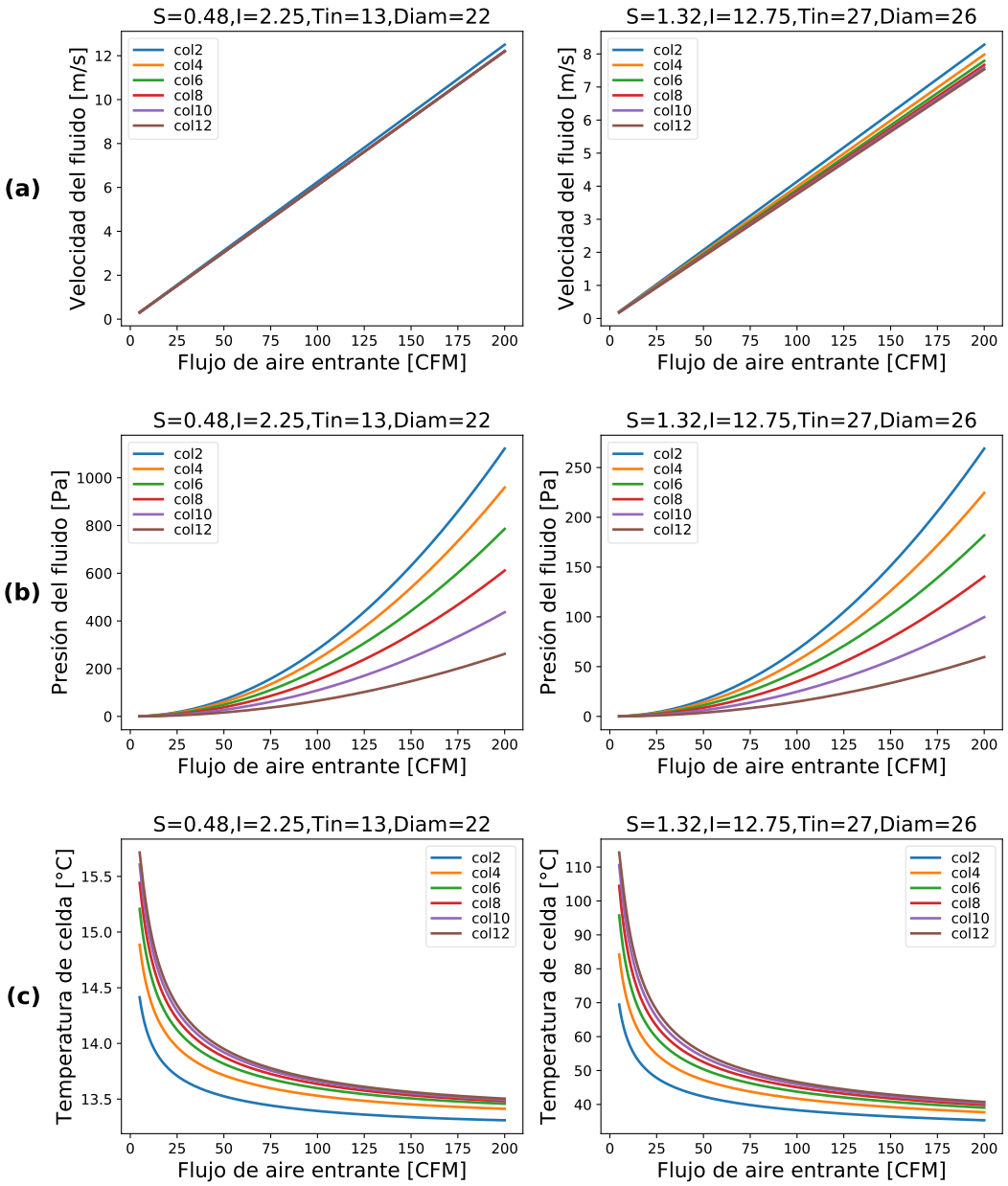


Figura 4.9: Efecto del flujo de aire entrante en el modelo de Aguilar. Para dos combinaciones distintas de las variables de entrada, se muestra el efecto sobre: (a) la velocidad del fluido, (b) la presión del fluido y (c) la temperatura de celda. En cada una de las gráficas se tienen 6 curvas asociadas a las distintas columnas del modelo.

De las curvas mostradas se desprende que al aumentar la cantidad del flujo de aire a la entrada aumenta drásticamente la velocidad (Figuras 4.7 (a), 4.8 (a) y 4.9 (a)) y presión del fluido (Figuras 4.7 (b), 4.8 (b) y 4.9 (b)). En las Figuras 4.7 (c) y 4.8 (c) se observa que la temperatura de celda disminuye de manera significativa hasta alrededor de los 25 [CFM] y luego no varía en gran medida. Mientras que en la Figura 4.9 (c), se observa que la temperatura de celda disminuye de manera significativa hasta alrededor de los 25 [CFM] para las celdas más cercanas al ventilador y alrededor de los 50 [CFM] para las celdas más lejanas.

4.2.4. Efecto de la temperatura del flujo de aire entrante

Se observa el efecto en la velocidad, presión y temperatura al modificar la temperatura del aire entrante en el modelo de Born (Figura 4.10), en el modelo de García-Tapia (Figura 4.11) y en el modelo de Aguilar (Figura 4.12).

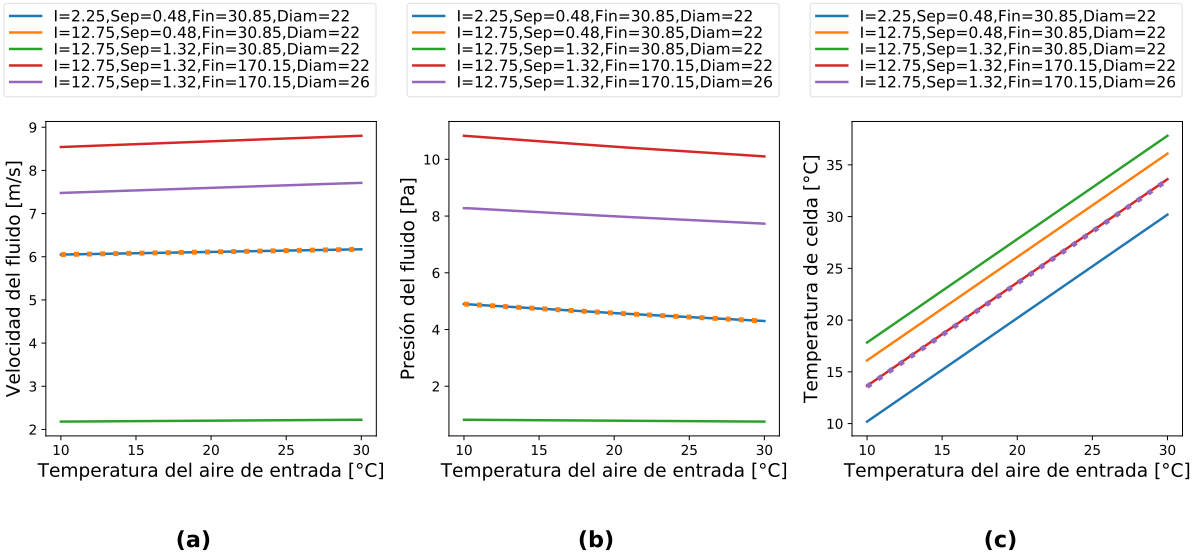


Figura 4.10: Efecto de la temperatura del aire entrante en el modelo de Born. Se muestra el efecto sobre: (a) la velocidad del fluido, (b) la presión del fluido y (c) la temperatura de celda. En cada una de las gráficas se tienen 5 combinaciones de las demás variables de entrada que se mantuvieron fijas.

