CONSTRUCCIÓN DE UNA HERRAMIENTA SOFTWARE PARA MEJORAMIENTO DEL POSICIONAMIENTO DE POZOS EN EL DESARROLLO DE UN CAMPO MADURO DE HIDROCARBUROS USANDO NEUROSIMULACIÓN

HECTOR EMILIO BARRIOS MOLANO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICO-QUÍMICAS
ESCUELA DE INGENIERÍA DE PETRÓLEOS
BUCARAMANGA
2009

CONSTRUCCIÓN DE UNA HERRAMIENTA SOFTWARE PARA MEJORAMIENTO DEL POSICIONAMIENTO DE POZOS EN EL DESARROLLO DE UN CAMPO MADURO DE HIDROCARBUROS USANDO NEUROSIMULACIÓN

HECTOR EMILIO BARRIOS MOLANO

Trabajo de Grado presentado como requisito para optar el título de Ingeniero de Petróleos

Director:
Ing. ELKIN RODOLFO SANTAFÉ RANGEL

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICO-QUÍMICAS
ESCUELA DE INGENIERÍA DE PETRÓLEOS
BUCARAMANGA
2009

Reconocimiento-No comercial-Compartir bajo la misma licencia 2.5 Colombia

Usted es libre de:

Copiar, distribuir y comunicar públicamente la obra

Hacer obras derivadas

Bajo las condiciones siguientes:

Reconocimiento — Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador (pero no de una manera que sugiera que tiene su apoyo o apoyan el uso que hace de su obra).

No comercial — No puede utilizar esta obra para fines comerciales.

Compartir bajo la misma licencia — Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

Con el entendimiento que:

Renuncia — Cualquiera de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor.

Otros derechos — De ninguna manera cualquiera de los siguientes derechos son afectados por la licencia:

- Su trato justo o los derechos de uso justo;
- Los derechos morales del autor;
- Derechos que otras personas pueden tener, tanto en la propia obra o en la forma en que la obra se utiliza, como la publicidad o derechos de privacidad.

Aviso — Al reutilizar o Distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.

Esto es un resumen fácilmente legible del texto legal (la licencia completa).
`http://creativecommons.org/licenses/by-nc-sa/2.5/co/legalcode`

Redacción y edición de tesis
con LaTeX 2$_\varepsilon$, *VIM*
y sistema operativo libre
GNU/LINUX.

A mi papá Libardo mi mamá Nancy Piedad y mi hermano Cristian Alberto

♡

Un proceso no puede ser comprendido más que interrumpiéndolo. La comprensión debe fluir al mismo tiempo que el proceso, debe unirse a él y caminar con él.

Frank Herberth

Creo que el conocimiento científico posee propiedades fractales, que no importa cuánto podamos aprender; que todo lo que quede, por pequeño que pueda parecer, es tan infinitamente complejo como el total del cual partimos. Ése, creo, es el secreto del universo.

Isaac Asimov

Agradecimientos

A DIOS.

A mi familia, porque todo lo que soy se lo debo a mi papá, mi mamá y mi hermano.

A la Universidad Industrial de Santander, por fomentar un ambiente académico e investigativo propicio para el desarrollo de este proyectos y permitir que me formara tanto en la parte académica como en la parte personal.

A el Grupo de Investigación en Tecnologías Alternativas para Hidrocarburos (GITAH) por todo su apoyo, aportes y sugerencias durante el desarrollo de este trabajo.

En especial a el profesor Elkin Rodolfo Santafé por su apoyo, ayuda y consejos que permitieron desarrollar este trabajo.

A cada una de las cosas, a cada granito de arena que estaba en el lugar y el momento que debía estar para que al final de esta madrugada yo esté escribiendo estas líneas y pueda decir que he terminado el libro de mi tesis.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

RESUMEN

TÍTULO: CONSTRUCCIÓN DE UNA HERRAMIENTA SOFTWARE PARA MEJORAMIENTO DEL POSICIONAMIENTO DE POZOS EN EL DESARROLLO DE UN CAMPO MADURO DE HIDROCARBUROS USANDO NEUROSIMULACIÓN[a]

AUTOR: HECTOR EMILIO BARRIOS MOLANO[b]

PALABRAS CLAVES: Neurosimulación, Software Libre, Redes Neuronales Artificiales, Simulación Numérica de Yacimientos, Python, Desarrollo de Campos Maduros, Tecnologías Alternativas, Infill.

El presente trabajo muestra el desarrollo de una herramienta software que implementa una técnica de neurosimulación para la ubicación de pozos aplicado en el desarrollo de un campo de hidrocarburos heterogéneo y de geometría irregular.

Durante el desarrollo de un campo de hidrocarburos la ubicación de pozos es una de las tareas más importantes, ya que un cambio pequeño en la ubicación puede representar ganancias o pérdidas durante el resto de vida productiva del campo, los métodos convencionales usados consumen gran cantidad de tiempo y de esfuerzo computacional, una alternativa a estos métodos es usar neurosimulación, técnica que forma un puente eficaz entre redes neuronales artificiales y simulación numérica de yacimientos. En este trabajo se describe el desarrollo de una herramienta de software que hace uso de la técnica de neurosimulación para la ubicación de pozos, por medio de una interfaz gráfica es posible ingresar los datos necesarios para correr el programa. La salida del programa son las ubicaciones de los pozos nuevos, con la opción de visualizar los resultados por medio de gráficas.

La herramienta software fue desarrollada utilizando herramientas de código abierto y software libre, como por ejemplo el lenguaje de programación Python, tanto para reducir costos durante su elaboración, como para demostrar la potencia y utilidad e incentivar su uso en la investigación, en la academia y en la industria de los hidrocarburos.

Con la ayuda de la herramienta software desarrollada en este trabajo es posible obtener de una manera sencilla, rápida y precisa la ubicación de los pozos para un determinado programa de perforación durante el desarrollo de un campo maduro de hidrocarburos.

[a] Proyecto de Grado

[b] Facultad de Ingenierías Fisicoquímicas. Escuela de Ingeniería de Petróleos. Director Ingeniero de Petróleos Elkin Rodolfo Santafé Rangel

ABSTRACT

TITLE: CONSTRUCTION OF A SOFTWARE TOOL FOR IMPROVING THE WELL POSITIONING FOR THE DEVELOPMENT OF A MATURE HYDROCARBON FIELD USING NEUROSIMULATION[c]

AUTHOR: HECTOR EMILIO BARRIOS MOLANO[d]

KEYWORDS: Neurosimulation, Free Software, Artificial Neural Networks, Numerical Reservoir Simulation, Python, Mature Field Development, Alternative Technologies, Infill.

This work shows the development of a software tool that implement a neurosimulation technique for the well placement applied in developing a heterogeneous hydrocarbon field with an irregular geometry.

During the development of a hydrocarbon field the well placement is a major task, because a small change in location can make gains or losses during the remaining productive life of the field, the conventional methods of well placement are expensive and consume large amounts of time and computational effort, an alternative to this methods is neurosimulation, this technique forms a bridge between artificial neural networks (ANN) and numerical reservoir simulation (NRS). This work describes the development of a software tool that make use of the neurosimulation technique for well placement, through a graphical interface is possible input necessary data to run the program. The program output are the new well locations, with the option of visualize the results through graphics.

The software tool was developed using open source tools and free software, for example the Python programing language, to encourage their use and development in the field of research in academia and hydrocarbon industry.

With the help of the software tool developed on this work is possible obtain in a simple way fast and accurate well locations for a given drilling plan during the development of a mature hydrocarbon field.

[c]Undergraduate Project

[d]Physiochemical Engineering Faculty. School of Petroleum Engineering. Director Petroleum Engineer Elkin Rodolfo Santafé Rangel

1 INTRODUCCIÓN

Algunos de los principales desafíos enfrentados durante el desarrollo de nuevos pozos de un campo de hidrocarburos son: (1)la presencia de gran número de grados de libertad; (2)la posibilidad de tener gran interacción entre los parámetros; (3)la naturaleza dispersa y escasa de los datos de campo; (4)limitaciones de tiempo; (5)limitaciones económicas. Convencionalmente se usan los datos de producción para realizar un ajuste histórico. Las mejores ubicaciones de los pozos se deciden corriendo el simulador de yacimientos colocando los pozos nuevos en diferentes ubicaciones. Esto consume altas cantidades de tiempo y esfuerzo computacional ya que los modelos típicos de yacimientos tienen alrededor de 10^5 a 10^6 celdas.

En este trabajo se expone la técnica de neurosimulación como alternativa a los métodos convencionales de ubicación de pozos, y se muestra su implementación en una herramienta software, tomando como base de su desarrollo software libre y código abierto.

1.1. UBICACIÓN DE POZOS

El desarrollo óptimo de un yacimiento depende en gran medida de las localizaciones de pozos. La determinación de dichas localizaciones es un problema complejo que depende de muchos factores como las propiedades del yacimiento y de los fluidos, especificaciones de los equipos de superficie y criterio económico[5]. Tradicionalmente esta tarea se ha llevado a cabo por medio de un proceso manual de prueba y error, colocando los pozos nuevos en diferentes ubicaciones en el modelo de simulación y haciendo varias corridas[7, 13]. Otra solución tradicional es el uso de técnicas empíricas basadas en estimados de áreas de drenaje que no tienen en cuenta el impacto de las heterogeneidades del yacimiento.

Para solucionar este problema existen diferentes técnicas, algunas usan algoritmos genéticos[12], otras usan algoritmos híbridos (algoritmos genéticos, algoritmo politopo, algoritmo de kriging, redes neuronales artificiales)[5], (algoritmos genéticos y mapas de calidad)[4], simulación streamline[10], y neurosimulación[6, 7, 8], esta última es utilizada en el presente trabajo.

2 GENERALIDADES

2.1. REDES NEURONALES ARTIFICIALES

Una red neuronal artificial (RNA), usualmente llamada “red neuronal”, es un modelo matemático o modelo computacional que intenta simular la estructura y/o aspectos funcionales de las redes neuronales biológicas. En la mayoría de los casos una red neuronal artificial es un sistema adaptativo que cambia su estructura basado en información interna o externa que fluye a través de la red en la fase de aprendizaje. Las redes neuronales son consideras herramientas estadísticas no lineales de modelamiento de datos. Pueden ser utilizadas para modelar relaciones complejas entre entradas y salidas o para encontrar patrones en los datos.

2.1.1. Definición

No existe una definición general entre investigadores de lo que es una red neuronal artificial, pero la mayoría concuerda que esta involucra una red de elementos de procesamiento simple(neuronas), las cuales pueden exhibir un comportamiento global complejo dependiendo de las conexiones entre los elementos de procesamiento y los parámetros de los elementos. La inspiración original para la técnica viene de la examinación del sistema nervioso central y las neuronas, sus axones, dendritas y sinapsis Figura 2.1, las cuales constituyen uno de los elementos de procesamiento de información más significativos. Otra similitud con entre redes neuronales biológicas y artificiales es que las funciones son realizadas colectivamente y en paralelo por las unidades, en lugar de tener una clara delimitación de subtareas asignadas a ciertas unidades. En la Figura 2.2 se muestra la semejanza entre las redes neuronales artificiales y las redes neuronales biológicas.

En las implementaciones modernas de software de redes neuronales artificiales el

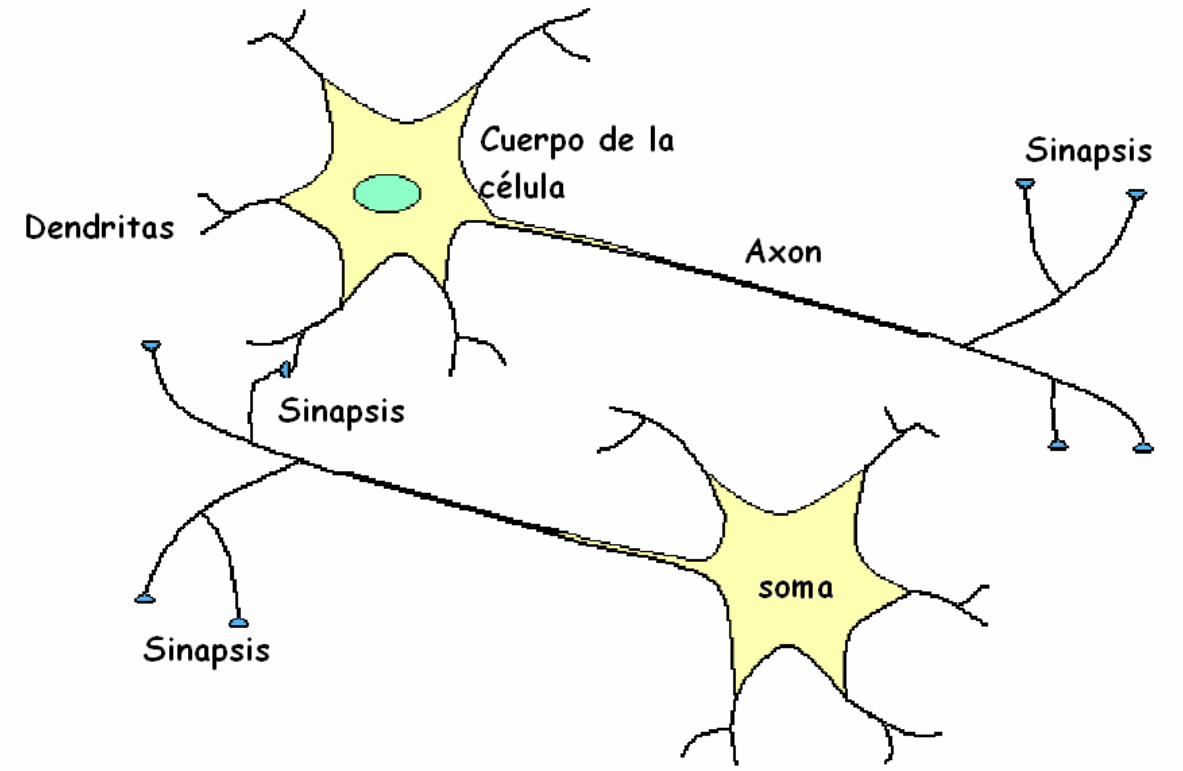


Figura 2.1: Neurona biológica y sus partes.*Tutorial de RNA's Universidad Tecnológica de Pereira*

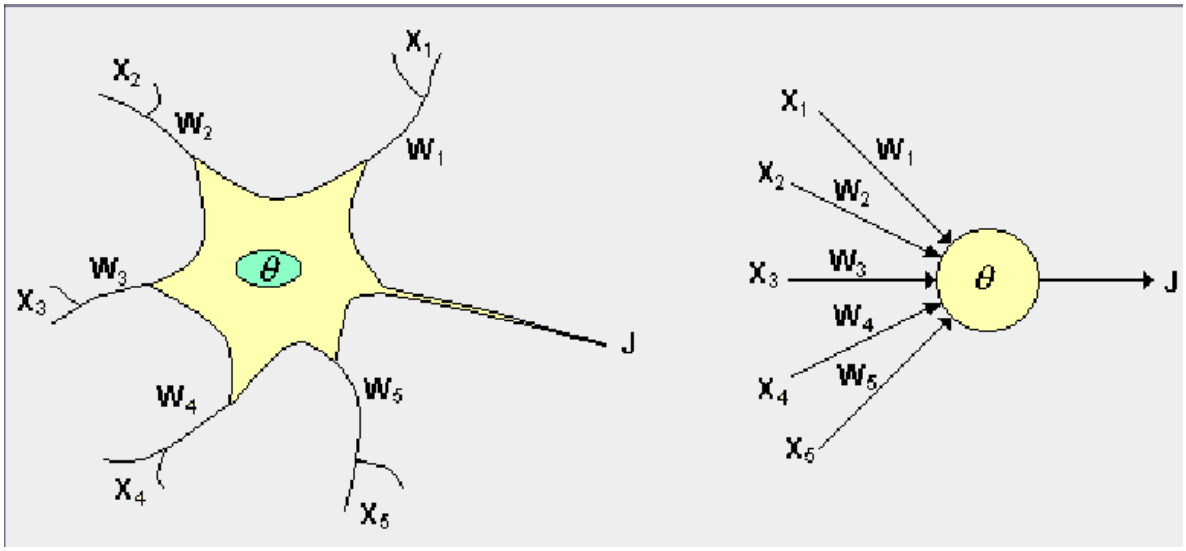


Figura 2.2: Semejanza entre redes neuronales biológicas y redes neuronales artificiales.*Tutorial de RNA's Universidad Tecnológica de Pereira*

enfoque inspirado por la biología ha sido la mayor parte abandonado por un enfoque más práctico basado en estadística y procesamiento de señales. En algunos de los sistemas, las redes neuronales artificiales o partes de redes neuronales artificiales son usadas como componentes de sistemas más grandes que combinan tanto elementos adaptativos como no adaptativos.

2.1.2. Funcionamiento

Cada neurona que compone la red neuronal artificial recibe una serie de entradas a través de interconexiones y emite una salida que viene dada por tres funciones:

Función de propagación

Conocida también como función de excitación, es generalmente la sumatoria de cada entrada multiplicada por el peso de su interconexión, si el peso es positivo se denomina *excitatoria*, si es negativo se denomina *inhibitoria*.

Función de transferencia

Aplicada al valor devuelto por la función de propagación. Se utiliza para acotar la salida de la neurona y generalmente viene dada por la interpretación que se le den a estas. Algunas funciones de transferencias muy usadas son la función sigmoidea Ecuación 2.1 con la cual se obtienen valores en el intervalo $[0, 1]$, la tangente hiperbólica Ecuación 2.2 con la cual se obtienen valores en el intervalo $[-1, 1]$. En la Figura 2.3 se muestra el esquema de una neurona con dos entradas (X), sus respectivos pesos (W), la función de propagación, la función de transferencia y la salida (Y).

$$S(t) = \frac{1}{1 + e^{-t}} \tag{2.1}$$

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.2}$$

2.1.3. Tipos de Redes Neuronales Artificiales

Las redes neuronales artificiales según el patrón de conexiones se pueden clasificar:

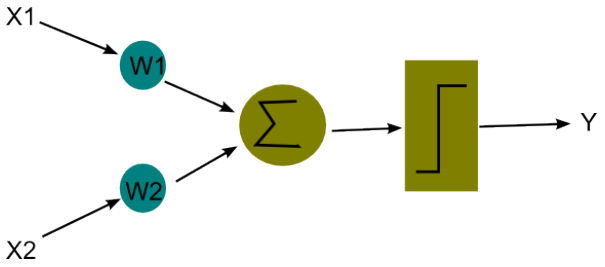


Figura 2.3: Esquema general de una neurona con dos entradas.

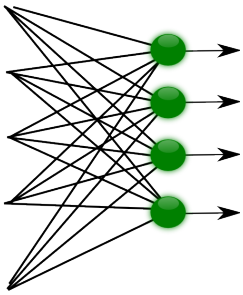


Figura 2.4: Red neuronal monocapa.

Propagación hacia adelante

En estas redes las señales van desde la capa de entrada hacia la salida sin ciclos ni conexiones entre neuronas de la misma capa.