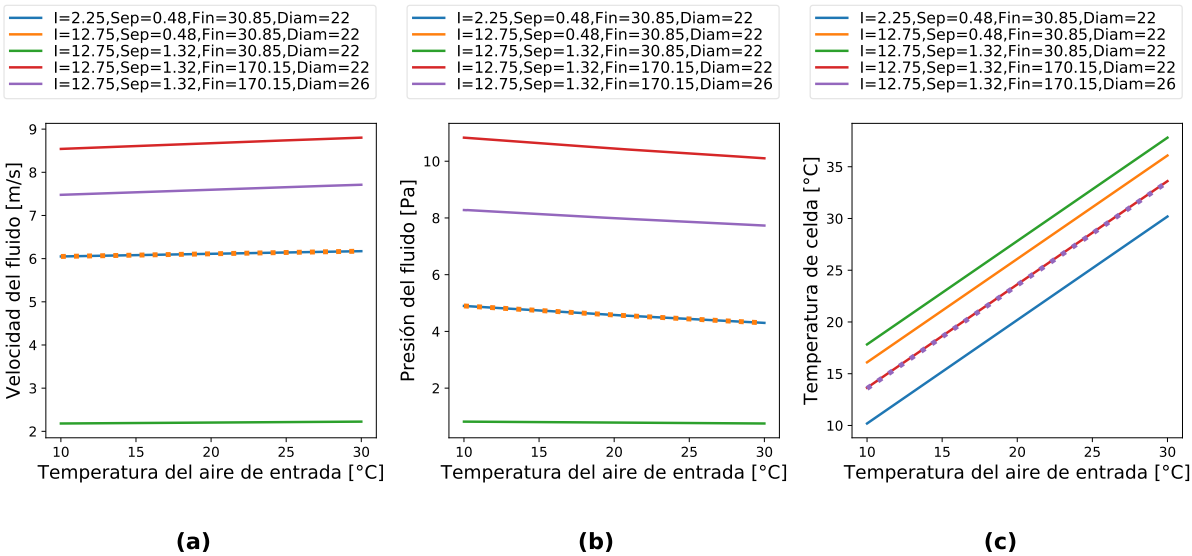


Figura 4.11: Efecto de la temperatura del aire entrante en el modelo de García-Tapia. Se muestra el efecto sobre: (a) la velocidad del fluido, (b) la presión del fluido y (c) la temperatura de celda. En cada una de las gráficas se tienen 5 combinaciones de las demás variables de entrada que se mantuvieron fijas.

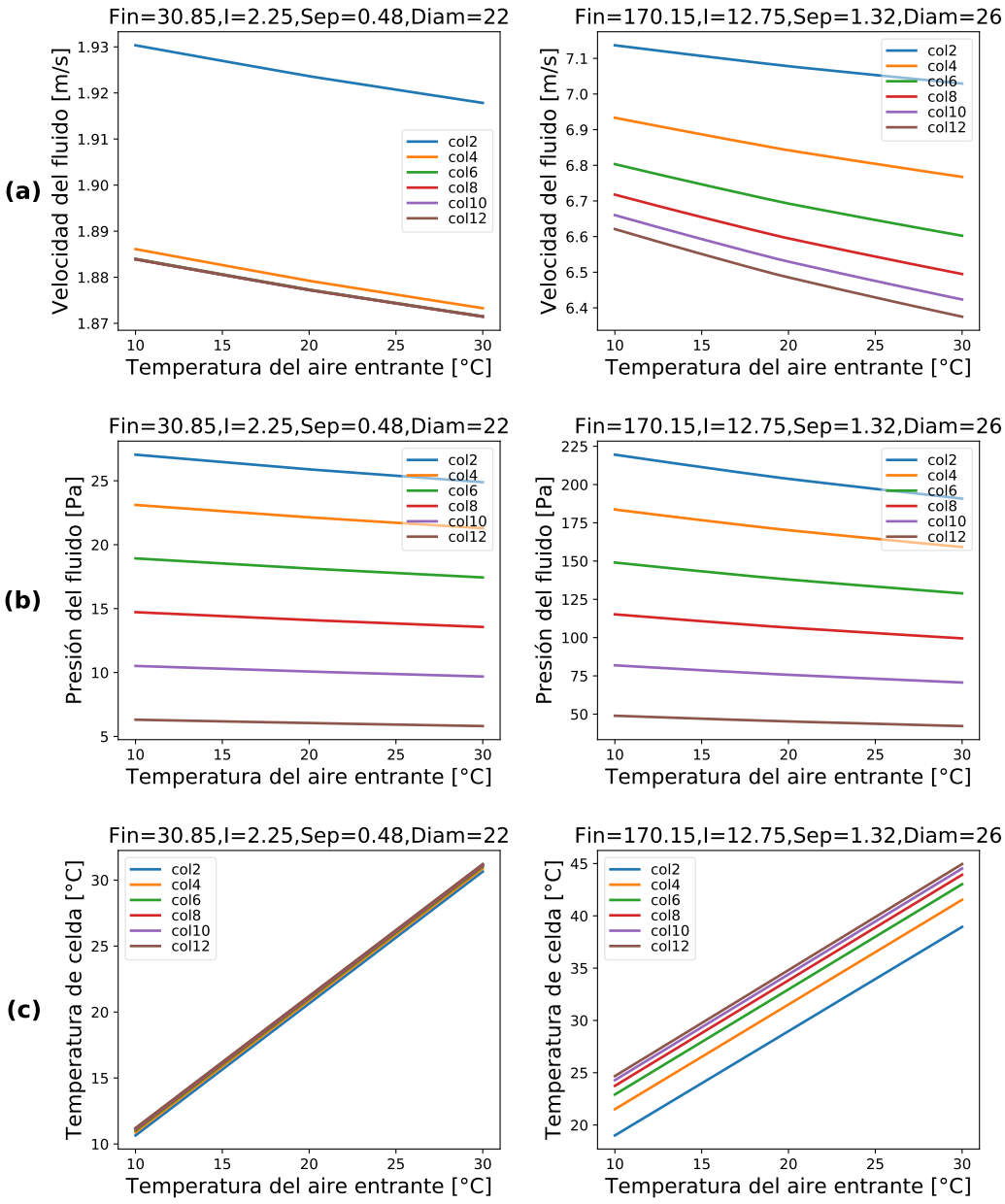


Figura 4.12: Efecto de la temperatura del aire entrante en el modelo de Aguilar. Para dos combinaciones distintas de las variables de entrada, se muestra el efecto sobre: (a) la velocidad del fluido, (b) la presión del fluido y (c) la temperatura de celda. En cada una de las gráficas se tienen 6 curvas asociadas a las distintas columnas del modelo.

De las curvas mostradas se desprende que al aumentar la temperatura del aire de entrada: la velocidad (Figuras 4.10 (a), 4.11 (a) y 4.12 (a)) y presión del fluido (Figuras 4.10 (b), 4.11 (b) y 4.12 (b)) no varían en gran medida. Por el contrario, al aumentar la temperatura del aire de entrada (Figuras 4.10 (c), 4.11 (c) y 4.12 (c)), la temperatura de la celda aumenta de forma considerable.

4.3. Resultados primera aproximación

Este resultado corresponde a la Sección 3.1.2. A partir de una base de datos generada con el individuo Ind_GT, el algoritmo propuesto debía encontrar Cdrag^{GT}. Suponiendo conocidos $\text{Friction}^{\text{GT}}$ y Nu^{GT}. La ecuación para el coeficiente de arrastre encontrada es:

$$\text{Coef arrastre encontrado} = 0.891 \cdot \text{An} - \text{An}^2 + \text{Rem}^{0.1452}, \tag{4.4}$$

la cuál es igual a Cdrag^{GT}. Por lo que se cumplió este objetivo de manera exacta.

4.4. Resultados segunda aproximación

Este resultado corresponde a la Sección 3.1.3. A partir de una base de datos generada con el individuo Ind_GT, el algoritmo debía encontrar Cdrag^{GT}, $\text{Friction}^{\text{GT}}$ y Nu^{GT}. Estas expresiones junto a las encontradas son:

$$\begin{aligned} \text{Cdrag}^{\text{GT}} &= 0.891 \cdot \text{An} - \text{An}^2 + \text{Rem}^{0.1452}, \\ \text{Coef arrastre encontrado} &= 0.8905 \cdot \text{An} - 0.9997 \cdot \text{An}^2 + \text{Rem}^{0.1452}, \end{aligned} \tag{4.5}$$

$$\begin{aligned} \text{Friction}^{\text{GT}} &= \text{Dfn} \cdot \text{S}^{(24.5261\text{Rem}^{-0.3891}-2)}, \\ \text{Coef fricción encontrado} &= \text{Dfn}^{(2.5175\cdot\text{Rem}^{-0.1006})} \cdot \text{S}^{(22.7098\cdot\text{Rem}^{-0.3774}-2.0218)}, \end{aligned} \tag{4.6}$$

$$\begin{aligned} \text{Nu}^{\text{GT}} &= 2.0232 \cdot \text{Rem}^{0.5528}, \\ \text{Número Nusselt encontrado} &= 1.9324 \cdot \text{Rem}^{0.5577} + 1. \end{aligned} \tag{4.7}$$

El error cuadrático medio entre las salidas teóricas y las obtenidas con las expresiones de la segunda aproximación es de 0.0011[m/s], 0.0096[Pa] y 0.0040[°C] para la velocidad del fluido, presión del fluido y temperatura de celda respectivamente. Teniendo en cuenta que las salidas teóricas tienen una media de 8.7809[m/s], 18.1097[Pa] y 21.6843[°C] se tiene un muy buen desempeño. Para poder analizar de mejora manera el comportamiento, en la Figura 4.13 se comparan las curvas de las salidas teóricas y las obtenidas para valores típicos de las variables {An, Rem, Dfn, S}. En esta figura se puede ver cómo las 3 expresiones encontradas generan curvas casi iguales a las de Ind_GT.

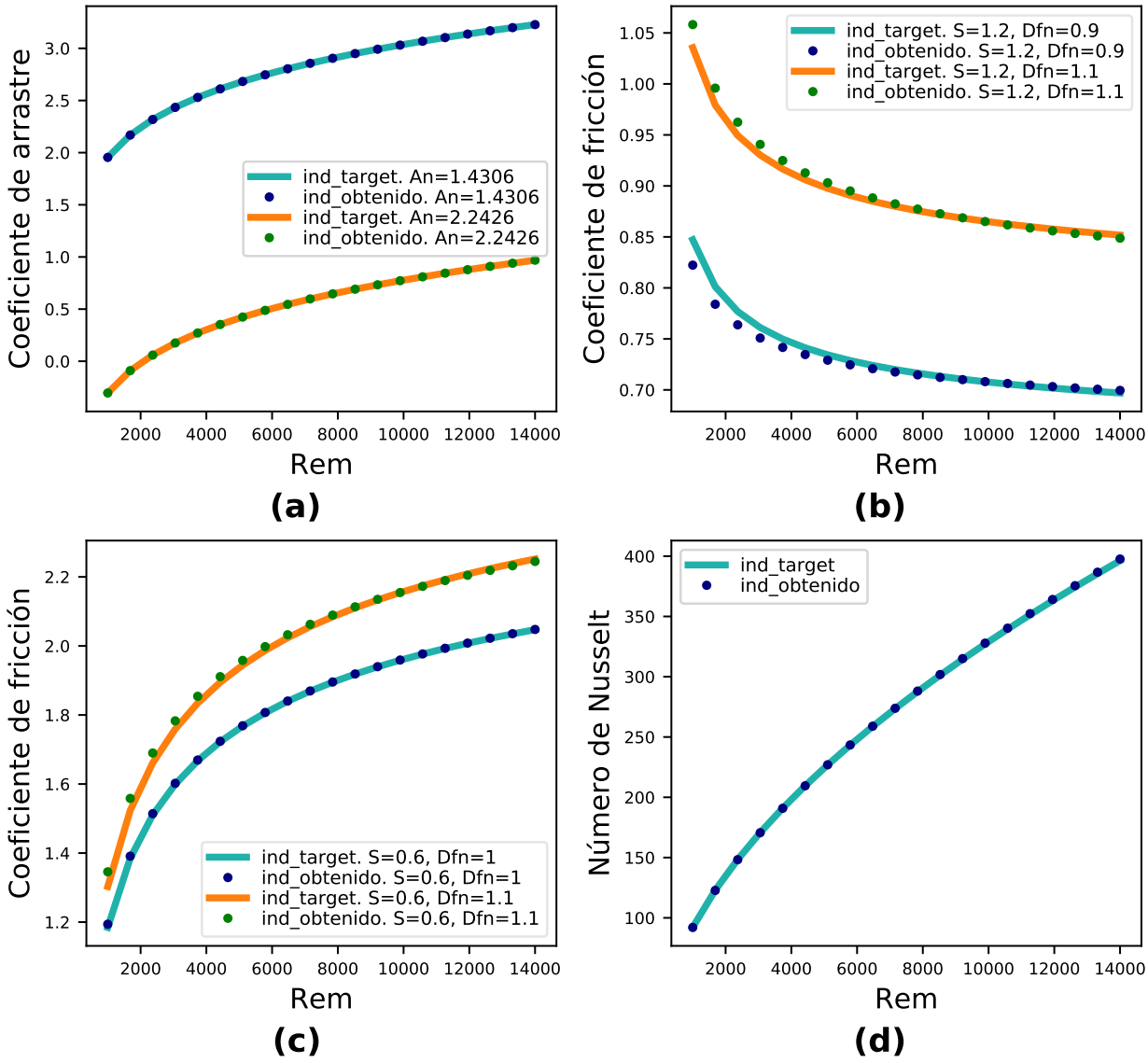


Figura 4.13: Comparación entre las curvas teóricas (ind_target) y las obtenidas (ind_obtenido). Las curvas se muestran en función de Rem y se hicieron combinaciones con las demás variables para distintos valores típicos. Se muestran dos pares de combinaciones para: (a) el coeficiente de arrastre modificando An, (b) el coeficiente de fricción modificando Dfn y manteniendo S=1.2, (c) el coeficiente de fricción modificando Dfn y manteniendo S=0.6. En (d) se muestra el número de Nusselt en función de Rem.

Capítulo 5

Discusión de los resultados

Si bien en la primera y segunda aproximación se esperaba encontrar las ecuaciones exactas de Ind_GT para poder continuar con el problema original. Esto sólo se logró en la primera aproximación, mientras que en la segunda hay resultados muy buenos pero no exactos. Este punto podría asociarse al problema que tiene evolución gramatical denominado **ripple effect**, el cuál es el efecto que ocurre al realizar los operadores de mutación o *crossover* directamente en el genotipo lineal de un individuo. Debido al modo de mapeo entre genotipo y fenotipo, un segmento del cromosoma que representa un subárbol de la expresión padre puede ser mapeado a un subárbol totalmente distinto en el hijo. Este efecto conlleva problemas para encontrar expresiones exactas como en la segunda aproximación.

5.1. Ejemplo de uso para el diseño de un empaquetamiento de baterías

A modo de ejemplificar el uso de los resultados obtenidos y según los resultados mostrados en la Sección 4.2, si se optimiza el diseño de un empaquetamiento de baterías utilizando los modelos de García-Tapia y de Aguilar se pueden seguir los siguientes pasos. Teniendo en cuenta que para realizar un empaquetamiento de baterías se debe tener clara la aplicación, se asume conocida la corriente máxima que se necesita y el rango de temperaturas al cuál trabajan las celdas para que su vida útil no se deteriore en exceso.

Con esto en mente, al realizar el diseño de empaquetamiento con los modelos estudiados se tiene que:

- Según la Sección 4.2.1, se debe considerar siempre la corriente máxima que se necesita como un parámetro. De esta forma se tiene la temperatura máxima que alcanzarán las celdas y se

puede mantener bajo su límite.

- Según la Sección 4.2.3, un flujo mayor a 50 [CFM] sólo aumenta la velocidad y presión del fluido pero no mejora la eficacia de la ventilación de manera significativa. Por lo que puede ser un costo innecesario.
- Según la Sección 4.2.4, la temperatura del flujo de aire entrante ayuda en la regulación de la temperatura de las celdas. Se puede asociar la temperatura mínima de entrada que es capaz de entregar el ventilador con la corriente máxima y la temperatura máxima del aire con la corriente mínima. Así junto con el volumen de flujo de aire entrante se tiene un máximo de la temperatura que se puede regular.
- La separación entre celdas es relevante ya que su comportamiento no se observa monótono. Como se mencionó anteriormente, la temperatura afecta la vida útil de las celdas y por este motivo es la salida más importante a analizar. De la Figura 4.6 (c) se observa que hay aproximadamente 3 mínimos con valores similares para las separaciones de 0.75, 1.15 y 1.5 [% diámetro] respectivamente. Si bien los mínimos son parecidos, se sugiere utilizar una separación de 0.75 [% diámetro] para disminuir el tamaño utilizado por el empaquetamiento.

El efecto de la separación de celdas en la temperatura de la celda central involucra la comprensión de complejos fenómenos propios de Mecánica de Fluidos y Transferencia de Calor. Para interpretar los resultados, se solicitó la ayuda del Profesor del departamento de Mecánica de la Universidad de Chile, Williams Calderón. En la Figura 5.1 (a) se muestra la temperatura en función de la separación entre celdas para el modelo de García-Tapia. Luego de ver esta figura, el Profesor indicó que la disminución en la separación entre celdas puede provocar variados fenómenos, que podrían favorecer el aumento o disminución de la temperatura. Entre los fenómenos mencionados se encuentra el aumento de la velocidad del fluido por efecto *jet* que produce disminución de la temperatura y la interferencia en la refrigeración debido a las celdas aguas arriba que haría aumentar la temperatura. Por lo que mencionó que para discernir si debiese aumentar o disminuir la temperatura, sería necesario entender con mayor profundidad la fluidodinámica alrededor de la celda central.