- **Monocapa** Figura 2.4
- **Multicapa** Figura 2.5

Recurrentes

Estas redes presentan al menos un ciclo cerrado, como se muestra en la Figura 2.6

2.1.4. Apendizaje

Según el tipo de aprendizaje las redes neuronales se pueden clasificar:

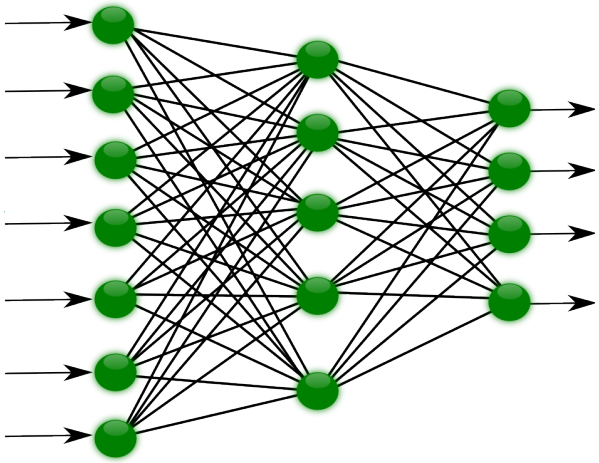


Figura 2.5: Red neuronal multicapa.

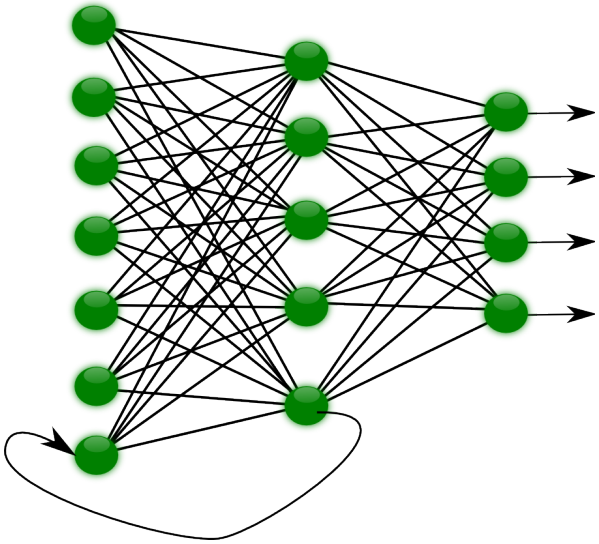


Figura 2.6: Red neuronal recurrente.

Aprendizaje supervisado

Se necesita de un conjunto de datos de entrada el cual se conoce la respuesta. Este tipo de redes son usadas en el reconocimiento de patrones y regresión (aproximación de funciones).

Aprendizaje no supervisado

Este tipo de redes no necesita de un conjunto inicial de datos de entrada y salida, en su lugar necesitan de datos de entrada y la función costo a ser minimizada. Aplicaciones de estas redes puede ser la estimación de distribuciones estadísticas, compresión y filtrado de datos.

Aprendizaje reforzado

En este caso los datos de entrada no son dados a la red neuronal, pero son generados por las interacciones con el ambiente. Las tareas que caen dentro del paradigma de aprendizaje reforzado son problemas de control, juegos y otras tareas de toma de decisiones secuencial.

2.1.5. Aplicación en la ingeniería de petróleos

Las características de las redes neuronales artificiales son adecuadas en los siguientes casos:

- Problemas altamente no lineales en los cuales no se pueden definir de forma clara reglas para resolverlos o son difíciles de formalizar, pero existen conjuntos de datos de entrada y salida conocidos.
- Problemas que tienen *ruido* en los datos.
- Problemas que requieren alta velocidad de procesamiento.

Ali J.K.[2] muestra las aplicaciones de las redes neuronales en la industria de los hidrocarburos estas son:

Geología y geofísica.

- Estimación de reservas.
- Identificación de minerales en los registros de pozo.
- Localización de pozos a perforar en datos geofísicos.
- Comprensión de datos sismográficos en la prospección petrolífera.

Perforación y completamiento de pozos.

- Análisis de datos de perforación.
- Diagnóstico de desgaste de brocas.
- Selección y monitoreo de brocas y lodos de perforación.

Evaluación de formaciones.

- Predicción de la porosidad.
- Predicción de la permeabilidad.
- Detección y delineamiento de fracturas.
- Identificación e interpretación de registros.
- Presiones de poro.

Producción y facilidades.

- Análisis y diagnóstico de problemas en la bomba de subsuelo de bombeo mecánico mediante dinagramas.
- Análisis e implementación de producciones de gas y petróleo.
- Inspección de tuberías de producción.
- Localización de plataformas de perforación costa afuera.

Ingeniería de yacimientos.

- Análisis de parámetros de producción de gas y petróleo.
- Inversión de modelos de simulación.

- Ajuste histórico automático.
- Reconocimiento de patrones de presión en pruebas de pozo.
- Predicción de las propiedades del yacimiento.

Negocios del petróleo.

- Análisis de mercadeo de gas y petróleo.
- Análisis de riesgo.
- Optimización de portafolio de negocios.
- Coordinación de datos económicos.

2.2. SIMULACIÓN NUMÉRICA DE YACIMIENTOS

La simulación de un yacimiento de petróleo se refiere a la construcción y operación de un modelo cuyo comportamiento asume la apariencia del comportamiento del yacimiento actual[15].

El objetivo de la simulación de yacimientos es entender los complejos procesos químicos, físicos, y de flujo de fluidos que ocurren en un yacimiento de hidrocarburos lo suficiente para optimizar el recobro de hidrocarburos, para cumplir este objetivo se debe estar en capacidad de predecir el comportamiento del yacimiento bajo diferentes esquemas de explotación[14].

2.2.1. Modelamiento

Las principales etapas durante el modelamiento de un yacimiento se muestran en la Figura 2.7.

Modelo físico

En esta etapa se desarrolla un modelo físico del proceso de flujo donde se incorporan tantos detalles físicos como sean necesarios para describir el proceso o problema.

Formulación matemática

Se obtiene una formulación matemática del modelo físico, usualmente implica sistemas de ecuaciones diferenciales parciales.

Modelo numérico

Una vez las propiedades del modelo matemático tales como existencia, singularidad y regularidad de la solución son entendidas lo suficiente y las propiedades parecen compatibles con el modelo físico, se produce un modelo numérico discretizado.

Modelo Computacional

Luego, se construye un programa de cómputo capaz de realizar eficientemente los cálculos para el modelo numérico.

Comparación con datos reales

Finalmente una vez se ha desarrollado el programa sus resultados son comparados con datos medidos del proceso físico. Si los resultados no son semejantes se debe regresar al inicio del modelamiento y hacer los cambios necesarios hasta que se termine el proceso de modelamiento.

2.2.2. Lo que un modelo de simulación de yacimiento puede responder

Las siguientes son preguntas que pueden ser resueltas usando un modelo de yacimiento:

- ¿Cómo debe ser desarrollado y producido el campo para maximizar el recobro económico de hidrocarburos?

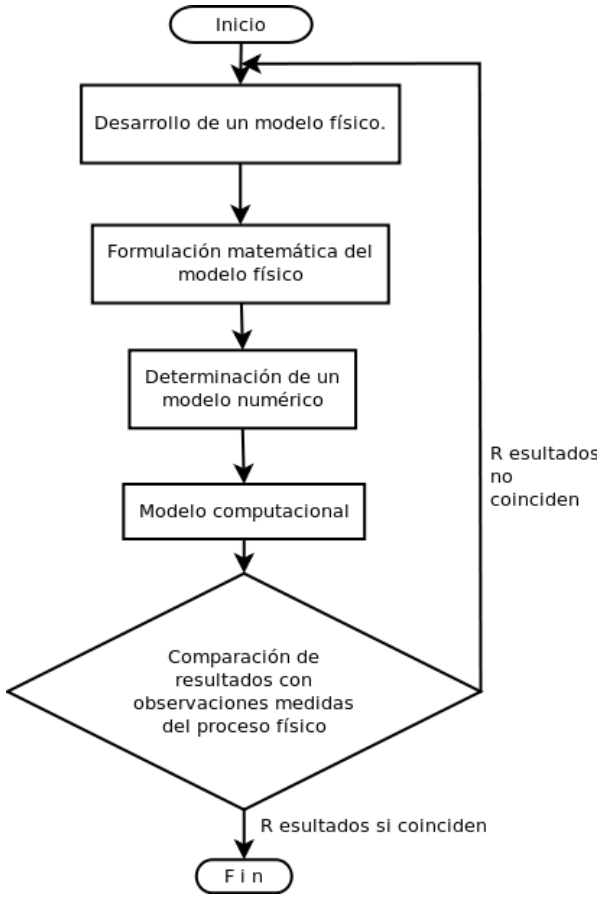


Figura 2.7: Proceso de modelamiento.

- ¿Cuál es el mejor esquema de recobro mejorado para el yacimiento?¿cómo debe ser implementado?
- ¿Cuál es el recobro final del campo?
- ¿cuál es la sensibilidad del modelo a diferentes variables?
- ¿Cuales son los parámetros críticos que deben ser medidos en el campo para un esquema de recobro?
- ¿Cuál es el mejor esquema de completamiento para los pozos en el yacimiento?
- ¿De cuales zonas del yacimiento se está produciendo?

2.3. NEUROSIMULACIÓN

La técnica de neurosimulación busca el establecimiento de una simbiosis poderosa entre *computación dura* (rigurosos cálculos numéricos) y protocolos de *computación suave* (redes neuronales artificiales). En neurosimulación, las técnicas de computación dura están acopladas con técnicas de computación suave para el desarrollo de sistemas expertos poderosos. Los modelos numéricos proveen una "experiencia" precisa y formal, a un costo computacional significativo, que puede ser enseñada a una herramienta de computación suave que, una vez entrenada, puede aprovechar y aplicar la experiencia aprendida a un trabajo computacional mucho menos intenso.

Esta técnica ha sido aplicada a:

- Análisis del desempeño del ciclo de inyección de gas en yacimientos de gas / condensado[3].
- Predicción del desempeño en proyectos de metano asociado a mantos de carbón y secuestro de CO_2[11].
- Flujo trifásico contracorriente usando tomografía de rayos X computarizada y neurosimulación[1].
- Ubicación de pozos de desarrollo[8, 7, 6].

2.4. SOFTWARE LIBRE

La Fundación para el Software Libre[a] define:

> El software libre es una cuestión de la libertad de los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Más precisamente, significa que los usuarios de programas tienen las cuatro libertades esenciales.
>
> - La libertad de ejecutar el programa, para cualquier propósito (libertad 0).

[a] `http://www.gnu.org/philosophy/free-sw.es.html`

- La libertad de estudiar cómo trabaja el programa, y cambiarlo para que haga lo que usted quiera (libertad 1). El acceso al código fuente es una condición necesaria para ello.
- La libertad de redistribuir copias para que pueda ayudar al prójimo (libertad 2).
- La libertad de distribuir copias de sus versiones modificadas a terceros (la 3ª libertad). Si lo hace, puede dar a toda la comunidad una oportunidad de beneficiarse de sus cambios. El acceso al código fuente es una condición necesaria para ello.

Estas libertades están contenidas en la Licencia Pública General GNU (GNU GPL), en el siguiente enlace se puede encontrar el texto completo de la licencia `http://www.gnu.org/copyleft/gpl.html`

3 METODOLOGÍA

3.1. METODOLOGÍA SEGÚN LITERATURA

Según la literatura [6] la metodología para la ubicación de pozos de desarrollo usando neurosimulación se muestra en la Figura 3.1.

La metodología consiste en primero que todo dividir el dominio físico de interés (en este caso el yacimiento) y ubicar los pozos existentes o "viejos", usualmente se toman las mismas divisiones hechas en el modelo de simulación, este paso es conocido como *discretización*.

Luego, un grupo de pozos llamados *pozos de entrenamiento son seleccionados*, estos pozos solo son ubicados en el modelo de simulación y es por medio de estos que la información acerca del yacimiento es obtenida para entrenar la red neuronal artificial, el número de pozos de entrenamiento y la ubicación de estos dependen del caso estudiado, sin embargo, hay que tener en cuenta que las redes neuronales artificiales realizan una interpolación no lineal entre las coordenadas de los pozos de entrenamiento para predecir la producción de cada pozo nuevo, es por esto que se debe asegurar que las ubicaciones que va a predecir la red neuronal artificial no sean obtenidas por un proceso de extrapolación, a continuación se describe donde deben ubicarse los pozos de entrenamiento:

- Los pozos de entrenamiento deben delinear el yacimiento.
- Espacios entre los pozos existentes.
- Donde se presenten fuertes cambios en el modelo petrofísico y en las propiedades PVT del fluido.

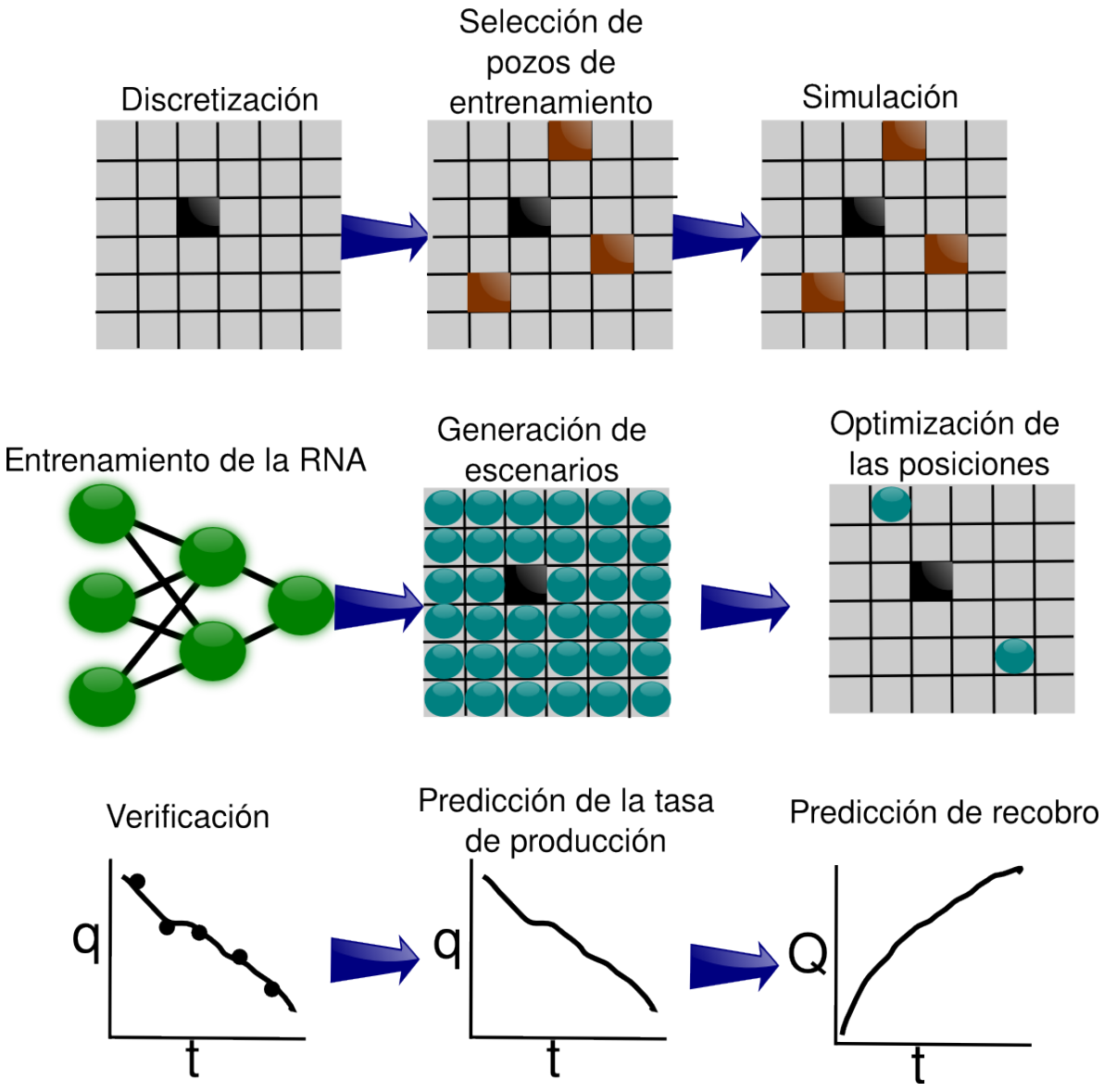


Figura 3.1: El proceso de Neurosimulación. *Key parameters controlling the performance of neuro-simulation applications in field development[6]*

- En discontinuidades internas.
- Cerca de pozos existentes (para efectos de interferencia).

Adicionalmente, se añaden las siguientes restricciones para la ubicación de pozos de entrenamiento:

- Ubicaciones en las fronteras.
- Ubicaciones adyacentes a bloques que contengan pozos existentes.

Esto se hace porque no se espera perforar pozos nuevos en esas ubicaciones y al removerlas el esfuerzo computacional disminuye.

Lo que los pozos de entrenamiento buscan es dar la mejor descripción del yacimiento e implícitamente proveen información acerca de propiedades como permeabilidad, porosidad, presión del yacimiento, saturación de fluidos a la red neuronal artificial.

El objetivo en este paso de la metodología es ubicar el mínimo número de pozos de entrenamiento con los cuales sea posible entrenar la red neuronal artificial y que ésta sea capaz de realizar predicciones precisas. Se debe tener en cuenta que un gran número de pozos de entrenamiento representa un esfuerzo computacional alto y un tiempo de simulación alto, y un número muy reducido de pozos de entrenamiento no obtienen datos suficientes para una buena descripción del yacimiento, llevando a un mal entrenamiento de la red neuronal artificial.

Una vez se hayan seleccionado los pozos de entrenamiento, se generan todas las combinaciones posibles entre todos los pozos de entrenamiento, y luego se filtran, permitiendo sólo un pozo en los límites del yacimiento, siendo las combinaciones permitidas:

- Todos los pozos dentro de los límites del yacimiento.
- Un pozo ubicado a lo largo de los límites y el resto de pozos dentro de los límites.