Como la opinión anterior fue basada en lo observado en las curvas generadas con el modelo de García-Tapia, los cambios bruscos de la curva de la Figura 5.1 (a) pueden deberse a alguno de los siguientes casos:

1. Para ciertos valores de las variables de entrada en la base de datos el fluido cambia entre un estado laminar y uno turbulento. Por lo que los modelos se intentan ajustar a ello.
2. Es un artefacto del modelo de García-Tapia.
3. El modelo de Reyes tiene este comportamiento intrínsecamente, sin dependencia de los parámetros que están siendo ajustados con algoritmos evolutivos.
4. Ocurren fenómenos que aumentan la temperatura y otros que la disminuyen, tal como mencionó el Profesor Williams.

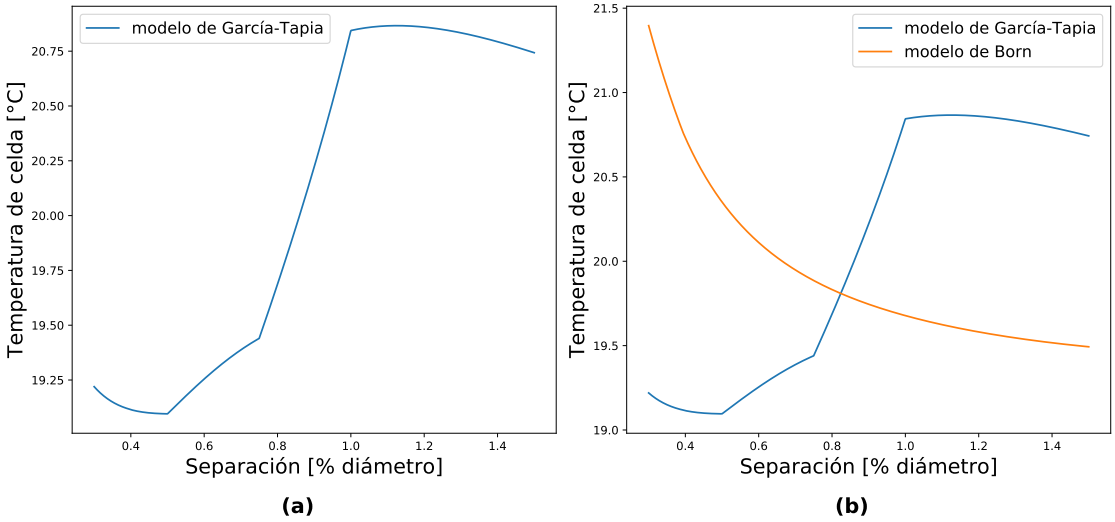


Figura 5.1: (a) Efecto de la separación entre celdas para la temperatura de las celdas en el modelo de García-Tapia. (b) Comparación de las curvas de temperatura de celdas en función de la separación de celdas para los modelos de García-Tapia y de Born.

Para verificar que es lo que ocurre, se propone replicar esta curva con el *software* ANSYS. Más allá de los resultados de temperatura que entrega el *software*, este permite visualizar lo que ocurre en el momento de la simulación, por lo que se debe prestar atención si hay cambios en el comportamiento del fluido. Con esto se puede comprobar si el comportamiento no monótono de las curvas es debido al punto 1 señalado anteriormente. En las curvas de la temperatura en función de la separación del modelo de Aguilar (Figura 4.6 (c)) también se encuentran cambios bruscos, esto hace improbable el punto 2. Para saber si el comportamiento es debido al punto 3, se revisó el código de manera exhaustiva. En este caso se encontraron los parámetros de optimización a1, a2, a3 utilizados por Reyes que no han sido modificados en el modelo de García-Tapia ni en el modelo de Aguilar. Se tiene que a1 y a2 son funciones escalonadas respecto a la separación entre celdas y en particular los escalones ocurren para S=0.5, S=0.75 y S=1. Esto último podría ser la causa de los resultados no monótonos. El modelo de Born se generó sin utilizar los parámetros a1 y a2 y obtuvo resultados similares al modelo de García-Tapia respecto a los errores cuadráticos medios para las salidas. En la Figura 5.1 (b), se graficó la curva utilizando el modelo de Born sobre la curva generada con el modelo de García-Tapia. En ambas curvas se tienen rangos para la temperatura similares lo que tiene sentido con las métricas obtenidas. Sin embargo, el comportamiento es totalmente opuesto y generan dos opciones de optimización de empaquetamiento muy diferente: según el modelo de García-Tapia, la temperatura es mínima para S=0.5 [% diámetro]; mientras que según el modelo de Born, a mayor separación se disminuye la temperatura. Para conocer si alguno de esos comportamientos es el correcto, es necesario obtener las curvas teóricas con ANSYS. Finalmente, el punto 4 es intrínseco al modelo de baterías con el que se hizo el modelo fenomenológico. Por lo que si el comportamiento es debido a esto, se tendría un punto de optimización importante en el momento de diseñar un empaquetamiento con esta distribución.

Capítulo 6

Conclusiones

Si bien se restringió el análisis a un modelo de 5 celdas, se realizaron grandes contribuciones a la resolución del problema. Se definió un criterio para validar la interpretación física de los modelos a través de curvas que pueden ser generadas para cualquier número de celdas. Por otro lado se demostró el potencial del algoritmo de evolución gramatical sobre programación genética en este tipo de problemas donde las expresiones que se quieren obtener tienen un sentido físico. A través de la gramática se puede restringir la evolución de fórmulas con el sentido físico que se quiera. Se puede diseñar la gramática para que el número de Nusselt sea creciente con el número de Reynolds para ciertos rangos, tal como indica la teoría. De la misma manera, se puede mantener la simplicidad matemática para no evolucionar expresiones complejas como suele suceder en los algoritmos evolutivos.

De los estudios realizados, se puede diferenciar evolución gramatical de programación genética al encontrar expresiones de la siguiente forma:

- Se puede guiar las expresiones que evolucionan a través de una gramática en evolución gramatical, mientras que en programación genética se deben hacer restricciones que pueden afectar el proceso evolutivo. Esto permite que en evolución gramatical se pueda mantener un sentido físico correcto para el 100 % de los individuos. Y también se puede controlar la complejidad de las expresiones.
- Los operadores de *crossover* y mutación en programación genética son menos destructivos que en evolución gramatical debido al ripple effect. Esto implica que se heredan de mejor manera las características de los individuos padres en programación genética.
- Programación genética tiene una implementación más sencilla en términos del mapeo de los individuos. Donde únicamente se escogen los nodos terminales y nodos no terminales. Mientras que evolución gramatical depende mucho de la gramática elegida.

6.1. Propuesta de Trabajos Futuro

Al utilizar algoritmos evolutivos se sabe que el costo computacional puede ser extremadamente alto dependiendo del costo de calcular el desempeño de un individuo. En este caso, evaluar un individuo incluso en un modelo de 5 celdas requiere un tiempo relativamente alto, es por esto que la primera propuesta es optimizar el algoritmo implementado en esta memoria para que permita realizar pruebas con un mayor número de individuos y generaciones. Permitiendo mejores resultados en el mismo tiempo. Para ello se propone modificar la función de optimización de constantes por una específica para el algoritmo de evolución gramatical implementado. Se necesita una función que mejore las constantes con un número bajo de iteraciones, sin importar si es el mínimo local exacto. Esto último debido a que si se utiliza una función de optimización de constantes muy costosa computacionalmente, no queda tiempo para un proceso evolutivo prolongado.

Un punto relevante a tener en cuenta es que se desea tener un modelo que funcione para distinto número de celdas. Luego de lograr un algoritmo funcional que pueda evolucionar expresiones que tengan sentido físico, el siguiente paso es encontrar las expresiones que permitan interpolar y extrapolar el número de celdas. Si bien en la memoria de Aguilar se agregaron terminales referentes al índice de la columna de celda y al número total de columnas, al evolucionar el programa se utilizó de *fitness* el desempeño en ajustar las curvas únicamente del modelo de 102 celdas y una vez obtenido eso, se probó su desempeño en el modelo de 53 y 151 celdas. Para obtener un modelo más general se propone utilizar de *fitness* el desempeño tanto en el modelo de 53, 102 y 151 celdas. Luego para el estudio de extrapolación e interpolación se propone generar con ANSYS una base de datos para 32, 74, 130 y 179 celdas. Sin utilizar estas últimas bases durante el entrenamiento. También se propone agregar el terminal *col_idx/total_columns*, donde *col_idx* es el índice de la columna de celda y *total_columns* es el número total de columnas que tiene el modelo.