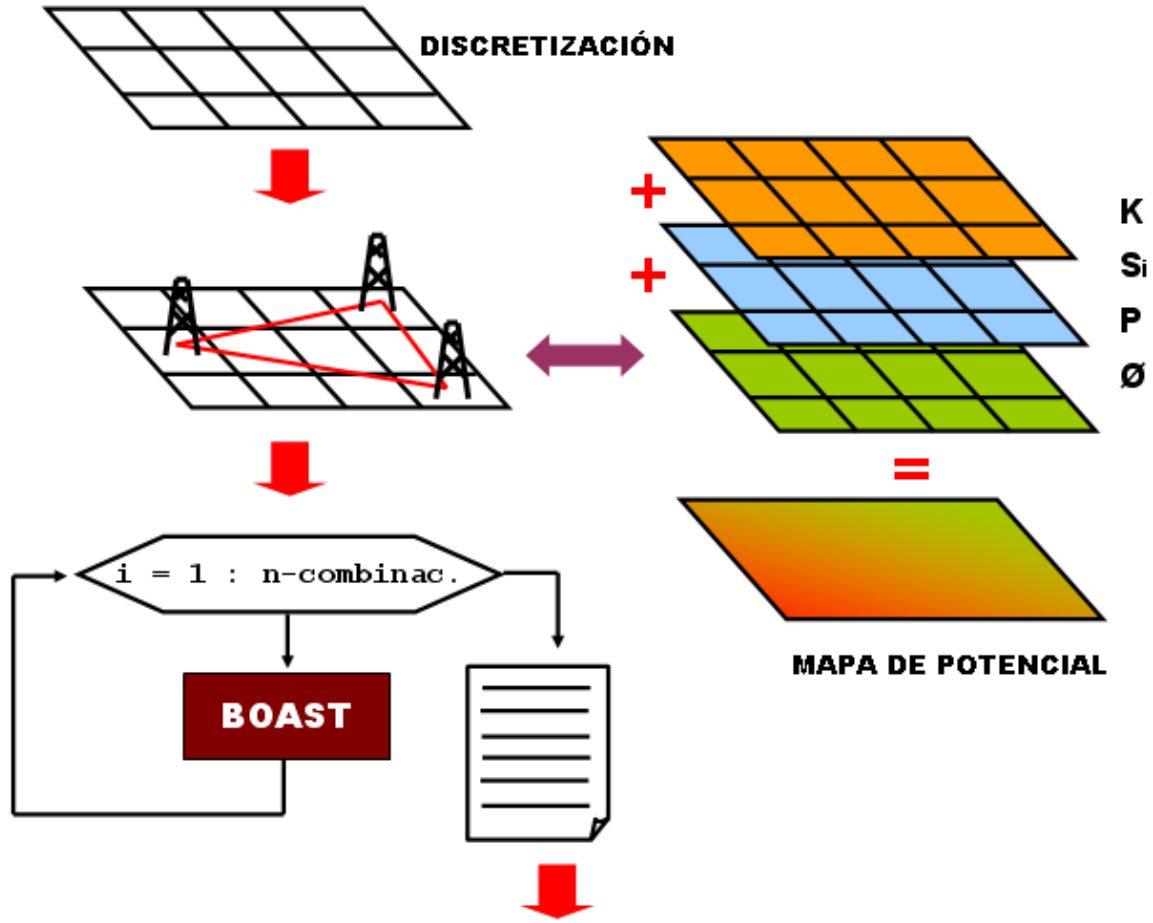


Figura 3.2: Metodología detallada 1.

Cada una de las combinaciones resultantes son *simuladas* usando el simulador numérico de yacimientos, obteniéndose los datos de producción para luego entrenar la red neuronal artificial. Los tres pasos anteriormente descritos se muestran en la Figura 3.2.

Con los datos obtenidos de la simulación numérica de yacimientos *la red neuronal es entrenada*, antes de que estos datos ingresen a la red neuronal artificial deben ser normalizados. El proceso de entrenamiento consiste en el ajuste de los pesos sinápticos (w) de cada neurona en función del error entre los datos obtenidos por la red y los datos de entrenamiento, esto se hace a través de épocas (iteraciones)

hasta que se alcance un valor permitido de error. Este tipo de entrenamiento es típico en las redes neuronales de propagación hacia adelante, utilizadas en este paso, con tres capas (1 entrada, 1 oculta, 1 salida) y algoritmo de entrenamiento back-propagation.

El número y cuales datos de entrada deben ingresarse a la red neuronal dependen del caso estudiado, pero de forma general son los siguientes:

- Posición X Y de los pozos nuevos.
- Tiempo al cual se quiere predecir la producción acumulada.
- Expresiones geométricas.
- Enlaces funcionales.

Las *expresiones geométricas* proveen información acerca de la dependencia geométrica de la interferencia entre pozos y los efectos de los límites, estas pueden ser las distancias de los pozos nuevos a los pozos viejos, el perímetro, o la distancia entre pozos nuevos, en casos heterogéneos se debe incluir la variable heterogénea como por ejemplo la porosidad, o la permeabilidad.

Los *enlaces funcionales* son funciones matemáticas de las entradas básicas tales como las coordenadas y el tiempo, y deben ser incluidas en la forma de funciones cuadráticas y funciones recíprocas. Estas funciones incrementan la sensibilidad de la red neuronal artificial a pequeños cambios en los datos de entrada. Mientras la función cuadrática amplifica pequeños cambios en los datos de entrada, la función recíproca es de ayuda en la amplificación de valores muy pequeños en los datos de entrada a valores más grandes para propósitos de identificación.

Adicionalmente puede ser necesario la inclusión de valores que dan cierto nivel de experiencia a la red neuronal artificial, como por ejemplo un factor de expectativa (Prueba 5.2) en el cual a cada celda se le asigna un valor dependiendo de que tanto se espera ubicar un pozo nuevo en dicha posición, estos valores pueden depender

de variables específicas del yacimiento estudiado, y su uso es propio para cada caso.

Una vez la red neuronal esté entrenada se *generan las ubicaciones de los pozos que van a ser evaluadas*. Si se toman todas las combinaciones posibles de las ubicaciones permitidas el número es significativamente alto, por ejemplo, para ubicar 3 pozos nuevos con 300 ubicaciones posibles el resultado son 4.455.100 (Ecuación 3.1) combinaciones no ordenadas. Para evitar que la red neuronal prediga todos esos escenarios, se toman las ubicaciones de cada grupo de pozos aleatoriamente usando un generador de números aleatorios, éste genera números aleatorios uniformemente distribuidos entre $[0, 1]$ que lego son escalados linealmente a las dimensiones del campo, por cada pozo se generan dos números aleatorios (coordenadas X Y), las combinaciones aleatorias obtenidas se filtran usando las mismas restricciones usadas para la ubicación de pozos de entrenamiento.

$$\binom{300}{3} = 4455100 \tag{3.1}$$

Cada combinación del grupo de posiciones a evaluar recién generado es evaluado por la red neuronal artificial, la red neuronal como resultado a cada combinación ingresada devuelve el valor de los datos de producción normalizados para cada pozo, es por esto que a cada resultado se le aplica un proceso de "desnormalización" y las configuraciones son ordenadas en base a la mayor producción, la mejor combinación es *verificada* usando el simulador numérico de yacimientos, calculando el error entre el resultado predicho por la red neuronal y el simulador de yacimientos, una vez los resultados de la red neuronal sean validados, las tasas de declinación finales y el recobro acumulado son calculados, en las Figuras 3.3 y 3.4 se muestra los pasos descritos.

3.2. METODOLOGÍA PROPUESTA

En esta sección se exponen los cambios hechos a la metodología descrita en la sección 3.1, los tres primeros pasos, discretización, selección de pozos de entrenamiento y simulación, permanecen iguales como se muestra en la Figura 3.5.

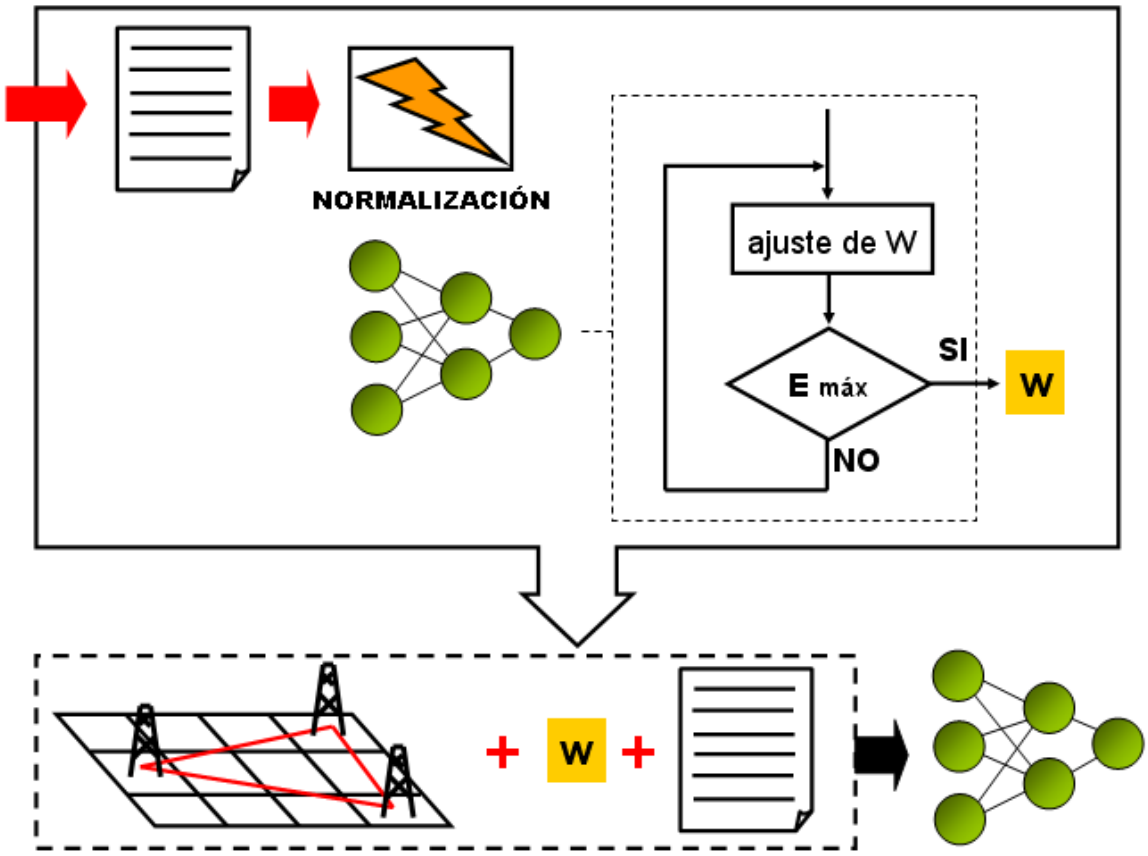


Figura 3.3: Metodología detallada 2.

Como se puede observar en la Figura 3.6 en la metodología propuesta se incluye un grupo de pozos de validación para evitar sobre-entrenar la red neuronal artificial, los pozos de validación son un grupo de pozos de los cuales se obtiene los mismos datos que los pozos de entrenamiento, pero en lugar de ser usados para entrenar la red neuronal artificial, esta periódicamente los evalúa, el error es calculado pero los pesos sinápticos no son cambiados, siendo su propósito probar la respuesta de la red neuronal a entradas que no ha visto durante el entrenamiento.

Para la generación de los pozos de validación se deben seguir las mismas consideraciones que para la elección de pozos de entrenamiento, teniendo en cuenta de no tomar pozos de entrenamiento como pozos de validación.

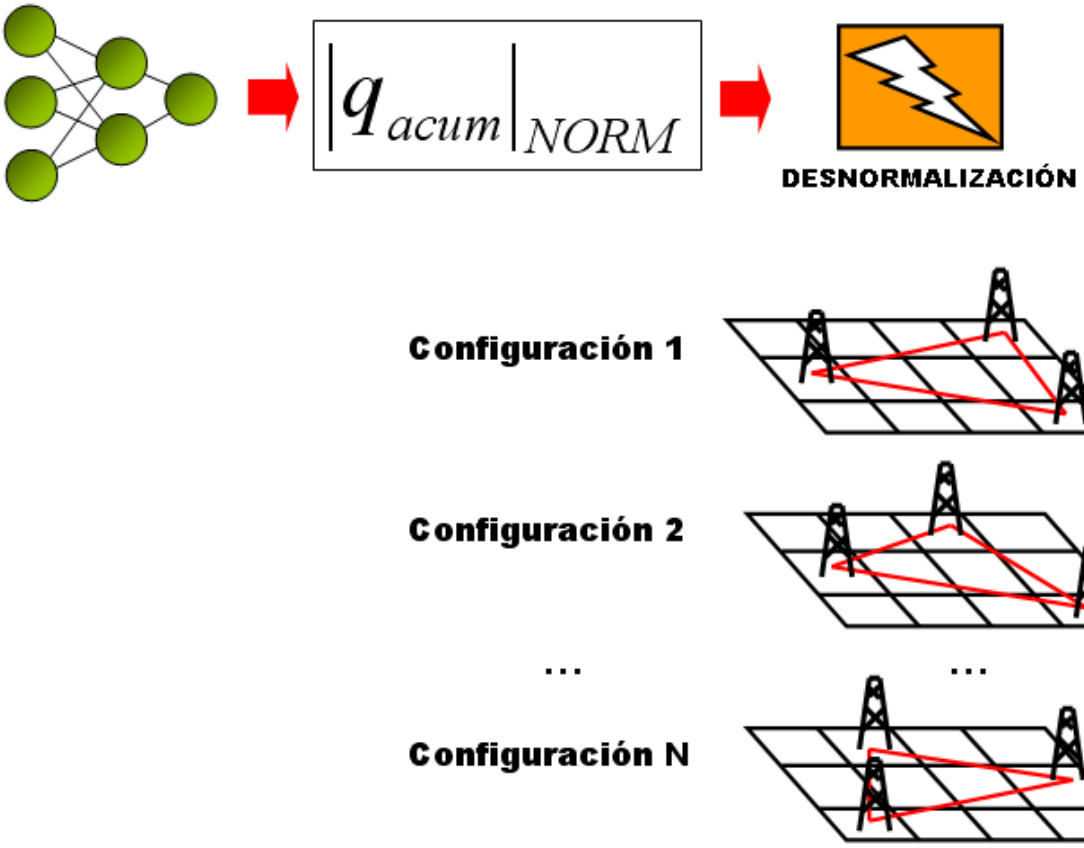


Figura 3.4: Metodología detallada 3.

En la Figura 3.6 también se muestra dos aspectos importantes que no son comentados en la metodología convencional, estos son la normalización de datos de entrada de la red neuronal artificial y la desnormalización de datos de salida de la red neuronal artificial, a continuación se muestran las propuestas hechas para realizar estos dos procesos.

3.2.1. Normalización de los datos de entrada

A continuación se muestra la propuesta para la normalización de los datos de entrada:

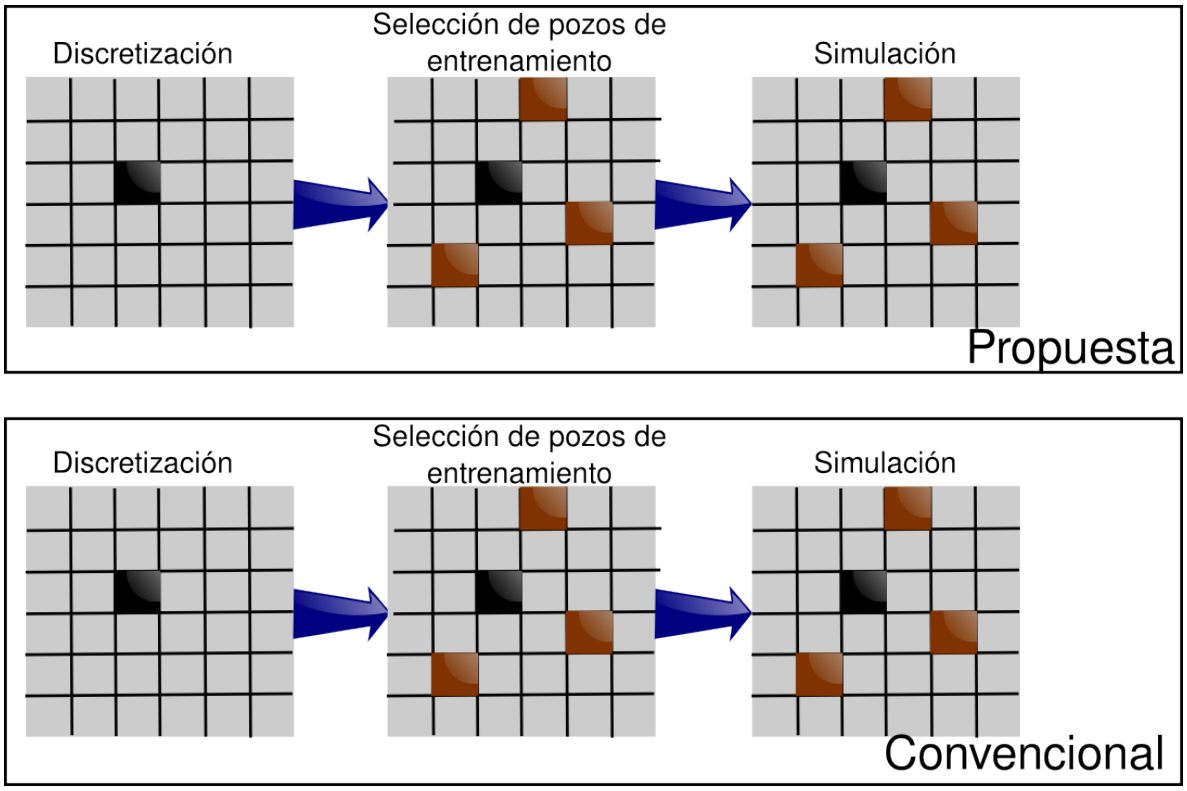


Figura 3.5: Metodología Propuesta vs. Convencional, primeros pasos.

- Para las coordenadas

$$\frac{x}{distancia\ x\ max} \tag{3.2}$$

- Para el tiempo

$$\frac{tiempo\ producción\ pozos\ nuevos}{tiempo\ total\ producción} \tag{3.3}$$

- En general

$$\frac{valor}{valor\ máximo} \tag{3.4}$$

3.2.2. Normalización de los datos de salida

Para los caudales acumulados se propuso una normalización en base a la producción total acumulada a cada tiempo

$$Producción\ normalizada\ Pozo\ j_{tiempo\ i} = \frac{Producción\ Pozo\ j_{tiempo\ i}}{Mayor\ Producción\ total\ pozos\ nuevos_{tiempo\ i}} \tag{3.5}$$

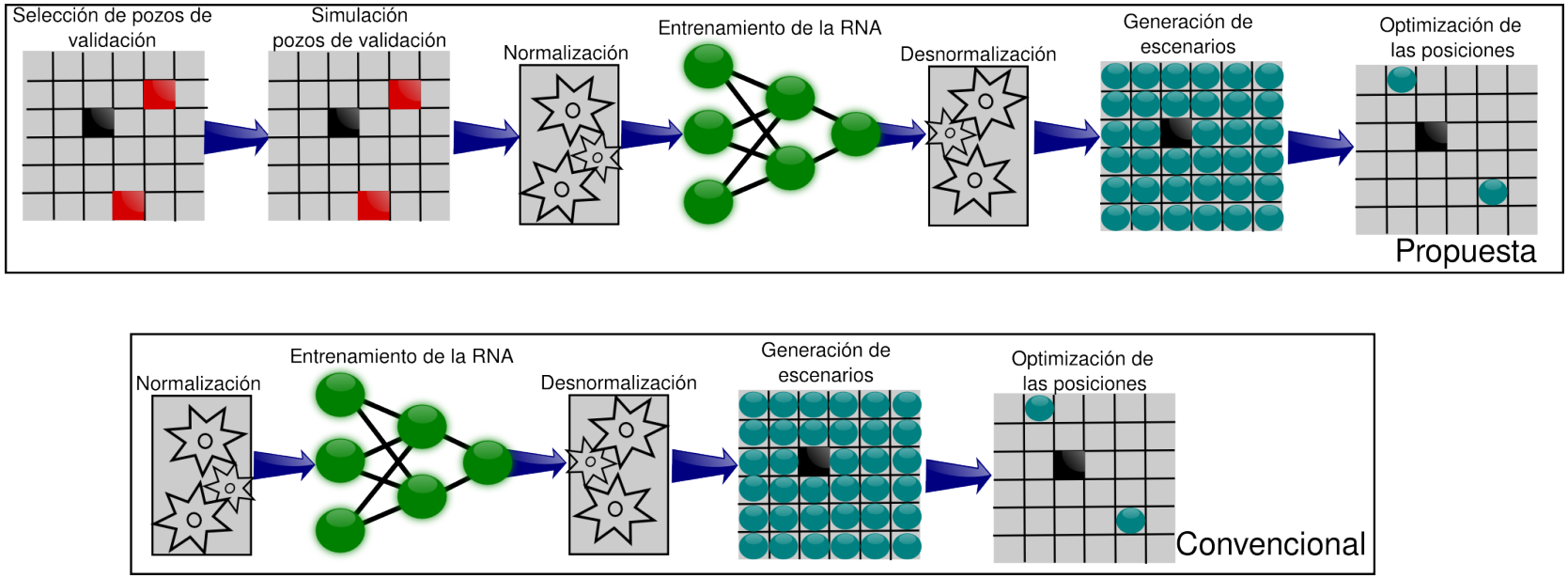


Figura 3.6: Metodología Propuesta vs. Convencional, pasos intermedios.

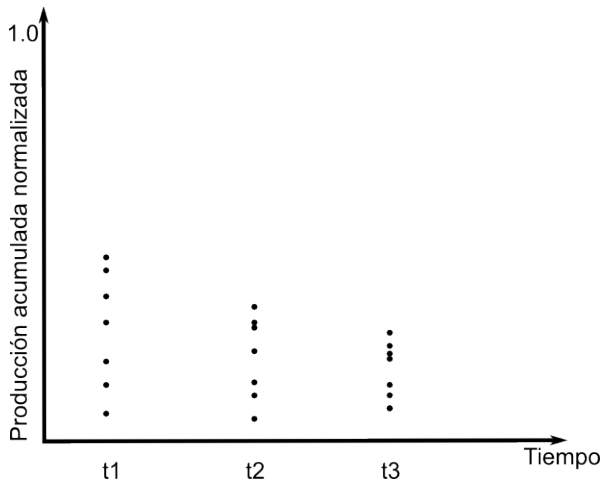


Figura 3.7: Producción acumulada normalizada usando Ecuación 3.5.

Sin embargo, al ser los datos de producción total acumulada a cada tiempo valores mucho mayores que los valores de producción acumulada de cada pozo, los valores no cubren por completo el intervalo $[0, 1]$, por lo tanto la red neuronal pierde sensibilidad ante pequeños cambios en la producción acumulada de cada pozo; además, a medida que el tiempo aumenta, la producción total acumulada es mucho mayor que la producción acumulada de cada pozo lo que se ve reflejado en un valor normalizado menor que a tiempos menores. Lo anterior hace que no exista una tendencia a través del tiempo para que la red neuronal pueda asociar los datos y llegar a una predicción precisa, la Figura 3.7 muestra los datos normalizados con la Ecuación 3.5.

Debido a lo anterior, se propone una forma diferente de normalizar los datos de producción acumulada mostrada en la Ecuación 3.6, en lugar de utilizar la producción total acumulada a cada tiempo, se usa el valor del pozo que tenga la mayor producción acumulada al tiempo final mas el $10\,\%$ de dicho valor. Este valor es mucho menor que el tomado en la Ecuación 3.5, por lo tanto los valores normalizados estarán mejor distribuidos en el intervalo $[0, 1]$, además como se puede observar en la Figura 3.8, existe una tendencia de los datos a medida que pasa el tiempo lo que facilita la asociación de los datos por parte de la red neuronal artificial. La adición del $10\,\%$ se hace porque existe la posibilidad de encontrar valores mayores a los calculados con los pozos de entrenamiento, y el factor del $10\,\%$ asegura que el valor

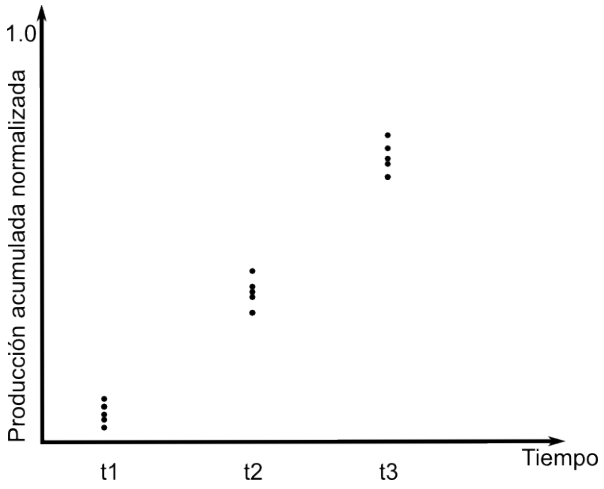


Figura 3.8: Producción acumulada normalizada usando Ecuación 3.6.

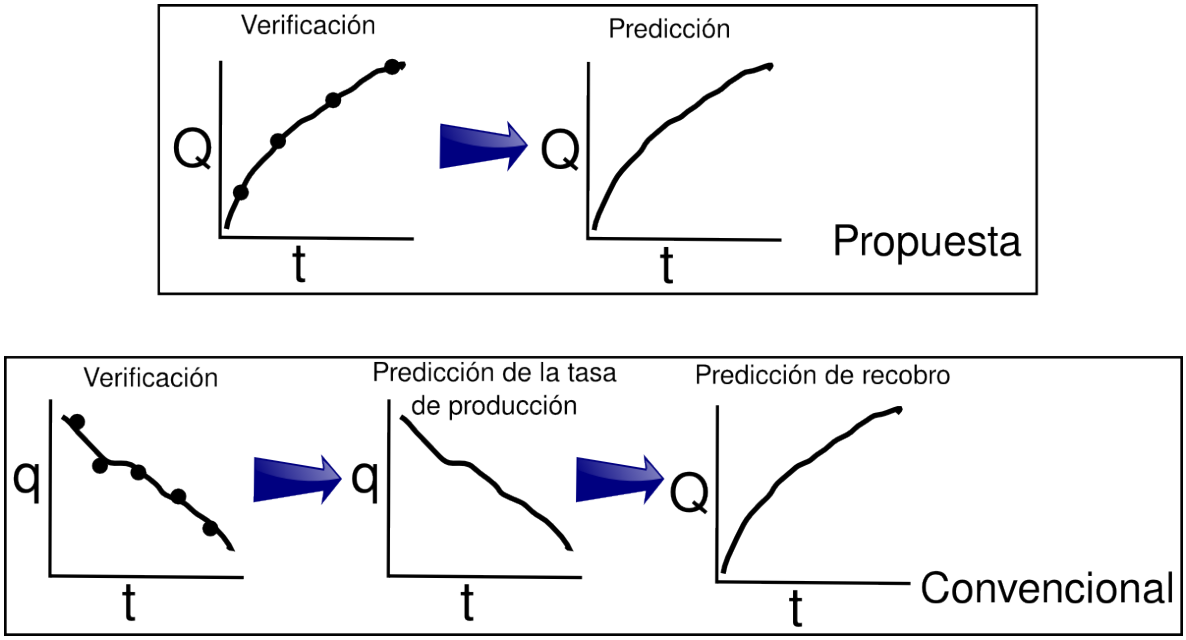


Figura 3.9: Metodología Propuesta vs. Convencional, últimos pasos.

normalizado se encuentre en el intervalo $[0, 1]$.

$$Producción\ normalizada\ Pozo\ j_{tiempo\ i} = \frac{Producción\ Pozo\ j_{tiempo\ i}}{Mayor\ Producción\ Pozo_{tiempo\ final} * 1{,}1} \tag{3.6}$$

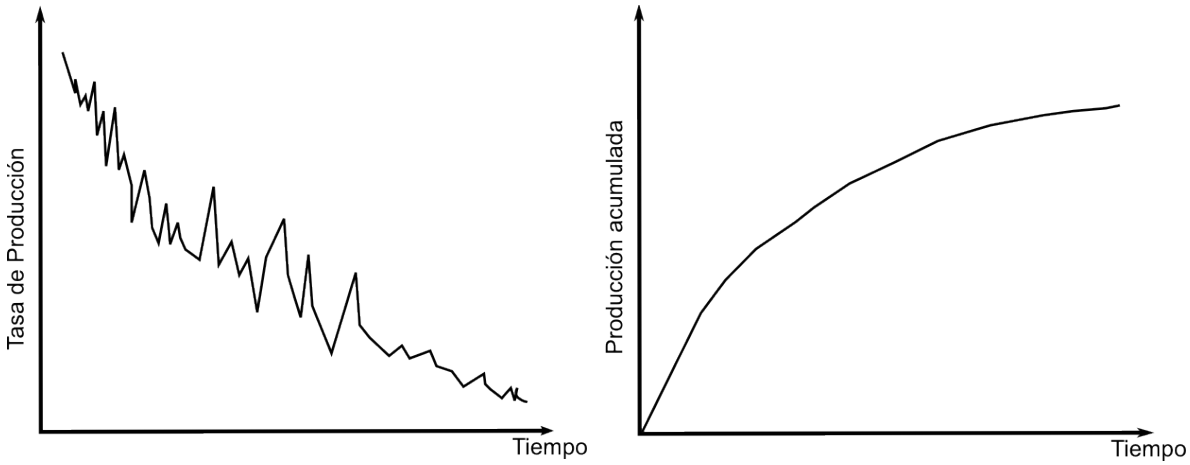


Figura 3.10: Cambio en los datos de salida.

En la Figura 3.9 se muestra el cambio que se hizo respecto a los datos de salida de la red neuronal artificial, en lugar de predecir la tasa de flujo de cada pozo, la red neuronal artificial predice la producción acumulada por cada pozo, esto se debe a que la tasa de flujo presenta un comportamiento oscilatorio al contrario de la producción acumulada, que presenta un comportamiento más estable (Figura 3.10), esto mejora la asociación de los datos de entrada-salida para una red de tipo propagación hacia adelante, tipo de red usada en el proyecto.

En la Figura 3.11 se muestra de forma general la metodología propuesta.

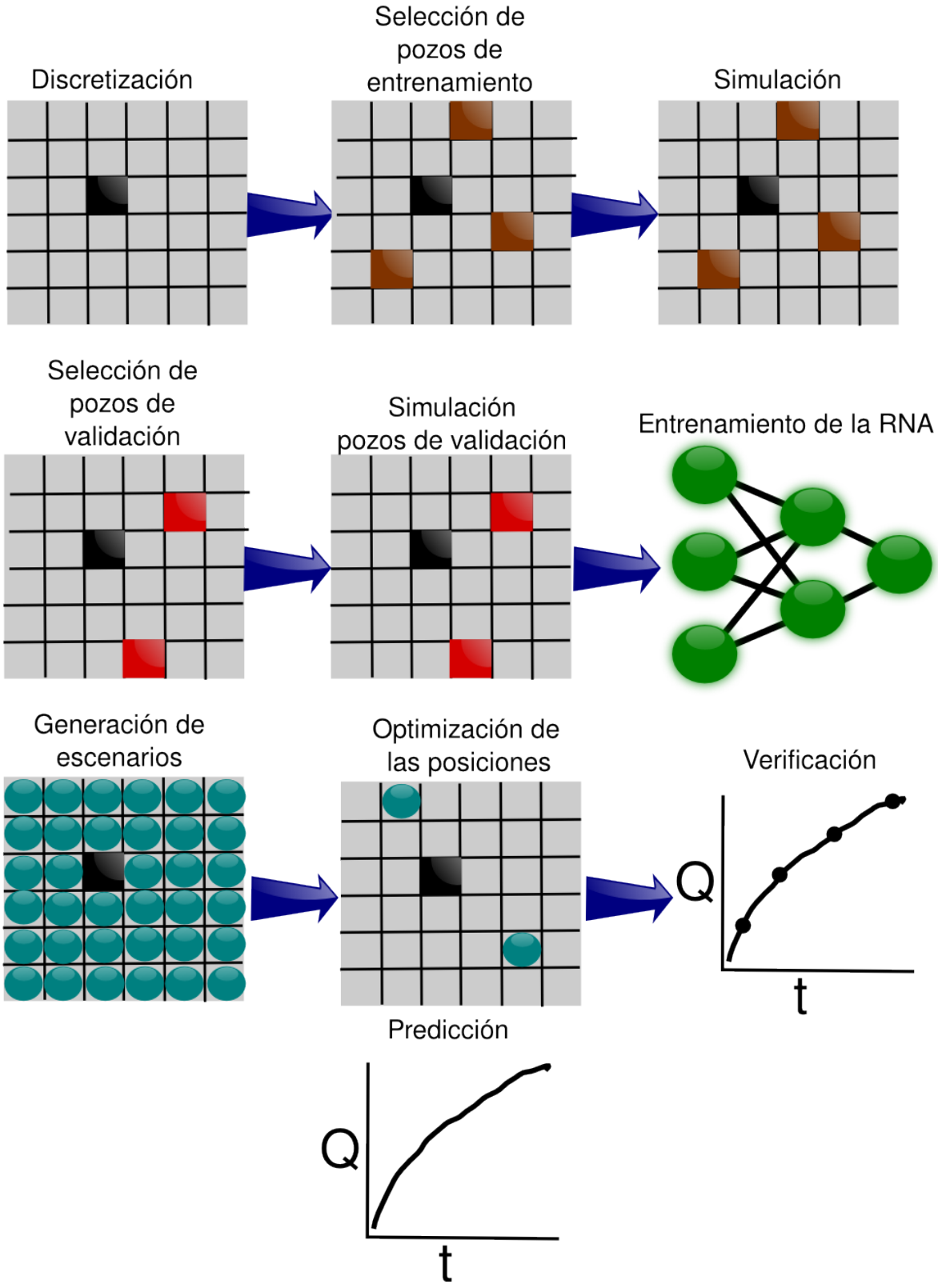


Figura 3.11: El proceso de Neurosimulación propuesto.

4 SOFTWARE

Para la realización de la herramienta software objetivo de este proyecto se tomó como base el uso de herramientas de software libre, en primer lugar por la libertad que eso implica tanto en su uso no dependiente de pago de licencias, lo que reduce significativamente el costo del proyecto, como por el hecho de que estas herramientas pueden reutilizarse dentro del proyecto y así se evita el "reinventar la rueda" y de esta forma agilizar el desarrollo de la herramienta sin disminuir la calidad del resultado final.

De la misma forma se utilizó un sistema operativo libre: *GNU/Linux*.

Dos herramientas necesarias para la construcción de la herramienta son un *simulador numérico de yacimientos* y un *simulador de redes neuronales artificiales*, en la siguiente sección se muestra la revisión hecha acerca del software existente y su utilización dentro del proyecto.

4.1. OPCIONES DE SOFTWARE LIBRE

4.1.1. Simulador numérico de yacimientos

Las alternativas libres de simuladores numéricos de yacimientos encontradas fueron el BOAST II[a] y sus derivados:

BOAST II

Herramienta de Simulación Aplicada a Petróleo Negro por sus siglas en inglés (Black Oil Applied Simulation Tool) es un simulador de tridimensional trifásico escrito en el

[a] `http://www.netl.doe.gov/technologies/oil-gas/software/simulat.html`

lenguaje de programación FORTRAN.

BOAST 3

Versión modificada de BOAST II contiene postprocesadores B3PLOT para graficar los datos de la simulación y COLORGRID para gráficos de la malla de simulación.

BOAST 98

Actualización visual, dinámica e interactiva del BOAST 3, interactúa con EdBOAST (editor de datos del yacimiento por medio de diálogos). Escrito en FORTRAN 90.

BOAST-NRF

Simulador de aceite negro para formaciones naturalmente fracturadas usando un modelo de doble porosidad, doble permeabilidad. Basado en BOAST.

WinB4D

Es un programa de tipo académico[9] basado en BOASTII, incluyendo mejoras como la realización de balance de materiales para un modelo de tanque, representación de pozos horizontales o desviados, y el cálculo de información geofísica del yacimiento.

Para este proyecto se usó el simulador de yacimientos WinB4D por ser el más actual y con mejores características de las opciones encontradas.

4.1.2. Simulador de redes neuronales artificiales

Aquí se listan los simuladores de redes neuronales libres:

Neura-Lab

[b]Simulador de redes neuronales desarrollado por la universidad de Guanajuato, está construido para el sistema operativo Windows, la versión actual es la 3.1. Esta versión soporta redes neuronales artificiales clásicas e introduce redes neuronales

[b] http://www.dicis.ugto.mx/profesores/sledesma/documentos/index.htm

en el dominio complejo (cada neurona tiene dos entradas y dos salidas una para la parte real y otra para la parte imaginaria). Esta opción no se usó debido a que no dispone de versión para sistemas GNU/Linux

Emergent

[c]Es un simulador de redes neuronales principalmente enfocado en la creación de modelos complejos y sofisticados del cerebro y procesos cognitivos, pero también puede ser usado para cualquier tarea en las cuales las redes neuronales artificiales sean adecuadas.

Stuttgart Neural Network Simulator

SNNS[d] Simulador de redes neuronales para maquinas de trabajo unix desarrollado por el Instituto para Sistemas Paralelos y Distribuidos de Alto Desempeño (IPVR) en la universidad de Stuttgart. En Julio del 2008 la licencia fue cambiada a GNU LGPL (software libre). SNNS está escrito alrededor de un núcleo (kernel) de simulación al cual se le pueden añadir funciones de activación, procedimentos de aprendizaje y funciones de salida escritas por el usuario. Además incluye las siguientes arquitecturas de redes neuronales y algoritmos de entrenamiento:

- Backpropagation para redes de propagación hacia adelante.
  - Vanilla backpropagation.
  - Backpropagation con término de momentum y eliminación de superficie plana.
  - Backpropagation por baches (batch backpropagation).
- Counterpropagation.
- Quickprop.
- Backpercolation 1.
- RProp.

[c] http://grey.colorado.edu/emergent/index.php/Main_Page
[d] http://www.ra.cs.uni-tuebingen.de/SNNS/

- Funciones de base radial generalizadas (RBF).
- ART1.
- ART2.
- ARTMAP.
- Correlación de cascada.
- Correlación de cascada recurrente.
- LVQ Dinámico.
- Backpropagation a través del tiempo (BPTT) para redes recurrentes.
- Quickprop a través del tiempo (para redes recurrentes).
- Mapas auto-organizados (mapas de Kohonen).
- TDNN (redes de tiempo-retraso) con Backpropagation.
- Redes de Jordan.
- Redes de Elman y Redes de Elman jerárquicas extendidas.
- Memoria asociativa.

El simulador de redes neuronales artificiales usado en este proyecto es SNNS.

4.1.3. Lenguaje de programación

El lenguaje de programación Python[e] fue usado para el desarrollo de la herramienta software. Python es un lenguaje de programación interpretado[f] creado por Guido van Rossum en el año de 1991. Es desarrollado como un proyecto de código abierto.

Python al ser un lenguaje interpretado ahorra tiempo considerable en el desarrollo del programa, ya que no es necesario compilar ni enlazar.

Dentro de las características de Python se encuentran:

[e] `www.python.org`

[f] Está diseñado para ser ejecutado por medio de un intérprete, no es necesario compilar el código fuente.