Otro punto relevante hace referencia a las curvas generadas en función de la separación entre celdas. Debido a que existe la posibilidad de que el fluido pase a un estado turbulento para algunas de las configuraciones que se realizaron, es necesario revisar en el programa ANSYS un hiperespacio para las variables de entrada donde el fluido se asegure en estado laminar. Esto es de suma importancia, porque si es que hubiesen casos donde el fluido tiene otro comportamiento, perdería el sentido de encontrar una sola expresión para cada coeficiente.

Por último se propone el análisis de los efectos de entrada directamente en ANSYS para verificar lo obtenido de las curvas de trabajos previos, esto se puede realizar generando las curvas descritas en 3.2 y asegurarse del estado del fluido como se mencionó anteriormente.

Bibliografía

[1] Gert Berckmans, Maarten Messagie, Jelle Smekens, Noshin Omar, Lieselot Vanhaverbeke, and Joeri Van Mierlo. Cost projection of state of the art lithium-ion batteries for electric vehicles up to 2030. *Energies*, 10(9):1314, 2017.

[2] Jorge Reyes. Reporte del Modelo paramétrico. Technical report, Universidad de Chile, 2014.

[3] Francisco Villa. Ajuste de Modelo Fenomenológico de Celdas de Batería usando Algoritmos Evolutivos. Memoria de Título, DIE, Universidad de Chile, 2015.

[4] I Yon. Informe proyecto: Estudio de parámetros y utilización de software de programación genética gpalta. Technical report, 2016.

[5] German García and Nicolás Tapia. Evolución del modelo fenomenológico de bancos de baterías. Technical report, 2018.

[6] D Gómez and A Larrañaga. Ajuste de modelo fenomenológico de celdas de batería usando algoritmos evolutivos. Technical report, 2017.

[7] Nicolás Aguilar. Optimización Multiobjetivo de un Modelo Fenomenológico para el Empaquetamiento de Baterías de Litio Mediante Programación Genética. Memoria de Título, DIE, Universidad de Chile, 2019.

[8] Jorge R Vergara. Multi-objective optimization of a lithium battery packaging using genetic programming. Presentación PPT, Universidad Tecnológica Metropolitana, 2019.

[9] Pradnya A Vikhar. Evolutionary algorithms: A critical review and its future prospects. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pages 261–265. IEEE, 2016.

[10] Xinjie Yu and Mitsuo Gen. *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010.

[11] Candida Ferreira. Gene expression programming: a new adaptive algorithm for solving problems. *Complex Systems*, 13:87–129, 2001.

[12] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[13] Michael O'Neill and Conor Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358, 2001.

[14] Sara Silva. GPLab: toolbox de programación genética para MATLAB, 2014.

[15] S Luke. The ECJ Owner's Manual, 2015.

[16] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[17] Farzad Noorian, Anthony Mihirana de Silva, Philip HW Leong, et al. gramevol: Grammatical evolution in r. *Journal of Statistical Software*, 71(1):1–26, 2016.

[18] Michael Fenton, James McDermott, David Fagan, Stefan Forstenlechner, Erik Hemberg, and Michael O'Neill. Ponyge2: Grammatical evolution in python, 2017.

Anexo A
Gramáticas utilizadas

A.1. Gramáticas utilizadas en la primera y segunda aproximación

Para la primera aproximación de la Sección 3.1.2 y la segunda aproximación de la Sección 3.1.3, se utilizaron las siguientes gramáticas. Para el coeficiente de arrastre se utilizó la gramática de la Figura 6.1. Para el coeficiente de fricción se utilizó la gramática que se dividió en dos partes debido al espacio, la primera de la Figura 6.4 y la segunda de la Figura 6.3. Para el número de Nusselt se utilizó la gramática de la Figura 6.2.

```
<start> ::=    <expr> + <expr> + <expr>|
               <expr> + <expr> + c[<idx>]*(<expr>)|
               <expr> + c[<idx>]*(<expr>) + <expr>|
               <expr> + c[<idx>]*(<expr>) + c[<idx>]*(<expr>)|
               c[<idx>]*(<expr>) + <expr> + <expr>|
               c[<idx>]*(<expr>) + <expr> + c[<idx>]*(<expr>)|
               c[<idx>]*(<expr>) + c[<idx>]*(<expr>) + <expr>|
               c[<idx>]*(<expr>) + c[<idx>]*(<expr>) + c[<idx>]*(<expr>)|
               <expr> + <expr>|
               <expr> + c[<idx>]*(<expr>)|
               c[<idx>]*(<expr>) + <expr>|
               c[<idx>]*(<expr>) + c[<idx>]*(<expr>)|
               <expr>|
               c[<idx>]*(<expr>)|

<expr>    ::=  An | Rem | An^c[<idx>] | Rem^c[<idx2>] |
               -An | -Rem | -An^c[<idx>] | -Rem^c[<idx2>] |
               Rem*An | An*An |
               -Rem*An | -An*An |
               (Rem^c[<idx2>])*An | (An^c[<idx>])*Rem | (Rem^c[<idx2>])*(An^c[<idx>]) |
               (-Rem^c[<idx2>])*An | (-An^c[<idx>])*Rem | (-Rem^c[<idx2>])*(An^c[<idx>]) |

<idx> ::= GE_constants:10
<idx2> ::= GE_constants:1
```

Figura 6.1: Gramática para el coeficiente de arrastre en la primera y segunda aproximación.

```
<start>   ::=  Rem^c[<idx2>] + Rem^c[<idx2>]|
               Rem^c[<idx2>] + c[<idx>]*(Rem^c[<idx2>])|
               c[<idx>]*(Rem^c[<idx2>]) + Rem^c[<idx2>]|
               c[<idx>]*(Rem^c[<idx2>]) + c[<idx>]*(Rem^c[<idx2>])|
               Rem^c[<idx2>]|
               c[<idx>]*(Rem^c[<idx2>])|

<idx> ::= GE_constants:10
<idx2> ::= GE_constants:1
```

Figura 6.2: Gramática para el número de Nusselt en la primera y segunda aproximación.

```
<start> ::=    <expr> + <expr> + <expr>|
               <expr> + <expr> + c[<idx>]*(<expr>)|
               <expr> + c[<idx>]*(<expr>) + <expr>|
               <expr> + c[<idx>]*(<expr>) + c[<idx>]*(<expr>)|
               c[<idx>]*(<expr>) + <expr> + <expr>|
               c[<idx>]*(<expr>) + <expr> + c[<idx>]*(<expr>)|
               c[<idx>]*(<expr>) + c[<idx>]*(<expr>) + <expr>|
               c[<idx>]*(<expr>) + c[<idx>]*(<expr>) + c[<idx>]*(<expr>)|
               <expr> + <expr>|
               <expr> + c[<idx>]*(<expr>)|
               c[<idx>]*(<expr>) + <expr>|
               c[<idx>]*(<expr>) + c[<idx>]*(<expr>)|
               <expr>|
               c[<idx>]*(<expr>)|

<expr>  ::=    Rem | Dfn | S |
               Rem^(<e_r>) | Dfn^(<e_d>) | S^(<e_s>) |
               Rem*S | S*Dfn | Rm*Dfn |
               Rem*(S^(<e_s>)) | S*(Dfn^(<e_d>)) | Rm*(Dfn^(<e_d>)) |
               (Rem^(<e_r>))*S | (S^(<e_s>))*Dfn | (Rem^(<e_r>))*Dfn |
               (Rem^(<e_r>))*(S^(<e_s>)) | (S^(<e_s>))*(Dfn^(<e_d>)) | (Rem^(<e_r>))*(Dfn^(<e_d>)) |
               Dfn*S*Rem |
               Dfn*S*(Rem^(<e_r>)) |
               Dfn*(S^(<e_s>))*Rem |
               Dfn*(S^(<e_s>))*(Rem^(<e_r>)) |
               (Dfn^(<e_d>))*S*Rem |
               (Dfn^(<e_d>))*S*(Rem^(<e_r>)) |
               (Dfn^(<e_d>))*(S^(<e_s>))*Rem |
               (Dfn^(<e_d>))*(S^(<e_s>))*(Rem^(<e_r>)) |
               -Rem | -Dfn | -S |
               -Rem^(<e_r>) | -Dfn^(<e_d>) | -S^(<e_s>) |
               -Rem*S | -S*Dfn | -Rm*Dfn |
               -Rem*(S^(<e_s>)) | -S*(Dfn^(<e_d>)) | -Rm*(Dfn^(<e_d>)) |
               -(Rem^(<e_r>))*S | -(S^(<e_s>))*Dfn | -(Rem^(<e_r>))*Dfn |
               -(Rem^(<e_r>))*(S^(<e_s>)) | -(S^(<e_s>))*(Dfn^(<e_d>)) | -(Rem^(<e_r>))*(Dfn^(<e_d>)) |
               -Dfn*S*Rem |
               -Dfn*S*(Rem^(<e_r>)) |
               -Dfn*(S^(<e_s>))*Rem |
               -Dfn*(S^(<e_s>))*(Rem^(<e_r>)) |
               -(Dfn^(<e_d>))*S*Rem |
               -(Dfn^(<e_d>))*S*(Rem^(<e_r>)) |
               -(Dfn^(<e_d>))*(S^(<e_s>))*Rem |
               -(Dfn^(<e_d>))*(S^(<e_s>))*(Rem^(<e_r>))
```

Figura 6.3: Primera parte de la gramática para el coeficiente de fricción en la primera y segunda aproximación.

```
<e_d> ::=      c[<idx>] | -c[<idx>] |
               c[<idx>]*S | c[<idx>]*S + c[<idx>] | c[<idx>]*S - c[<idx>]|
               -c[<idx>]*S | -c[<idx>]*S + c[<idx>] | -c[<idx>]*S - c[<idx>]|
               c[<idx>]*S^c[<idx>] | c[<idx>]*S^c[<idx>] + c[<idx>] | c[<idx>]*S^c[<idx>] - c[<idx>]|
               -c[<idx>]*S^c[<idx>] | -c[<idx>]*S^c[<idx>] + c[<idx>] | -c[<idx>]*S^c[<idx>] - c[<idx>]|
               c[<idx>]*S^-c[<idx>] | c[<idx>]*S^-c[<idx>] + c[<idx>] | c[<idx>]*S^-c[<idx>] - c[<idx>]|
               -c[<idx>]*S^-c[<idx>] | -c[<idx>]*S^-c[<idx>] + c[<idx>] | -c[<idx>]*S^-c[<idx>] - c[<idx>]|
               c[<idx>]*Rem | c[<idx>]*Rem + c[<idx>] | c[<idx>]*Rem - c[<idx>]|
               -c[<idx>]*Rem | -c[<idx>]*Rem + c[<idx>] | -c[<idx>]*Rem - c[<idx>]|
               c[<idx>]*Rem^c[<idx2>] | c[<idx>]*Rem^c[<idx2>] + c[<idx>] | c[<idx>]*Rem^c[<idx2>] - c[<idx>]|
               -c[<idx>]*Rem^c[<idx2>] | -c[<idx>]*Rem^c[<idx2>] + c[<idx>] | -c[<idx>]*Rem^c[<idx2>] - c[<idx>]|
               c[<idx>]*Rem^-c[<idx2>] | c[<idx>]*Rem^-c[<idx2>] + c[<idx>] | c[<idx>]*Rem^-c[<idx2>] - c[<idx>]|
               -c[<idx>]*Rem^-c[<idx2>] | -c[<idx>]*Rem^-c[<idx2>] + c[<idx>] | -c[<idx>]*Rem^-c[<idx2>] - c[<idx>]|

<e_s> ::=      c[<idx>] | -c[<idx>] |
               c[<idx>]*Dfn | c[<idx>]*Dfn + c[<idx>] | c[<idx>]*Dfn - c[<idx>]|
               -c[<idx>]*Dfn | -c[<idx>]*Dfn + c[<idx>] | -c[<idx>]*Dfn - c[<idx>]|
               c[<idx>]*Dfn^c[<idx>] | c[<idx>]*Dfn^c[<idx>] + c[<idx>] | c[<idx>]*Dfn^c[<idx>] - c[<idx>]|
               -c[<idx>]*Dfn^c[<idx>] | -c[<idx>]*Dfn^c[<idx>] + c[<idx>] | -c[<idx>]*Dfn^c[<idx>] - c[<idx>]|
               c[<idx>]*Dfn^-c[<idx>] | c[<idx>]*Dfn^-c[<idx>] + c[<idx>] | c[<idx>]*Dfn^-c[<idx>] - c[<idx>]|
               -c[<idx>]*Dfn^-c[<idx>] | -c[<idx>]*Dfn^-c[<idx>] + c[<idx>] | -c[<idx>]*Dfn^-c[<idx>] - c[<idx>]|
               c[<idx>]*Rem | c[<idx>]*Rem + c[<idx>] | c[<idx>]*Rem - c[<idx>]|
               -c[<idx>]*Rem | -c[<idx>]*Rem + c[<idx>] | -c[<idx>]*Rem - c[<idx>]|
               c[<idx>]*Rem^c[<idx2>] | c[<idx>]*Rem^c[<idx2>] + c[<idx>] | c[<idx>]*Rem^c[<idx2>] - c[<idx>]|
               -c[<idx>]*Rem^c[<idx2>] | -c[<idx>]*Rem^c[<idx2>] + c[<idx>] | -c[<idx>]*Rem^c[<idx2>] - c[<idx>]|
               c[<idx>]*Rem^-c[<idx2>] | c[<idx>]*Rem^-c[<idx2>] + c[<idx>] | c[<idx>]*Rem^-c[<idx2>] - c[<idx>]|
               -c[<idx>]*Rem^-c[<idx2>] | -c[<idx>]*Rem^-c[<idx2>] + c[<idx>] | -c[<idx>]*Rem^-c[<idx2>] - c[<idx>]|


<e_r> ::=      c[<idx2>] | -c[<idx2>] |
               c[<idx>]*Dfn | c[<idx>]*Dfn + c[<idx>] | c[<idx>]*Dfn - c[<idx>]|
               -c[<idx>]*Dfn | -c[<idx>]*Dfn + c[<idx>] | -c[<idx>]*Dfn - c[<idx>]|
               c[<idx>]*Dfn^c[<idx>] | c[<idx>]*Dfn^c[<idx>] + c[<idx>] | c[<idx>]*Dfn^c[<idx>] - c[<idx>]|
               -c[<idx>]*Dfn^c[<idx>] | -c[<idx>]*Dfn^c[<idx>] + c[<idx>] | -c[<idx>]*Dfn^c[<idx>] - c[<idx>]|
               c[<idx>]*Dfn^-c[<idx>] | c[<idx>]*Dfn^-c[<idx>] + c[<idx>] | c[<idx>]*Dfn^-c[<idx>] - c[<idx>]|
               -c[<idx>]*Dfn^-c[<idx>] | -c[<idx>]*Dfn^-c[<idx>] + c[<idx>] | -c[<idx>]*Dfn^-c[<idx>] - c[<idx>]|
               c[<idx>]*S | c[<idx>]*S + c[<idx>] | c[<idx>]*S - c[<idx>]|
               -c[<idx>]*S | -c[<idx>]*S + c[<idx>] | -c[<idx>]*S - c[<idx>]|
               c[<idx>]*S^c[<idx>] | c[<idx>]*S^c[<idx>] + c[<idx>] | c[<idx>]*S^c[<idx>] - c[<idx>]|
               -c[<idx>]*S^c[<idx>] | -c[<idx>]*S^c[<idx>] + c[<idx>] | -c[<idx>]*S^c[<idx>] - c[<idx>]|
               c[<idx>]*S^-c[<idx>] | c[<idx>]*S^-c[<idx>] + c[<idx>] | c[<idx>]*S^-c[<idx>] - c[<idx>]|
               -c[<idx>]*S^-c[<idx>] | -c[<idx>]*S^-c[<idx>] + c[<idx>] | -c[<idx>]*S^-c[<idx>] - c[<idx>]|

<idx> ::= GE_constants:100
<idx2> ::= GE_constants:1
```

Figura 6.4: Segunda parte de la gramática para el coeficiente de fricción en la primera y segunda aproximación.