- Lenguaje de tipado dinámico, es decir, una variable puede tomar valores de distinto tipo en distintos momentos.
- Es fácil de aprender debido a su sintaxis limpia.
- Chequeos de construcción a la hora de correr el programa ayuda a detectar errores y disminuir el tiempo de desarrollo.
- Programación con estructuras de datos anidadas y heterogéneas es fácil.
- Multiparadigma, programación orientada a objetos, programación estructurada y programación funcional.
- Soporte para cómputo científico.
- Integración con lenguajes como C, C++, FORTRAN y Java está muy bien soportada.
- Fácil programación de interfaces gráficas de usuario (GUI).
- Fácil reutilización de código.

4.2. FLUJO DE TRABAJO

En la Figura 4.1 se muestra el flujo de trabajo seguido mientras se aplica la metodología de neurosimulación, se pueden observar la interacción de los archivos de entrada, los scripts escritos en Python, y los archivos generados.

Estos son scripts de Python, escritos por el autor del proyecto, estos leen los archivos de entrada, generan archivos de comunicación, convierten datos a formatos legibles por los simuladores, automatizan las corridas del simulador numérico de yacimientos, generan combinaciones aleatorias para ser evaluadas por la red neuronal, filtran los datos, normalizan y desnormalizan datos de entrada y salida de la red neuronal, ordenan los resultados, generan los archivos para visualización, y verificación de los resultados, crean la interfaz gráfica de usuario entre otras. En total se escribieron 22 scripts y se utilizaron un poco más de 4400 líneas de código.

Esta imagen representa a los archivos de entrada.

Estos son archivos de salida de los scripts, estos pueden a su vez ser los archivos de entrada de otros scripts o del simulador de yacimientos o de redes neuronales artificiales o del visualizador de resultados.

Esta imagen representa la interfaz gráfica de usuario, por medio de la cual el usuario interactúa con el programa y por medio de la cual el programa presenta los resultados.

Este es el usuario del programa.

4.3. DISEÑO DE ARQUITECTURA SOFTWARE

La metodología de desarrollo de software usada en este proyecto es el desarrollo rápido de aplicaciones o RAD (acrónimo en inglés de *Rapid Application Development*), esta es una metodología que usa un mínimo de planeamiento en favor de un prototipado rápido. El “planeamiento” de software desarrollado usando RAD es

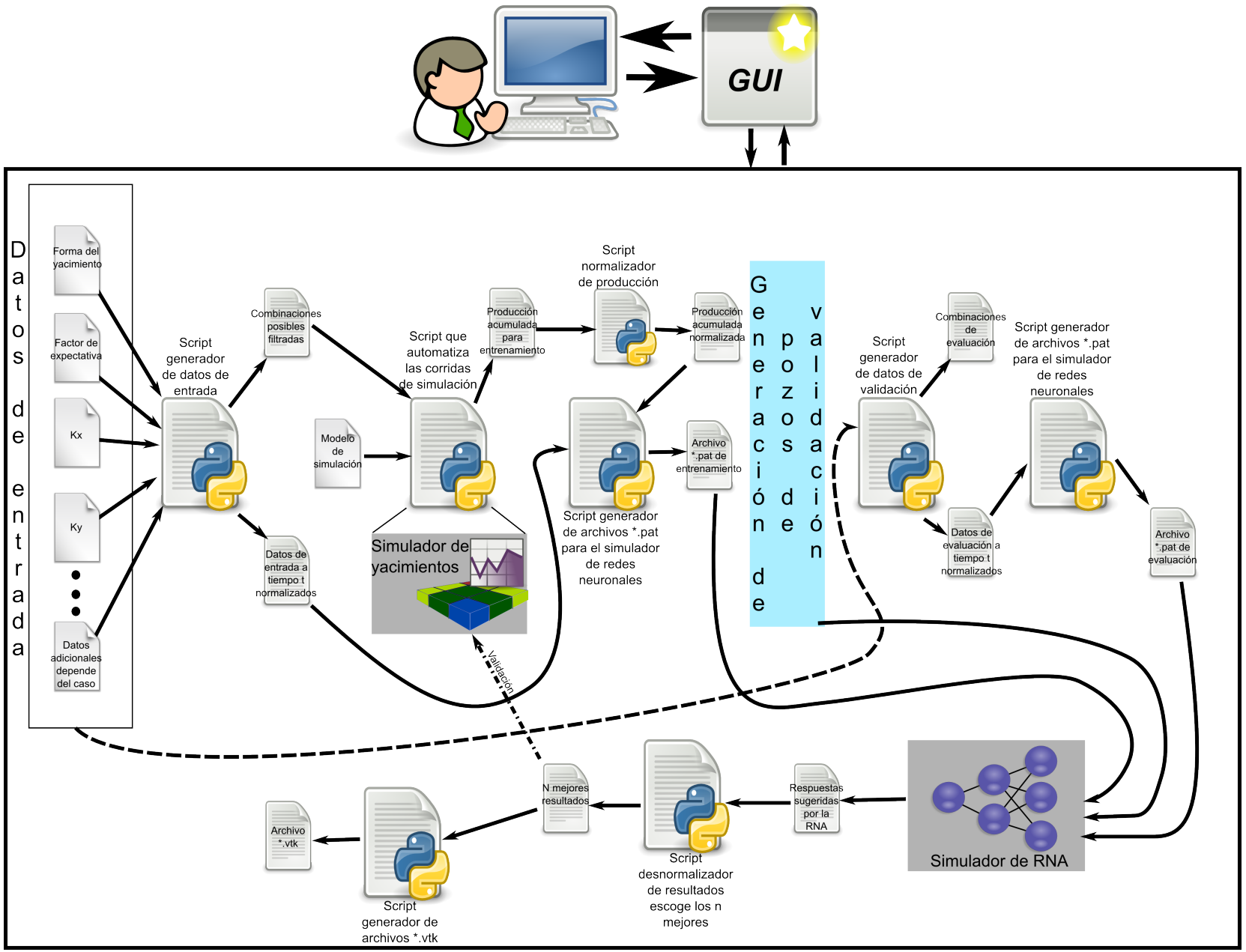


Figura 4.1: Flujo de trabajo

intercalado con la escritura del mismo. La ausencia de un extenso pre-planeamiento generalmente permite que el software sea escrito mucho más rápido, y se hace más fácil cambiar los requerimientos.

Al principio del desarrollo, se empezó a aplicar la metodología de forma “manual”, generando los archivos necesarios para la comunicación del simulador numérico de yacimientos y el simulador de redes neuronales, y visualizando los resultados con *GNUplot*[g]. Para aplicar la metodología se requirió mucho tiempo debido a la forma “manual” de la aplicación.

A medida que le proyecto avanzaba se escribieron los scripts de automatización, esto disminuyó considerablemente el tiempo de aplicación de la metodología en especial cuando se construyó el script de automatización de la corrida del simulador numérico de yacimientos, además, los scripts regularmente fueron revisados y mejorados, sin embargo, no estaban enlazados entre sí, por lo que su ejecución era de forma individual.

Una vez todas las etapas fueron automatizadas, se escribió un script que ejecuta cada una de las etapas de forma automática, lo cual redujo aún mas los tiempos de ejecución, también se añadió un script que genera un archivo *vtk*[h] lo cual permite visualizar los resultados en diversos visualizadores gráficos como Paraview[i] o Mayavi2[j].

Al final del proceso se construyó una interfaz de usuario simple la cual recoge los datos de entrada y ejecuta todo el proceso, la interfaz de usuario fue escrita con WxWidgets[k] API para la escritura de interfaces gráficas de usuario (GUI), escrita en el lenguaje de programación C++, multiplataforma, fácil de usar y libre, junto con los

[g] `http://www.gnuplot.info`
[h] Sistema software para gráficos computarizados en 3D `http://www.vtk.org/`
[i] `http://www.paraview.org/`
[j] `http://code.enthought.com/projects/mayavi/`
[k] `http://www.wxwidgets.org/about/`

enlaces (*bindings*) para el lenguaje de programación Python llamados WxPython[l], y el constructor de interfaces gráficas WxGlade[m].

En las Tablas 4.1 y 4.2 se muestran los scritps escritos junto con una pequeña descripción de su función.

La herramienta software fue construida buscando que fuera flexible, por eso su estructura es de "quitar y poner", es decir, cada uno de los módulos puede ser removido y cambiado por otro con pocas modificaciones en los códigos de los scripts, esto permite por ejemplo cambiar el simulador numérico de yacimientos WinB4D por otra opción para mejorar la predicción de la producción acumulada para ciertos tipos de yacimientos, o cambiar el simulador de redes neuronales artificiales, entre otras, esto hace que la herramienta tenga un mayor rango de aplicación, también como especializarse para un uso determinado con pocas modificaciones. En la Figura 4.2 se muestran los módulos "cambiables" de la herramienta software.

4.3.1. Formato de datos de entrada

Los datos de entrada para poder ejecutar la herramienta software deben almacenarse en un grupo de archivos de entrada, como lo muestra la Figura 4.1, a continuación se muestra el formato de datos como deben ser creados cada archivo.

Archivo con forma del yacimiento, entrenamiento

Este archivo contiene información acerca de la geometría del yacimiento, los pozos existentes, los pozos de entrenamiento y el área permitida para ubicar los pozos. Esta información se almacena por medio de una matriz, donde cada ubicación i,j de la matriz representa las coordenadas en la malla del yacimiento, cada componente de la matriz tendrá un código dependiendo si en dicha ubicación hay un pozo existente, o de entrenamiento, o si no pertenece al yacimiento. Los códigos se muestran a continuación:

- 0 Si no se permite ubicar pozos nuevos.

[l] `http://wxpython.org`
[m] `http://wxglade.sourceforge.net/`

Tabla 4.1: Scripts escritos en Python.

Nombre	Descripción
inter.py	Crea la interfaz gráfica de usuario
main.py	Llama a otros scritps
valid.py	Realiza la validación de los resultados con el simulador numérico de yacimientos
area_caudry.py	Contiene funciones usadas por el script sorteo_entre_heter.py
combina_recur.py	Crea una combinatoria recursiva crea grupos de a numero_pozos_nuevos a partir una lista
sorteo_entre_heter.py	Genera combinaciones de cualquier numero de pozos nuevos de entrenamiento genera una archivo con datos normalizados listos para construir un archivo .pat para entrenar una red neuronal artificial con SNNS
output_data.py	Script para mover el mouse y el teclado para automatizar la corrida de WinB4D. Lee un archivo de entrada con las coordenadas de los pozos y secuencialmente cambia el archivo wtemp.dat para cada corrida del WinB4D
normalizacion_q.py	Script que toma el archivo de caudales acumulados y los normaliza
RNA_entr.py	Script que recibe varios archivos y crea el archivo .pat para SNNS
area_caudry.py (val)	Contiene funciones usadas por el script sorteo_entre_heter.py (pozos de validación)
combina_recur.py (val)	Crea una combinatoria recursiva crea grupos de a numero_pozos_nuevos a partir una lista (pozos de validación)
sorteo_entre_heter.py (val)	Genera combinaciones de cualquier numero de pozos nuevos de entrenamiento genera una archivo con datos normalizados listos para construir un archivo .pat para entrenar una red neuronal artificial con SNNS (pozos de validación)
output_data.py (val)	Script para mover el mouse y el teclado para automatizar la corrida de WinB4D. Lee un archivo de entrada con las coordenadas de los pozos y secuencialmente cambia el archivo wtemp.dat para cada corrida del WinB4D (pozos de validación)
normalizacion_q.py (val)	Script que toma el archivo de caudales acumulados y los normaliza (pozos de validación)

Tabla 4.2: Scripts escritos en Python. (cont.)

Nombre	Descripción
RNA_entr.py (val)	Script que recibe varios archivos y crea el archivo .pat para SNNS (pozos de validación)
area_caudry.py (eval)	Contiene funciones usadas por el script sorteo_heter_eval.py (combinaciones de evaluación)
sorteo_heter_eval	Script que genera combinaciones aleatorias para cualquier numero de pozos nuevos para ser evaluados por la RNA, genera una archivo con datos normalizados listos para construir un archivo .pat leído por SNNS
RNA_eval.py	Scitp que recibe varios archivos y crea el archivo .pat para SNNS (esto es para evaluacion)
desn-ech.py	Script para desnormalizar y conocer los n caudales mas altos y los n caudales mas bajos de un archivo de salida de SNNS
res_vtk.py	Script que recibe los mejores resultados de la red y genera un archivo .vtk para visualización
visual.py	Script que genera el archivo .vtk del grid del modelo de simulación y ejecuta a Mayavi2 como visualizador
best_q_x_train.py	Toma un archivo con caudales para entrenar la red neuronal y obtiene los cinco mejores resultados
Líneas de código	4400

- 1 Si se permite ubicar un pozo nuevo.
- 2 Donde hay pozos existentes.
- 3 Donde hay pozos de entrenamiento.

Un ejemplo de este archivo se muestra en la Figura 4.3, en donde se resaltan los valores anteriormente nombrados.

Archivo con forma del yacimiento, validación

La única diferencia de este archivo con el anterior es que en lugar de contener las ubicaciones de los pozos de entrenamiento, contiene las ubicaciones de los pozos de validación. Por lo tanto el código 3 representa donde hay pozos de validación.

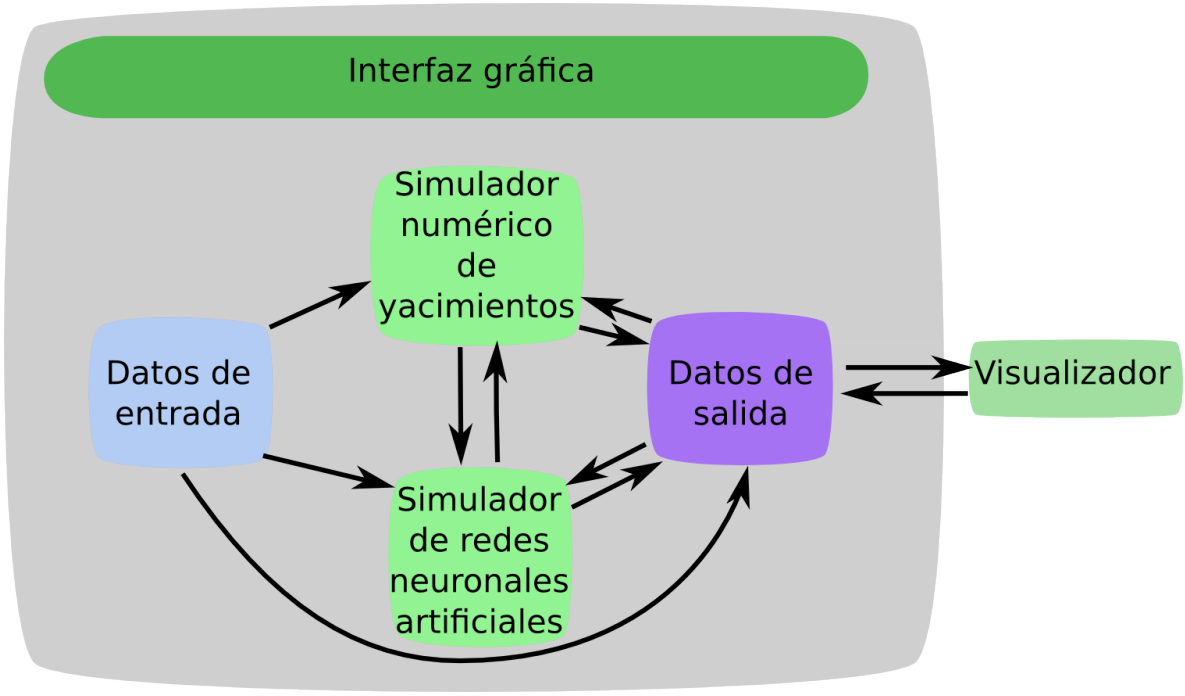


Figura 4.2: Modularidad de la herramienta software.

Archivos de propiedades

Estos archivos contienen cada uno una propiedad o valor usado en la entrada de la red neuronal artificial, por ejemplo, permeabilidad, factor de expectativa (Sección 5.2), si el caso requiere propiedades o valores adicionales se debe generar su archivo respectivo. Los datos son almacenados por medio de una matriz donde cada posición almacena el valor del bloque representado.

Cabe resaltar que los archivos mencionados deben ser creados por el usuario antes de utilizar la herramienta software.

Datos adicionales

Adicionalmente a los archivos mencionados anteriormente, hay que ingresar los siguientes datos:

- Número de pozos nuevos.

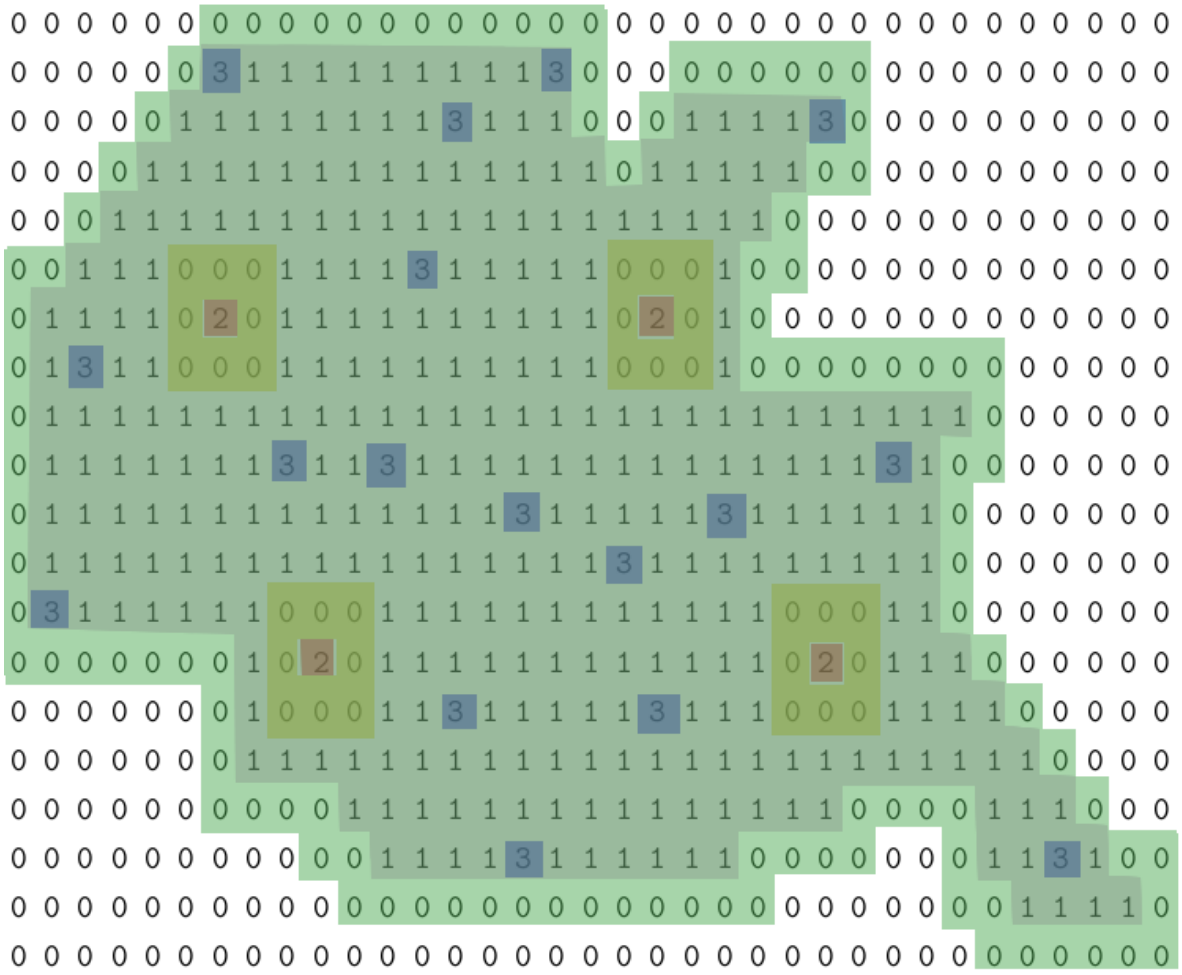


Figura 4.3: Ejemplo de archivo con forma del yacimiento.