A.2. Gramáticas utilizadas en el problema original

Para resolver el problema original descrito en la Sección 3.1.6 se utilizaron las siguientes gramáticas. Para el coeficiente de arrastre y el coeficiente de fricción se utilizó la misma regla de partida mostrada en la Figura 6.5. Las reglas restantes difieren según las variables dependientes de cada coeficiente. En la Figura 6.6 se muestran las reglas restantes utilizadas para el coeficiente de arrastre y en la Figura 6.7 se muestran las reglas restantes utilizadas para el coeficiente de fricción. En la Figura 6.8 se muestra la gramática utilizada para el número de Nusselt en el problema original.

```
<start>   ::=
    <e1> | <e2> | <e3> |
    c[<idx>]*<e1> | c[<idx>]*<e2> | c[<idx>]*<e3> |
    <e1> + <e2> | <e1> + <e3> | <e2> + <e3> |
    c[<idx>]*<e1> + <e2> | c[<idx>]*<e1> + <e3> | c[<idx>]*<e2> + <e3> |
    <e1> + c[<idx>]*<e2> | <e1> + c[<idx>]*<e3> | <e2> + c[<idx>]*<e3> |
    c[<idx>]*<e1> + c[<idx>]*<e2> | c[<idx>]*<e1> + c[<idx>]*<e3> | c[<idx>]*<e2> + c[<idx>]*<e3> |
    <e1> + <e1> | <e2> + <e2> |
    <e1> + c[<idx>]*<e1> | <e2> + c[<idx>]*<e2> |
    c[<idx>]*<e1> + <e1> | c[<idx>]*<e2> + <e2> |
    c[<idx>]*<e1> + c[<idx>]*<e1> | c[<idx>]*<e2> + c[<idx>]*<e2> |
    <e1> + <e2> + <e3> |
    <e1> + <e2> + c[<idx>]*<e3> |
    <e1> + c[<idx>]*<e2> + <e3> |
    <e1> + c[<idx>]*<e2> + c[<idx>]*<e3> |
    c[<idx>]*<e1> + <e2> + <e3> |
    c[<idx>]*<e1> + <e2> + c[<idx>]*<e3> |
    c[<idx>]*<e1> + c[<idx>]*<e2> + <e3> |
    c[<idx>]*<e1> + c[<idx>]*<e2> + c[<idx>]*<e3> |
    <e1> + <e1> + <e1> |
    <e1> + <e1> + c[<idx>]*<e1> |
    <e1> + c[<idx>]*<e1> + c[<idx>]*<e1> |
    c[<idx>]*<e1> + c[<idx>]*<e1> + c[<idx>]*<e1> |
    <e2> + <e2> + <e2> |
    <e2> + <e2> + c[<idx>]*<e2> |
    <e2> + c[<idx>]*<e2> + c[<idx>]*<e2> |
    c[<idx>]*<e2> + c[<idx>]*<e2> + c[<idx>]*<e2> |
    <e1> + <e1> + <e2> |
    <e1> + <e1> + c[<idx>]*<e2> |
    <e1> + c[<idx>]*<e1> + <e2> |
    <e1> + c[<idx>]*<e1> + c[<idx>]*<e2> |
    c[<idx>]*<e1> + c[<idx>]*<e1> + <e2> |
    c[<idx>]*<e1> + c[<idx>]*<e1> + c[<idx>]*<e2> |
    <e2> + <e2> + <e1> |
    <e2> + <e2> + c[<idx>]*<e1> |
    <e2> + c[<idx>]*<e2> + <e1> |
    <e2> + c[<idx>]*<e2> + c[<idx>]*<e1> |
    c[<idx>]*<e2> + c[<idx>]*<e2> + <e1> |
    c[<idx>]*<e2> + c[<idx>]*<e2> + c[<idx>]*<e1> |
    <e1> + <e1> + <e3> |
    <e1> + <e1> + c[<idx>]*<e3> |
    <e1> + c[<idx>]*<e1> + <e3> |
    <e1> + c[<idx>]*<e1> + c[<idx>]*<e3> |
    c[<idx>]*<e1> + c[<idx>]*<e1> + <e3> |
    c[<idx>]*<e1> + c[<idx>]*<e1> + c[<idx>]*<e3> |
```

Figura 6.5: Regla de partida en la gramática utilizada para el coeficiente de arrastre y el coeficiente de fricción del problema original.

```
<e1> ::=
     Rem | An | S |
     -Rem | -An | -S |
     Rem^c[idx2] | An^c[idx] | S^c[idx] |
     Rem^-c[idx2] | An^-c[idx] | S^-c[idx] |
     -Rem^c[idx2] | -An^c[idx] | -S^c[idx] |
     -Rem^-c[idx2] | -An^-c[idx] | -S^-c[idx] |
<e2> ::=
     Rem*An | Rem*S | An*S |
     -Rem*An | -Rem*S | -An*S |
     Rem^c[idx2]*An | Rem^c[idx2]*S | An^c[idx]*S |
     Rem^-c[idx2]*An | Rem^-c[idx2]*S | An^-c[idx]*S |
     -Rem^c[idx2]*An | -Rem^c[idx2]*S | -An^c[idx]*S |
     -Rem^-c[idx2]*An | -Rem^-c[idx2]*S | -An^-c[idx]*S |
     Rem*An^c[idx] | Rem*S^c[idx] | An*S^c[idx] |
     Rem*An^-c[idx] | Rem*S^-c[idx] | An*S^-c[idx] |
     -Rem*An^c[idx] | -Rem*S^c[idx] | -An*S^c[idx] |
     -Rem*An^-c[idx] | -Rem*S^-c[idx] | -An*S^-c[idx] |
     Rem^c[idx2]*An^c[idx] | Rem^c[idx2]*S^c[idx] | An^c[idx]*S^c[idx] |
     Rem^c[idx2]*An^-c[idx] | Rem^c[idx2]*S^-c[idx] | An^c[idx]*S^-c[idx] |
     Rem^-c[idx2]*An^c[idx] | Rem^-c[idx2]*S^c[idx] | An^-c[idx]*S^c[idx] |
     Rem^-c[idx2]*An^-c[idx] | Rem^-c[idx2]*S^-c[idx] | An^-c[idx]*S^-c[idx] |
     -Rem^c[idx2]*An^c[idx] | -Rem^c[idx2]*S^c[idx] | -An^c[idx]*S^c[idx] |
     -Rem^c[idx2]*An^-c[idx] | -Rem^c[idx2]*S^-c[idx] | -An^c[idx]*S^-c[idx] |
     -Rem^-c[idx2]*An^c[idx] | -Rem^-c[idx2]*S^c[idx] | -An^-c[idx]*S^c[idx] |
     -Rem^-c[idx2]*An^-c[idx] | -Rem^-c[idx2]*S^-c[idx] | -An^-c[idx]*S^-c[idx] |
<e3> ::=
     An*Rem*S | -An*Rem*S |
     An*Rem*S^c[idx] | -An*Rem*S^c[idx] | An*Rem*S^-c[idx] | -An*Rem*S^-c[idx] |
     An*Rem^c[idx2]*S | -An*Rem^c[idx2]*S | An*Rem^-c[idx2]*S | -An*Rem^-c[idx2]*S |
     An*Rem^c[idx2]*S^c[idx] | An*Rem^c[idx2]*S^-c[idx] | An*Rem^-c[idx2]*S^c[idx] | An*Rem^-c[idx2]*S^-c[idx] |
     -An*Rem^c[idx2]*S^c[idx] | -An*Rem^c[idx2]*S^-c[idx] | -An*Rem^-c[idx2]*S^c[idx] | -An*Rem^-c[idx2]*S^-c[idx] |
     An^c[idx]*Rem*S | -An^c[idx]*Rem*S | An^-c[idx]*Rem*S | -An^-c[idx]*Rem*S |
     An^c[idx]*Rem*S^c[idx] | An^c[idx]*Rem*S^-c[idx] | An^-c[idx]*Rem*S^c[idx] | An^-c[idx]*Rem*S^-c[idx] |
     -An^c[idx]*Rem*S^c[idx] | -An^c[idx]*Rem*S^-c[idx] | -An^-c[idx]*Rem*S^c[idx] | -An^-c[idx]*Rem*S^-c[idx] |
     An^c[idx]*Rem^c[idx2]*S | An^c[idx]*Rem^-c[idx2]*S | An^-c[idx]*Rem^c[idx2]*S | An^-c[idx]*Rem^-c[idx2]*S |
     -An^c[idx]*Rem^c[idx2]*S | -An^c[idx]*Rem^-c[idx2]*S | -An^-c[idx]*Rem^c[idx2]*S | -An^-c[idx]*Rem^-c[idx2]*S |
     An^c[idx]*Rem^c[idx2]*S^c[idx] | An^c[idx]*Rem^c[idx2]*S^-c[idx] |
     An^c[idx]*Rem^-c[idx2]*S^c[idx] | An^c[idx]*Rem^-c[idx2]*S^-c[idx] |
     An^-c[idx]*Rem^c[idx2]*S^c[idx] | An^-c[idx]*Rem^c[idx2]*S^-c[idx] |
     An^-c[idx]*Rem^-c[idx2]*S^c[idx] | An^-c[idx]*Rem^-c[idx2]*S^-c[idx] |
     -An^c[idx]*Rem^c[idx2]*S^c[idx] | -An^c[idx]*Rem^c[idx2]*S^-c[idx] |
     -An^c[idx]*Rem^-c[idx2]*S^c[idx] | -An^c[idx]*Rem^-c[idx2]*S^-c[idx] |
     -An^-c[idx]*Rem^c[idx2]*S^c[idx] | -An^-c[idx]*Rem^c[idx2]*S^-c[idx] |
     -An^-c[idx]*Rem^-c[idx2]*S^c[idx] | -An^-c[idx]*Rem^-c[idx2]*S^-c[idx] |
<idx> ::= GE_constants:100
<idx2> ::= GE_constants:2
```

Figura 6.6: Segunda parte de la gramática utilizada para el coeficiente de arrastre del problema original.

```
<e1> ::=
     Rem | Vmfn | S |
     -Rem | -Vmfn | -S |
     Rem^c[idx2] | Vmfn^c[idx] | S^c[idx] |
     Rem^-c[idx2] | Vmfn^-c[idx] | S^-c[idx] |
     -Rem^c[idx2] | -Vmfn^c[idx] | -S^c[idx] |
     -Rem^-c[idx2] | -Vmfn^-c[idx] | -S^-c[idx] |
<e2> ::=
     Rem*Vmfn | Rem*S | Vmfn*S |
     -Rem*Vmfn | -Rem*S | -Vmfn*S |
     Rem^c[idx2]*Vmfn | Rem^c[idx2]*S | Vmfn^c[idx]*S |
     Rem^-c[idx2]*Vmfn | Rem^-c[idx2]*S | Vmfn^-c[idx]*S |
     -Rem^c[idx2]*Vmfn | -Rem^c[idx2]*S | -Vmfn^c[idx]*S |
     -Rem^-c[idx2]*Vmfn | -Rem^-c[idx2]*S | -Vmfn^-c[idx]*S |
     Rem*Vmfn^c[idx] | Rem*S^c[idx] | Vmfn*S^c[idx] |
     Rem*Vmfn^-c[idx] | Rem*S^-c[idx] | Vmfn*S^-c[idx] |
     -Rem*Vmfn^c[idx] | -Rem*S^c[idx] | -Vmfn*S^c[idx] |
     -Rem*Vmfn^-c[idx] | -Rem*S^-c[idx] | -Vmfn*S^-c[idx] |
     Rem^c[idx2]*Vmfn^c[idx] | Rem^c[idx2]*S^c[idx] | Vmfn^c[idx]*S^c[idx] |
     Rem^c[idx2]*Vmfn^-c[idx] | Rem^c[idx2]*S^-c[idx] | Vmfn^c[idx]*S^-c[idx] |
     Rem^-c[idx2]*Vmfn^c[idx] | Rem^-c[idx2]*S^c[idx] | Vmfn^-c[idx]*S^c[idx] |
     Rem^-c[idx2]*Vmfn^-c[idx] | Rem^-c[idx2]*S^-c[idx] | Vmfn^-c[idx]*S^-c[idx] |
     -Rem^c[idx2]*Vmfn^c[idx] | -Rem^c[idx2]*S^c[idx] | -Vmfn^c[idx]*S^c[idx] |
     -Rem^c[idx2]*Vmfn^-c[idx] | -Rem^c[idx2]*S^-c[idx] | -Vmfn^c[idx]*S^-c[idx] |
     -Rem^-c[idx2]*Vmfn^c[idx] | -Rem^-c[idx2]*S^c[idx] | -Vmfn^-c[idx]*S^c[idx] |
     -Rem^-c[idx2]*Vmfn^-c[idx] | -Rem^-c[idx2]*S^-c[idx] | -Vmfn^-c[idx]*S^-c[idx] |
<e3> ::=
     Vmfn*Rem*S | -Vmfn*Rem*S |
     Vmfn*Rem*S^c[idx] | -Vmfn*Rem*S^c[idx] | Vmfn*Rem*S^-c[idx] | -Vmfn*Rem*S^-c[idx] |
     Vmfn*Rem^c[idx2]*S | -Vmfn*Rem^c[idx2]*S | Vmfn*Rem^-c[idx2]*S | -Vmfn*Rem^-c[idx2]*S |
     Vmfn*Rem^c[idx2]*S^c[idx] | Vmfn*Rem^c[idx2]*S^-c[idx] | Vmfn*Rem^-c[idx2]*S^c[idx] | Vmfn*Rem^-c[idx2]*S^-c[idx] |
     -Vmfn*Rem^c[idx2]*S^c[idx] | -Vmfn*Rem^c[idx2]*S^-c[idx] | -Vmfn*Rem^-c[idx2]*S^c[idx] | -Vmfn*Rem^-c[idx2]*S^-c[idx] |
     Vmfn^c[idx]*Rem*S | -Vmfn^c[idx]*Rem*S | Vmfn^-c[idx]*Rem*S | -Vmfn^-c[idx]*Rem*S |
     Vmfn^c[idx]*Rem*S^c[idx] | Vmfn^c[idx]*Rem*S^-c[idx] | Vmfn^-c[idx]*Rem*S^c[idx] | Vmfn^-c[idx]*Rem*S^-c[idx] |
     -Vmfn^c[idx]*Rem*S^c[idx] | -Vmfn^c[idx]*Rem*S^-c[idx] | -Vmfn^-c[idx]*Rem*S^c[idx] | -Vmfn^-c[idx]*Rem*S^-c[idx] |
     Vmfn^c[idx]*Rem^c[idx2]*S | Vmfn^c[idx]*Rem^-c[idx2]*S | Vmfn^-c[idx]*Rem^c[idx2]*S | Vmfn^-c[idx]*Rem^-c[idx2]*S |
     -Vmfn^c[idx]*Rem^c[idx2]*S | -Vmfn^c[idx]*Rem^-c[idx2]*S | -Vmfn^-c[idx]*Rem^c[idx2]*S | -Vmfn^-c[idx]*Rem^-c[idx2]*S |
     Vmfn^c[idx]*Rem^c[idx2]*S^c[idx] | Vmfn^c[idx]*Rem^c[idx2]*S^-c[idx] |
     Vmfn^c[idx]*Rem^-c[idx2]*S^c[idx] | Vmfn^c[idx]*Rem^-c[idx2]*S^-c[idx] |
     Vmfn^-c[idx]*Rem^c[idx2]*S^c[idx] | Vmfn^-c[idx]*Rem^c[idx2]*S^-c[idx] |
     Vmfn^-c[idx]*Rem^-c[idx2]*S^c[idx] | Vmfn^-c[idx]*Rem^-c[idx2]*S^-c[idx] |
     -Vmfn^c[idx]*Rem^c[idx2]*S^c[idx] | -Vmfn^c[idx]*Rem^c[idx2]*S^-c[idx] |
     -Vmfn^c[idx]*Rem^-c[idx2]*S^c[idx] | -Vmfn^c[idx]*Rem^-c[idx2]*S^-c[idx] |
     -Vmfn^-c[idx]*Rem^c[idx2]*S^c[idx] | -Vmfn^-c[idx]*Rem^c[idx2]*S^-c[idx] |
     -Vmfn^-c[idx]*Rem^-c[idx2]*S^c[idx] | -Vmfn^-c[idx]*Rem^-c[idx2]*S^-c[idx] |
<idx> ::= GE_constants:100
<idx2> ::= GE_constants:2
```

Figura 6.7: Segunda parte de la gramática utilizada para el coeficiente de fricción del problema original.

```
<start>   ::=
    Rem | Rem^c[<idx2>] | Rem^-c[<idx2>] |
    c[<idx>]*Rem | c[<idx>]*Rem^c[<idx2>] | c[<idx>]*Rem^-c[<idx2>] |

    c[<idx>]+Rem | c[<idx>]+Rem^c[<idx2>] | c[<idx>]+Rem^-c[<idx2>] |
    -c[<idx>]+Rem | -c[<idx>]+Rem^c[<idx2>] | -c[<idx>]+Rem^-c[<idx2>] |
    c[<idx>]-Rem | c[<idx>]-Rem^c[<idx2>] | c[<idx>]-Rem^-c[<idx2>] |

    c[<idx>]+c[<idx>]*Rem | c[<idx>]+c[<idx>]*Rem^c[<idx2>] | c[<idx>]+c[<idx>]*Rem^-c[<idx2>] |
    -c[<idx>]+c[<idx>]*Rem | -c[<idx>]+c[<idx>]*Rem^c[<idx2>] | -c[<idx>]+c[<idx>]*Rem^-c[<idx2>] |
    c[<idx>]-c[<idx>]*Rem | c[<idx>]-c[<idx>]*Rem^c[<idx2>] | c[<idx>]-c[<idx>]*Rem^-c[<idx2>] |
<idx> ::= GE_constants:100
<idx2> ::= GE_constants:2
```

Figura 6.8: Gramática utilizada para el número de Nusselt del problema original.