- Tiempo total de simulación.
- Tiempos parciales para entrenar la red neuronal artificial.
- Número de neuronas de entrada.
- Número de neuronas ocultas.
- Número de neuronas de salida.
- Número de combinaciones aleatorias a evaluar.

Los requisitos para poder correr la herramienta software se presentan en el Anexo C.

4.3.2. Fortalezas y debilidades de la herramienta software

Fortalezas:

- Una vez la red es entrenada el poder de cómputo requerido es muy bajo.
- La herramienta es flexible al permitir la posibilidad de usar diferentes simuladores tanto de yacimientos como de redes neuronales artificiales.
- Fácil portabilidad.
- Gran cantidad de escenarios evaluados en cortos periodos de tiempo.

Debilidades:

- Solo se pueden ubicar 3 pozos nuevos.
- Solo maneja modelos en dos dimensiones.
- Solo maneja modelos monofásicos.

4.3.3. Aplicación de la herramienta

A continuación se describen los usos de la herramienta software desarrollada.

- Yacimientos de gas/aceite en donde a producción de las otras dos fases sea muy baja.
- Yacimientos tridimensionales que puedan ser tratados como un conjunto de capas bidimensionales, a cada capa se le aplica la metodología y luego se ubican los pozos nuevos indicando las capas a ser completadas.
- Yacimientos delgados o que su relación área espesor sea alta.

5 APLICACIONES

En este capítulo de muestra la aplicación de la metodología a diferentes casos. Debido a que la metodología es experimental y no se conocía cual pudiera ser su comportamiento, todos los casos fueron hechos en base a un yacimiento de gas, esto debido a que un yacimiento de gas es menos complejo que cualquier yacimiento de aceite, se tiene mayor control sobre las variables y permite una mayor atención a el desarrollo de la metodología de neurosimulación.

En todos los casos se ubican tres pozos nuevos, esto porque generalmente un programa de perforación no excede la cantidad de 4-5 pozos por etapa (simultáneamente) por lo tanto tres pozos es un término medio.

Todos los caso aplicados tratan de modelos de yacimientos en dos dimensiones, esto se hizo para reducir complejidad y centrar la atención sobre la metodología.

Los casos comienzan con el modelo más sencillo, al cual se le añaden cuidadosamente variaciones para poder evaluar la respuesta de la metodología a dichas variaciones. El primer caso consiste de un yacimiento de gas cuadrado, homogéneo con tres pozos produciendo, en el cual se estudió la sensibilidad de la red neuronal artificial a diferentes números de neuronas de entrada. El segundo caso añade variación en la permeabilidad. El tercer caso evalúa el desempeño de la metodología sobre un dominio que no es cuadrado. El cuarto caso añade la ubicación de seis pozos nuevos en dos etapas de perforación. La última prueba fue realizada aplicando todas las modificaciones explicadas en la metodología propuesta. Adicionalmente se muestra una serie de figuras en donde se observa el uso de la herramienta software con el último caso.

Tabla 5.1: Composición del gas para la prueba 5.1.

Metano	88.0 %
Etano	1.5 %
Propano	10.5 %

Tabla 5.2: Propiedades del yacimiento para la prueba 5.1.

Presión inicial del yacimiento	6200 psia
Temperatura del yacimiento	144 °F
Permeabilidad del yacimiento	25 md
Porosidad	25 %
$\Delta x = \Delta y$	75 ft
Espesor del yacimiento	75 ft
Radio del wellbore	0.25 ft

5.1. YACIMIENTO DE GAS HOMOGÉNEO Y CON FORMA REGULAR

Esta es la primera prueba realizada, consiste en un yacimiento de gas de composición mostrada en la Tabla **??** con propiedades mostradas en la Tabla 5.2. Se ha discretizado en una malla de 20x20 celdas Figura 5.1, existen 3 pozos que han producido por 500 días a una presión de fondo de 1100psia en las posiciones mostradas con cuadros negros en la Figura 5.1, se intenta ubicar tres pozos nuevos para producir por 500 días adicionales. En la Figura 5.1 se muestra con cuadros grises las ubicaciones permitidas para ubicar los nuevos pozos.

Para este caso se usaron 13 pozos de entrenamiento mostrados en la Figura 5.2 como cuadros verdes. Una vez los pozos de entrenamiento fueron seleccionados, se obtuvieron 286 combinaciones posibles de a tres pozos, estas fueron filtradas aplicando las restricciones mostradas en la sección 3.2, en la Figura 5.3 se muestran con triángulos las combinaciones de pozos permitidas.

Una vez aplicado el filtro, 90 combinaciones favorables fueron obtenidas para el entrenamiento, cada una de estas combinaciones fueron evaluadas por el simulador numérico de yacimientos para obtener la producción acumulada de cada po-

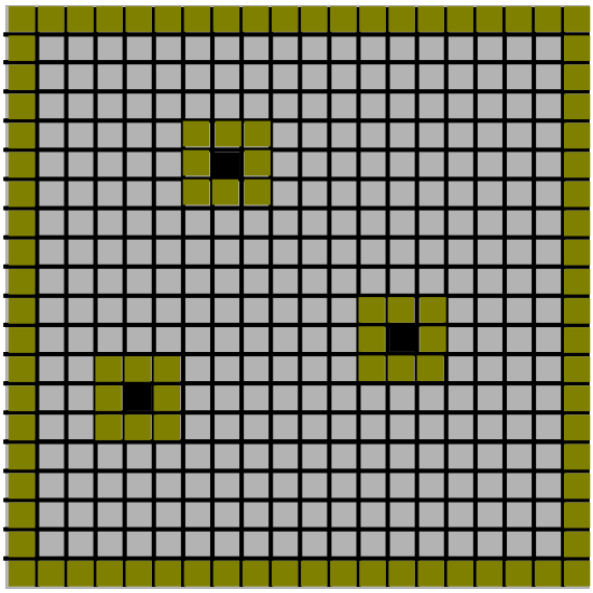


Figura 5.1: Discretización yacimiento para la prueba 5.1.

zo. Luego, los datos de entrada fueron obtenidos y normalizados. En esta prueba se evaluaron distintas redes neuronales artificiales para conocer la variación de los resultados ante diferentes números de datos de entrada, en total se evaluaron cuatro redes neuronales en las Tablas 5.3 a 5.6 se muestran el número de neuronas utilizados. Los resultados obtenidos con cada red se muestran en la Tabla 5.7 se obtuvieron buenos resultados (posiciones y flujo) con un número moderado de neuronas de entrada. La red sufre sobrecarga de información cuando el número de neuronas aumenta, lo cual es un factor a tener en cuenta en las siguientes pruebas ya que al aumentar en complejidad pueden requerir un número mayor de neuronas de entrada y por lo tanto, exhibir el mismo comportamiento.

5.2. YACIMIENTO DE GAS HETEROGÉNEO Y CON FORMA REGULAR

Este modelo cambia respecto al de la prueba 5.1 en:

- $\triangle$x = $\triangle$y = 500ft, yacimiento con volumen mayor.
- Inclusión de un mapa de permeabilidad para representar la variación de la

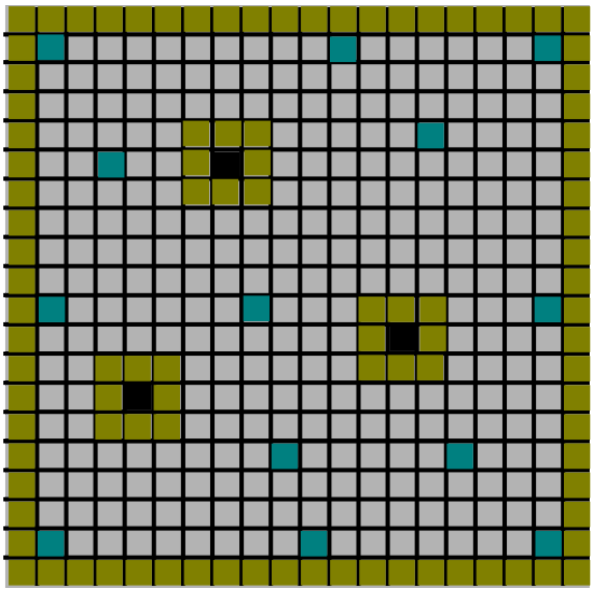


Figura 5.2: Posiciones de los pozos de entrenamiento.

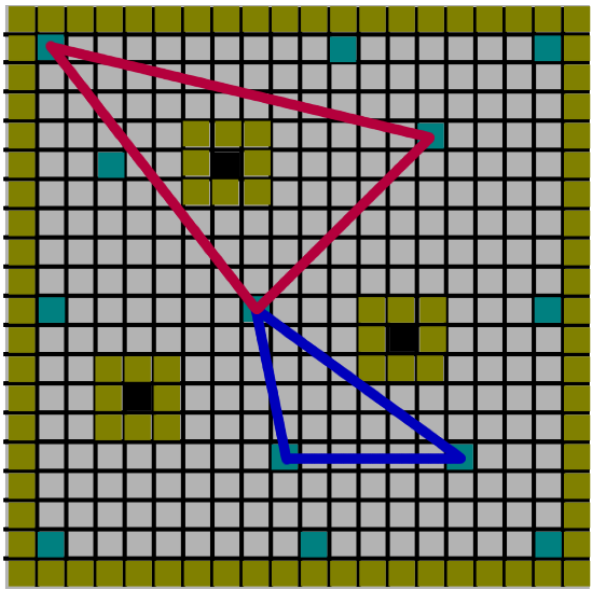


Figura 5.3: Combinaciones posibles entre pozos de entrenamiento.

Tabla 5.3: Número de neuronas usadas en la prueba 5.1.

Neuronas de entrada	
Tiempo	1 neurona
Coordenadas x y de cada pozo nuevo	6 neuronas
Perímetro del triángulo formado por los pozos nuevos	1 neurona
Cuadrado y recíproco del tiempo	2 neuronas
Cuadrado y recíproco de las coordenadas x y de cada pozo nuevo	12 neuronas
Tiempo/Cuadrado de la distancia interpozo	3 neuronas
	25 neuronas de entrada
Capa oculta	13 neuronas
Capa de salida	6 neuronas

Tabla 5.4: Número de neuronas usadas en la prueba 5.1.

Neuronas de entrada	
Tiempo	1 neurona
Coordenadas x y de cada pozo nuevo	6 neuronas
Distancia entre los 3 pozos nuevos	3 neuronas
Cuadrado y recíproco del tiempo	2 neuronas
Cuadrado y recíproco de las coordenadas x y de cada pozo nuevo	12 neuronas
Tiempo/Cuadrado de la distancia interpozo	3 neuronas
	27 neuronas de entrada
Capa oculta	13 neuronas
Capa de salida	6 neuronas

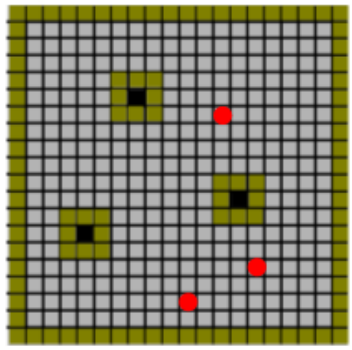
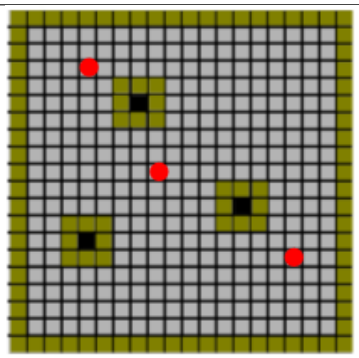
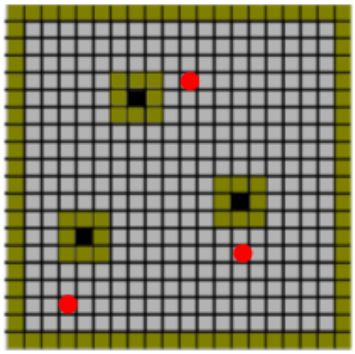
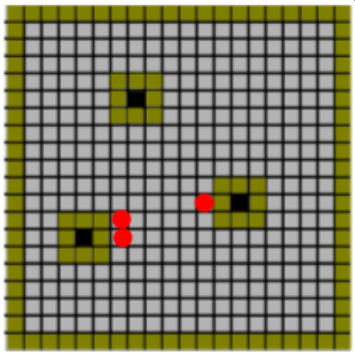
Tabla 5.5: Número de neuronas usadas en la prueba 5.1.

Neuronas de entrada	
Tiempo	1 neurona
Coordenadas x y de cada pozo nuevo	6 neuronas
Distancia entre los 3 pozos nuevos	3 neuronas
Perímetro del triángulo formado por los pozos nuevos	1 neurona
Cuadrado y recíproco del tiempo	2 neuronas
Cuadrado y recíproco de las coordenadas x y de cada pozo nuevo	12 neuronas
Tiempo/Cuadrado de la distancia inter-pozo	3 neuronas
	28 neuronas de entrada
Capa oculta	13 neuronas
Capa de salida	6 neuronas

Tabla 5.6: Número de neuronas usadas en la prueba 5.1.

Neuronas de entrada	
Tiempo	1 neurona
Coordenadas x y de cada pozo nuevo	6 neuronas
Distancia entre los 3 pozos nuevos	3 neuronas
Perímetro del triángulo formado por los pozos nuevos	1 neurona
Distancia entre los pozos nuevos y viejos	9 neuronas
Cuadrado y recíproco del tiempo	2 neuronas
Cuadrado y recíproco de las coordenadas x y de cada pozo nuevo	12 neuronas
Tiempo/Cuadrado de la distancia inter-pozo	3 neuronas
	37 neuronas de entrada
Capa oculta	13 neuronas
Capa de salida	6 neuronas

Tabla 5.7: Resultados de las configuraciones probadas para la red neuronal de la prueba 5.1.

Prueba	Posiciones	Producción (MMSCF)		% Error
		RNA	SIM	
Tabla 5.3		41358.75	43835	5.99
Tabla 5.4		43761.26	43693	0.16
Tabla 5.5		38319.82	43668	13.96
Tabla 5.6		37866.47	43386	12.72

capacidad conductiva de la roca. El mapa se puede ver en la Figura 5.4.

Con la configuración exitosa resultado de la prueba 5.1 e incluyendo en los datos de entrada la permeabilidad del bloque en la dirección X y en la dirección Y se aplicó la metodología, sin embargo, los resultados no fueron exitosos, la red neuronal no logró una buena asociación de los datos de entrada y de salida. Por lo cual que se incluyó un nuevo parámetro en los datos de entrada, un "factor de expectativa", este factor da un valor a cada bloque de la malla dependiendo de que tanto se espera ubicar un pozo en dicho bloque, a una expectativa alta se le asigna un valor alto en una escala arbitraria, el mapa de este factor es mostrado en la Figura 5.5, los valores usados para esta prueba son:

- 0.0 para ninguna esperanza de ubicar un pozo en dicha posición, usualmente usados en las ubicaciones de pozos existentes.
- 0.25 para una expectativa pobre, usualmente las ubicaciones adyacentes a los pozos existentes o en los límites del yacimiento.
- 0.5 para una expectativa media, ubicaciones adyacentes a las que tienen valores de 0.25.
- 1.0 ubicaciones de alta expectativa (bloques interiores).

Los valores que toma cada bloque del yacimiento se basa en la geometría específica del yacimiento y los pozos existentes. El factor de expectativa también puede incluir información proveniente de la heterogeneidad, distribución de presiones, distribución de saturación, y agotamiento regional, por lo tanto sus valores pueden cambiar arbitrariamente de un caso a otro.

13 pozos de entrenamiento fueron usados para esta prueba, sus ubicaciones se muestran como cuadros verdes en la Figura 5.6. Una vez la red fue entrenada (la Figura 5.7 muestra la arquitectura de la red usada), 20000 combinaciones aleatorias fueron tomadas, luego de aplicar las restricciones, el grupo se redujo a 16535 combinaciones. Sin embargo, no fue posible obtener buenos resultados, ya que los porcentajes de error entre la red neuronal artificial y el simulador numérico de

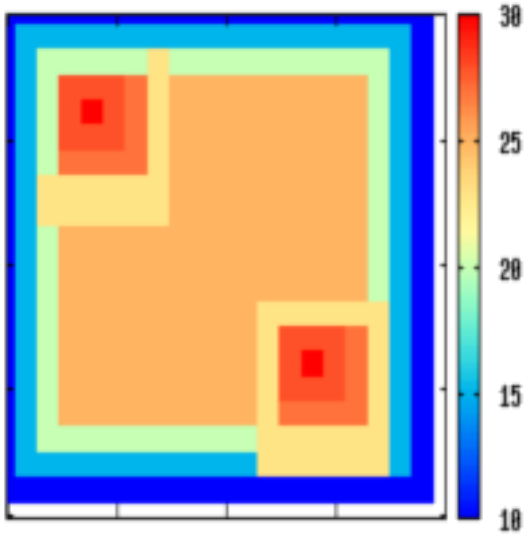


Figura 5.4: Mapa de permeabilidad asociado con la prueba 5.2.

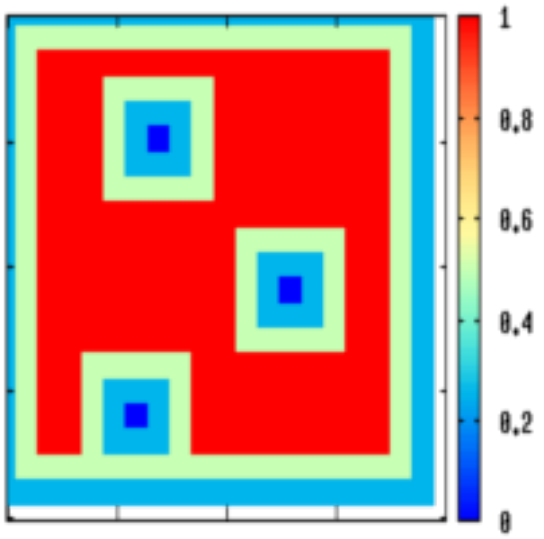


Figura 5.5: Mapa de factor de expectativa para la prueba 5.2.

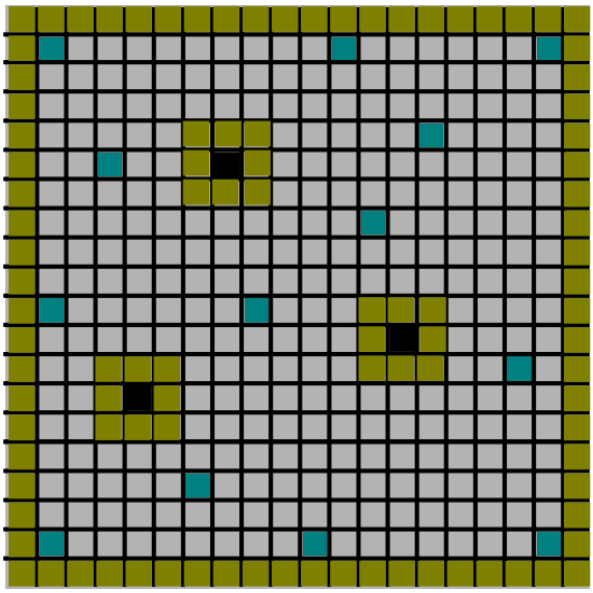


Figura 5.6: Pozos de entrenamiento prueba 5.2.

yacimientos eran mayores al $20\,\%$, mientras las curvas de entrenamiento muestran tendencias a bajos errores (Figura 5.8). De esta forma fue posible detectar que la forma de entrenar la red neuronal artificial puede afectar el ajuste de los resultados, por lo tanto, se escogió entrenar la red en pasos de 50 ciclos de iteraciones, permitiendo el ajuste de los pesos sinápticos lentamente y obtener valores de porcentaje de error de alrededor $12\,\%$ sobre el simulador de yacimientos. El número de iteraciones (épocas de entrenamiento) en el cual fue posible establecer este porcentaje de error fue 1000 como se muestra en la Figura 5.9. Los resultados para esta prueba se muestran en la Figura 5.10 y en la Figura 5.11 se muestra el aumento en el recobro con la perforación de los pozos nuevos.

5.3. YACIMIENTO DE GAS HETEROGÉNEO Y CON FORMA IRREGULAR

En este modelo se adiciona una geometría irregular, puede ser observada en la Figura 5.12, los valores de permeabilidad varían entre 30md y 0md, el último valor ha sido usado para representar una barrera de no flujo en los límites irregulares del yacimiento. El mapa de factor de expectativa es ajustado para que pueda aislar la ubicación de los nuevos pozos de los límites del yacimiento; para este caso, los val-

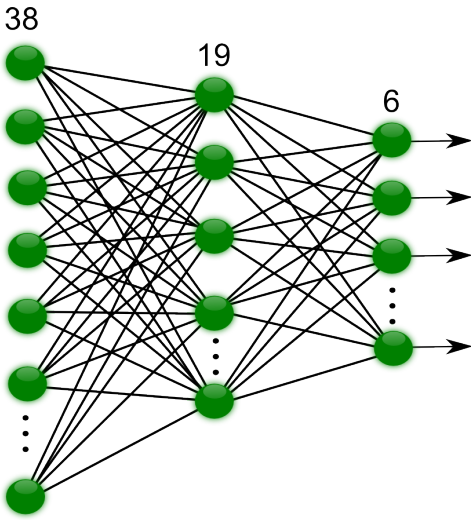


Figura 5.7: Arquitectura de la red neuronal usada en la prueba 5.2 y 5.3.

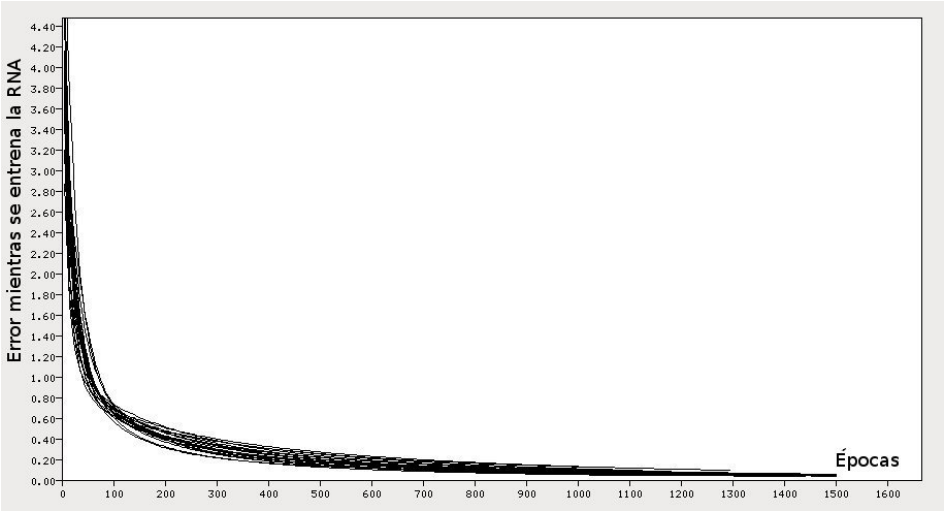


Figura 5.8: Múltiples pruebas no exitosas en la prueba 5.2.

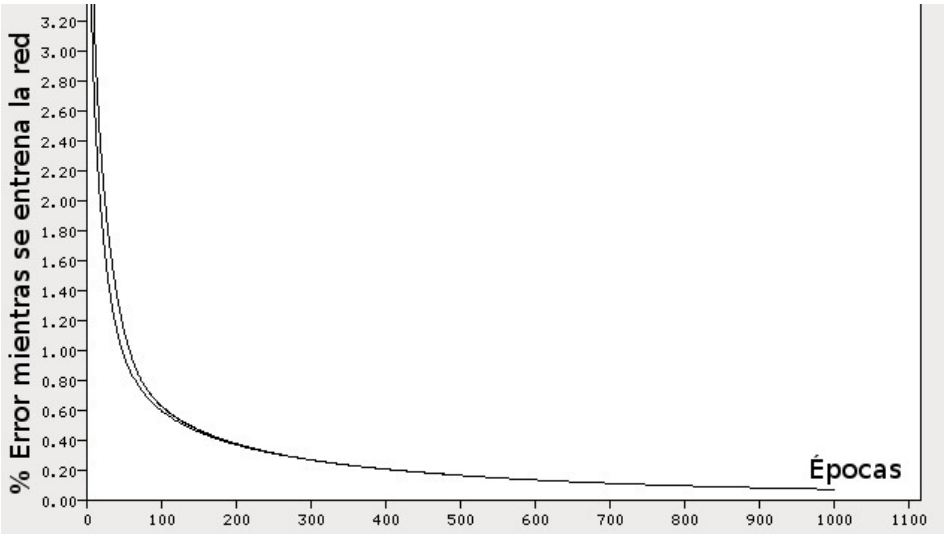


Figura 5.9: Curva de error asociada con un entrenamiento exitoso para la prueba 5.2.

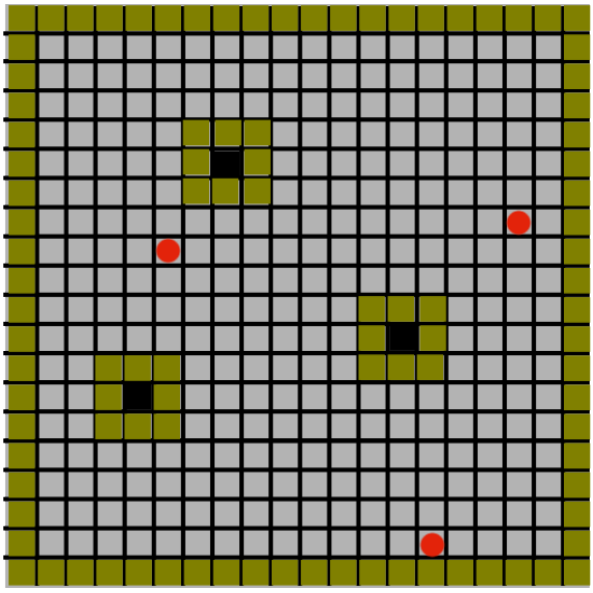


Figura 5.10: Resultados para la prueba 5.2.

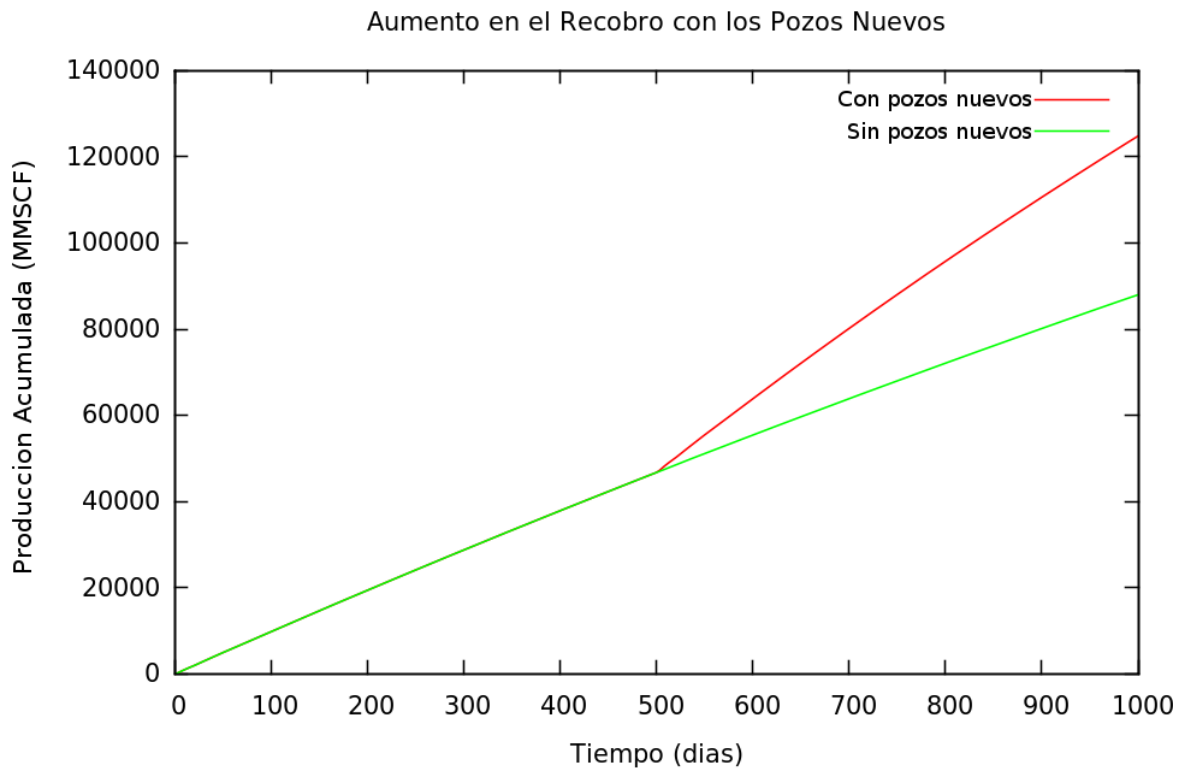


Figura 5.11: Aumento en el recobro con la perforación de los pozos nuevos, prueba 5.2.

ores del factor de expectativa fueron modificados con respecto a la prueba anterior para poder representar los límites irregulares (no cuadrados), los valores utilizados son:

- 0.0 para ninguna esperanza de ubicar un pozo en dicha posición, usualmente usados en las ubicaciones de pozos existentes.
- 0.1 para bloques en los límites del yacimiento.
- 0.25 para una expectativa pobre, usualmente las ubicaciones adyacentes a los pozos existentes o en los límites del yacimiento.
- 0.5 para una expectativa media, ubicaciones adyacentes a las que tienen valores de 0.25.
- 1.0 ubicaciones de alta expectativa (bloques interiores).

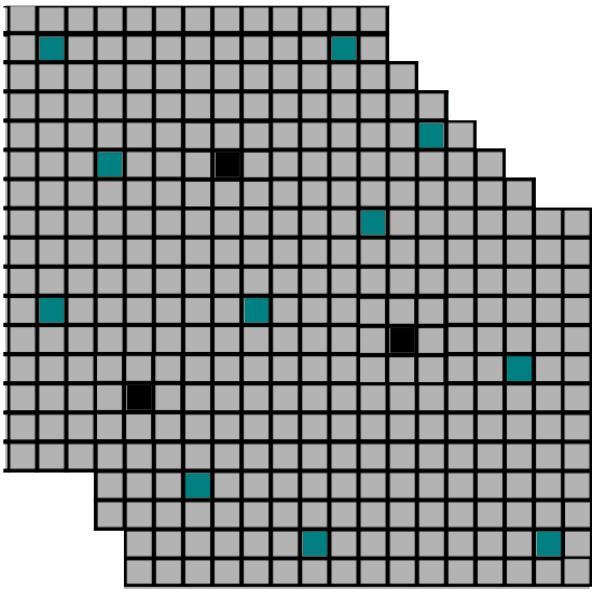


Figura 5.12: Yacimiento usado en la prueba 5.3, pozos existentes (cuadros negros), pozos de entrenamiento (cuadros verdes).

El mapa del factor de expectativa se muestra en la Figura 5.13. El número de pozos de entrenamiento fue 11, su distribución se muestra en la Figura 5.12. Una vez se entrenó la red neuronal artificial (arquitectura mostrada en la Figura 5.7, se tomaron 20000 combinaciones aleatorias y luego de aplicar los filtros se obtuvieron 15938 combinaciones. El patrón de comportamiento con respecto a las tasas de error obtenidas fue el mismo que para la prueba 5.2, por lo tanto se realizaron ciclos cortos de entrenamiento, para esta prueba tomó 1050 iteraciones para ajustar debidamente los pesos sinápticos (Figura 5.14). Los resultados tuvieron un porcentaje de error de $10{,}6\,\%$ en la mejor posición establecida por la red; en la Figura 5.15 se muestran los resultados para esta prueba y en la Figura 5.16 se muestra el aumento en el recobro con la perforación de los pozos nuevos.

5.4. YACIMIENTO DE GAS HETEROGÉNEO, CON FORMA IRREGULAR Y DOS ETAPAS DE PERFORACIÓN

Las propiedades del yacimiento se presentan en la Tabla 5.8, la forma del yacimiento se observa en la Figura 5.17, con cuatro pozos existentes (cajas amarillas) pro-

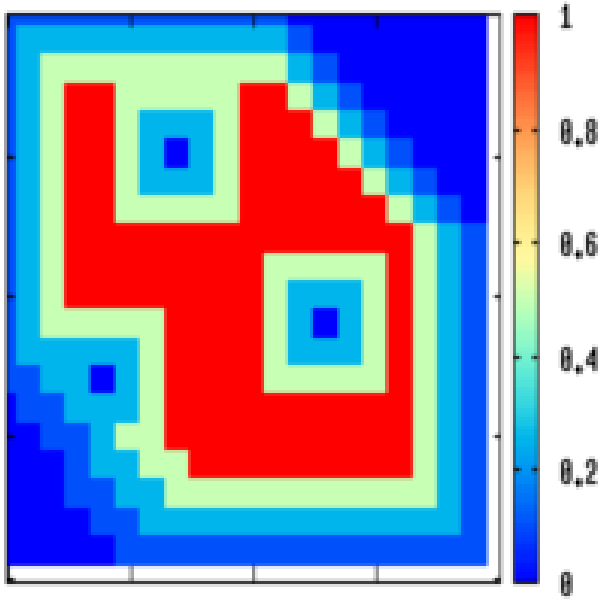


Figura 5.13: Factor de expectativa implementado en la prueba 5.3.

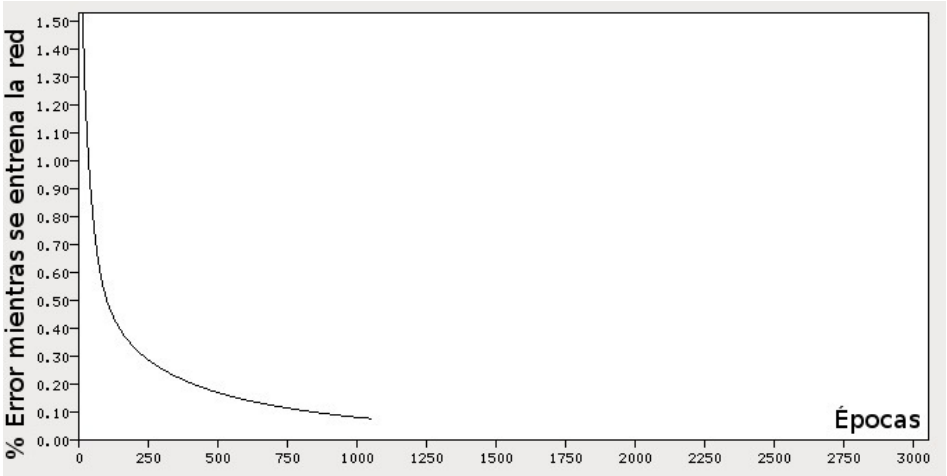


Figura 5.14: Curva de error asociada con entrenamiento exitoso para la prueba 5.3.

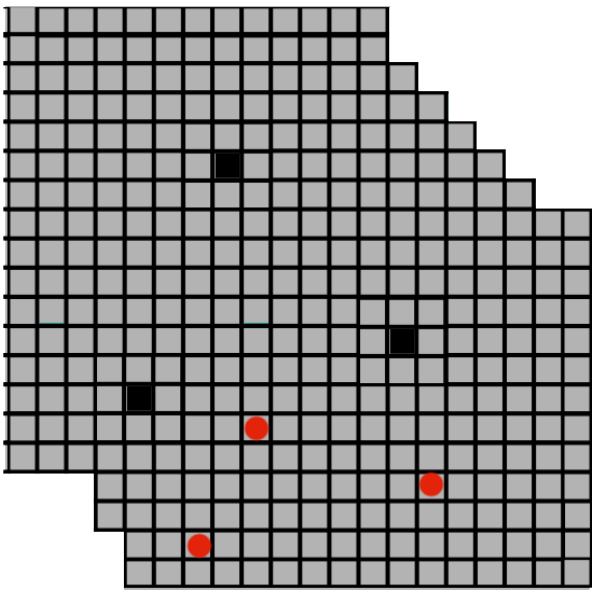


Figura 5.15: Resultados para la prueba 5.3.

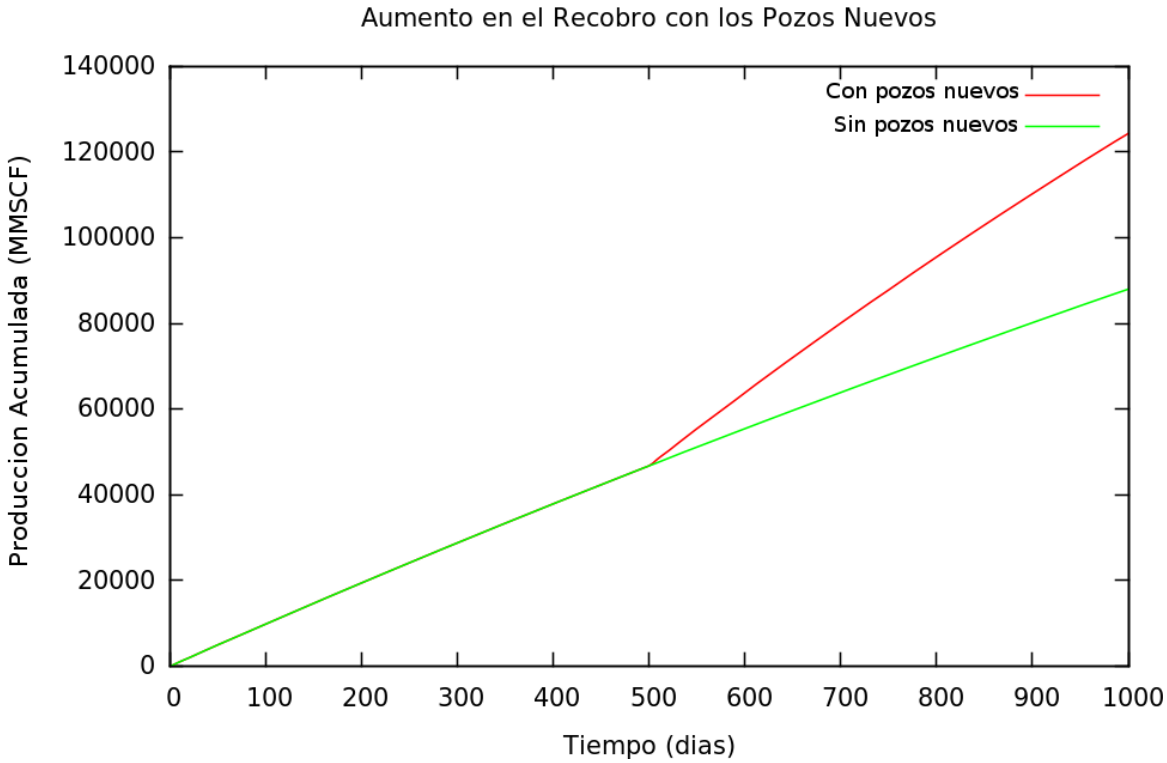


Figura 5.16: Aumento en el recobro con la perforación de los pozos nuevos, prueba 5.3.

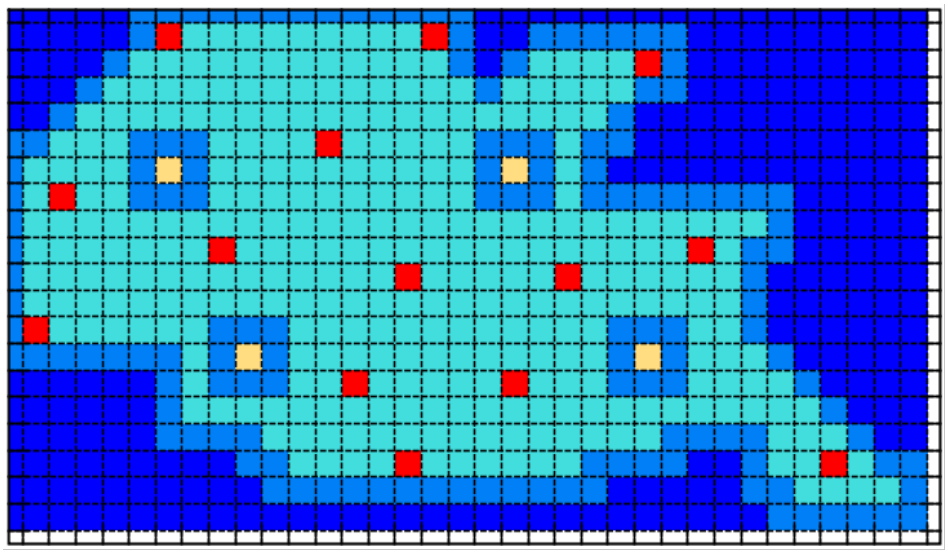


Figura 5.17: Modelo del yacimiento (35x20) para la prueba 5.4: Cajas amarillas, pozos existentes. Cajas rojas, pozos de entrenamiento.

duciendo por 500 días, es necesario ubicar un total de seis pozos nuevos en el siguiente orden:

- Una etapa en la cual se perforarán 3 pozos que producirán por 350 días.
- Una segunda etapa donde se perforarán otros 3 pozos que producirán por 350 días más.

Para la primera etapa de perforación se generó 14 pozos de entrenamiento (Figura 5.17). El mapa de permeabilidad se muestra en la Figura 5.18 y en la Figura 5.19 se muestra el mapa de factor de expectativa, los valores usados son los mismos que en la prueba 5.3. El entrenamiento de la red fue hecho en ciclos cortos, la arquitectura de la red neuronal artificial usada y la curva de error obtenida mientras la red neuronal fue entrenada son mostradas en las Figuras 5.20 y 5.21 respectivamente; un grupo de 50000 combinaciones aleatorias fue generado, después de aplicar los filtros el número de combinaciones fue reducido a 39278, los resultados para esta etapa son mostrados en la Figura 5.22 con círculos negros con un porcentaje de error de $10\,\%$.

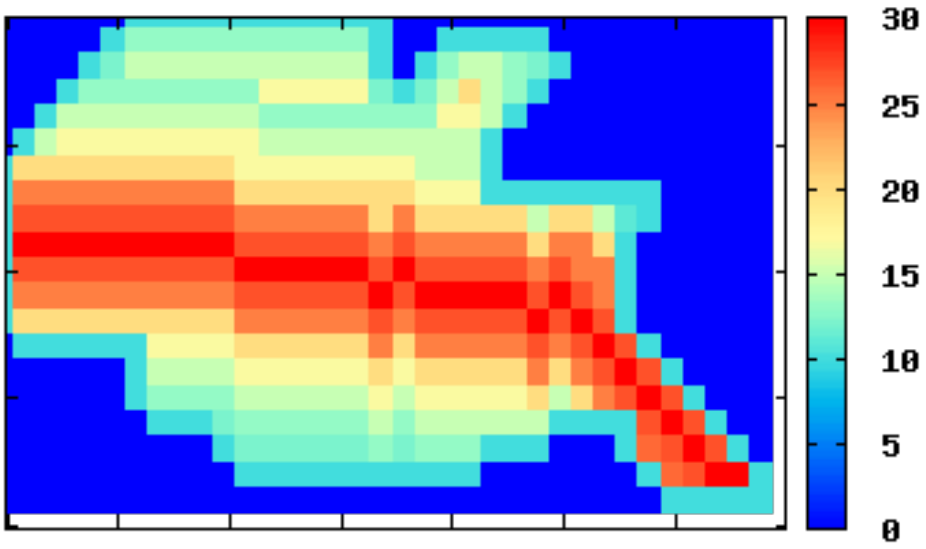


Figura 5.18: Mapa de permeabilidad para la prueba 5.4.

Tabla 5.8: Propiedades del yacimiento para la prueba 5.4.

Presión inicial del yacimiento	6200 psia
Temperatura del yacimiento	144 °F
Permeabilidad del yacimiento	Figura 5.18
Porosidad	25 %
$\Delta x = \Delta y$	500 ft
Espesor del yacimiento	75 ft
Radio del wellbore	0.25 ft

Para la segunda etapa, se tomaron los pozos nuevos ubicados en la primera etapa como pozos viejos, se mantuvo el mismo número y posiciones de los pozos de entrenamiento usados en la primera etapa, el mapa de factor de expectativa fue cambiado debido a la inclusión de los tres pozos nuevos de la primera etapa (Figura 5.23), la arquitectura de la red neuronal es mostrada en la Figura 5.24 y la curva de error mientras se entrenó la red se muestra en la Figura 5.25. Luego de que la red neuronal fue entrenada, se generó un grupo de 50000 combinaciones aleatorias y luego de los filtros el grupo fue de 38381 combinaciones, los resultados para esta etapa se presentan en la Figura 5.26. El aumento en el recobro por la perforación de los pozos nuevos se muestra en la Figura 5.27.

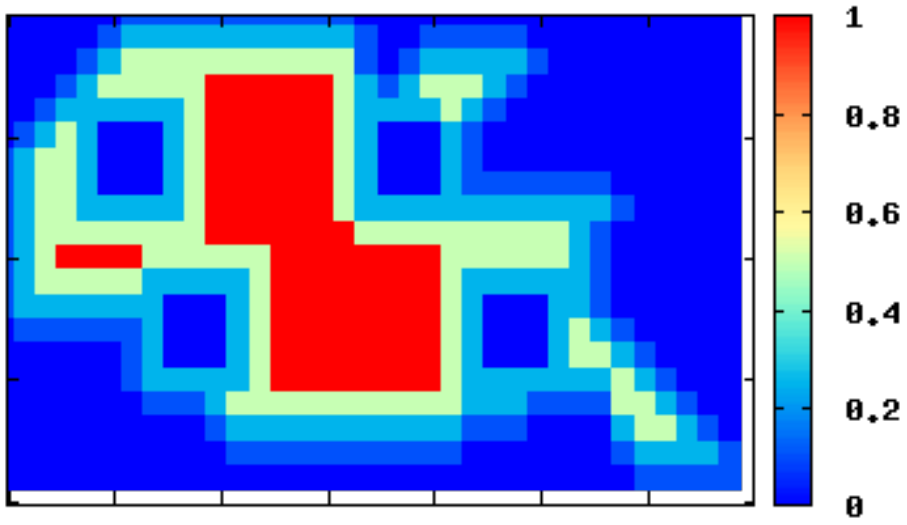


Figura 5.19: Mapa de factor de expectativa para la prueba 5.4.

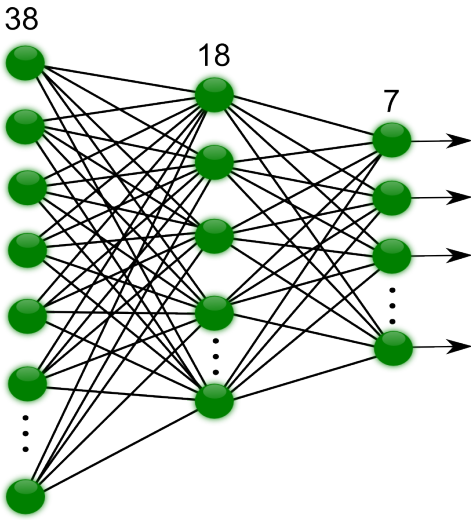


Figura 5.20: Arquitectura de la red neuronal usada para la primera etapa de perforación, prueba 5.4.

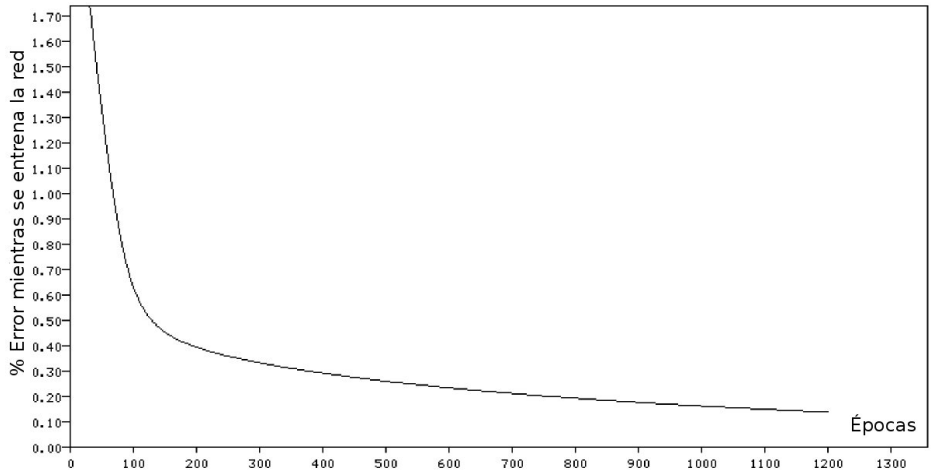


Figura 5.21: Curva de error para la primera etapa, prueba 5.4.

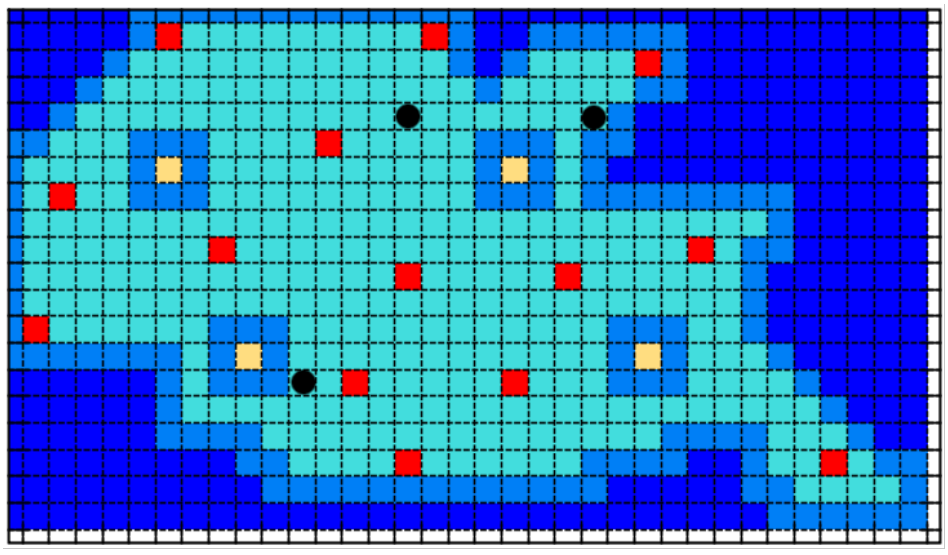


Figura 5.22: Resultados para la primera etapa, prueba 5.4.

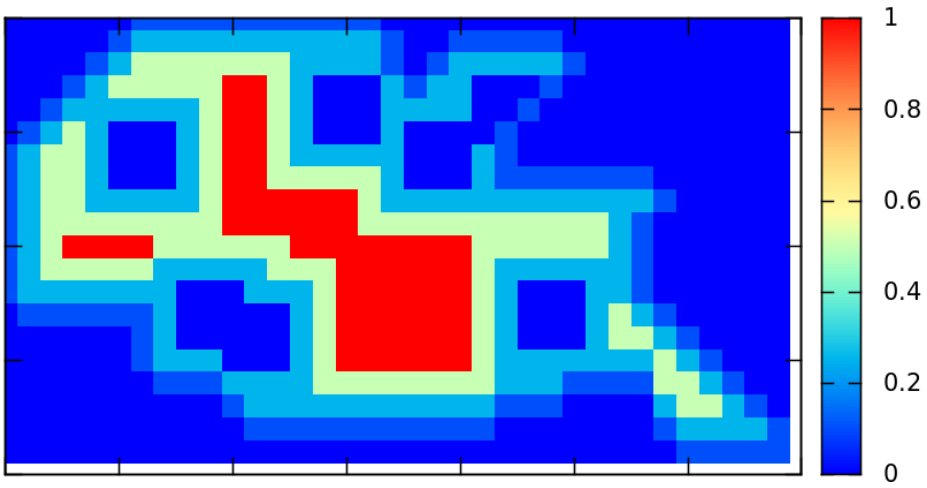


Figura 5.23: Mapa de factor de expectativa para la segunda etapa de la prueba 5.4.

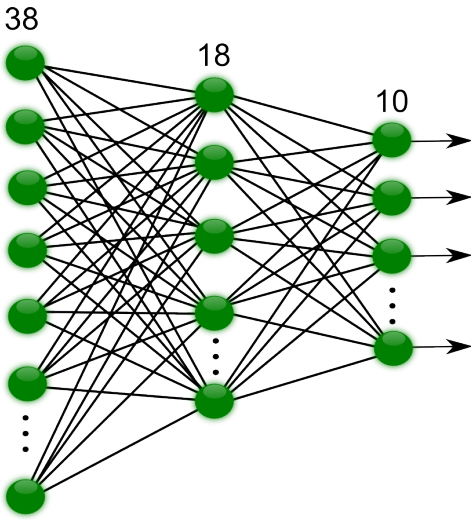


Figura 5.24: Arquitectura de la red neuronal usada para la segunda etapa de perforación, prueba 5.4.

5.5. INCLUSIÓN DE LOS POZOS DE VALIDACIÓN Y CAMBIO EN LA NORMALIZACIÓN DE LA PRODUCCIÓN ACUMULADA

Para esta prueba se toma el modelo usado en la prueba 5.4 usando solo una etapa de perforación, se incluye un grupo de 7 pozos de validación para controlar el sobre-

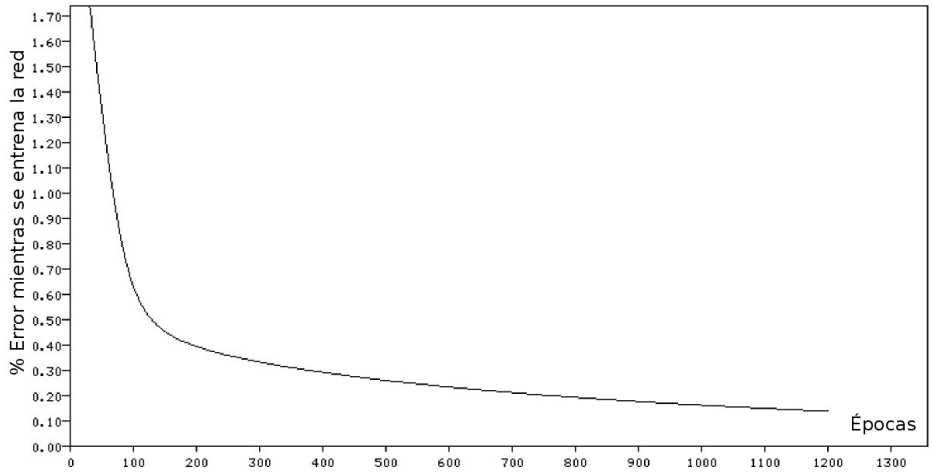


Figura 5.25: Curva de error para la segunda etapa, prueba 5.4.

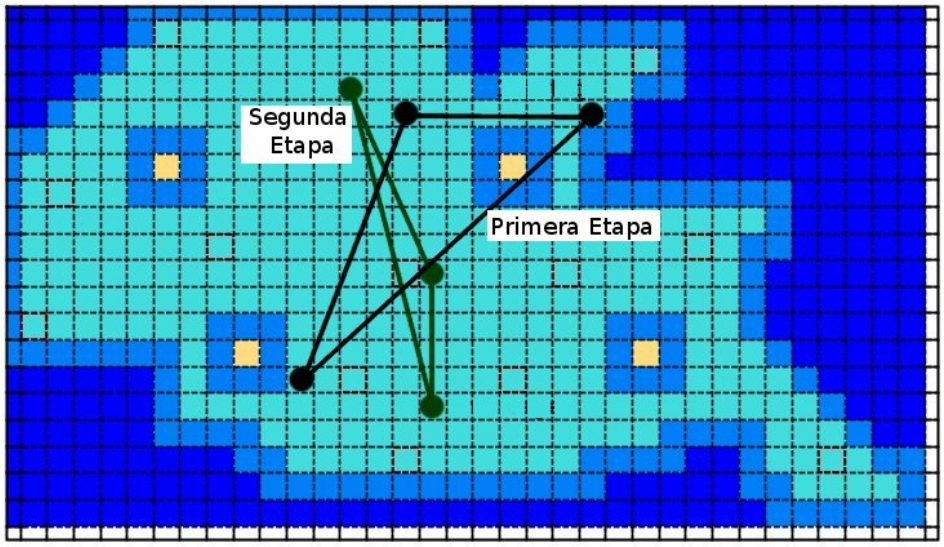


Figura 5.26: Resultados para la prueba 5.4: Negro, primera etapa. Verde, segunda etapa.

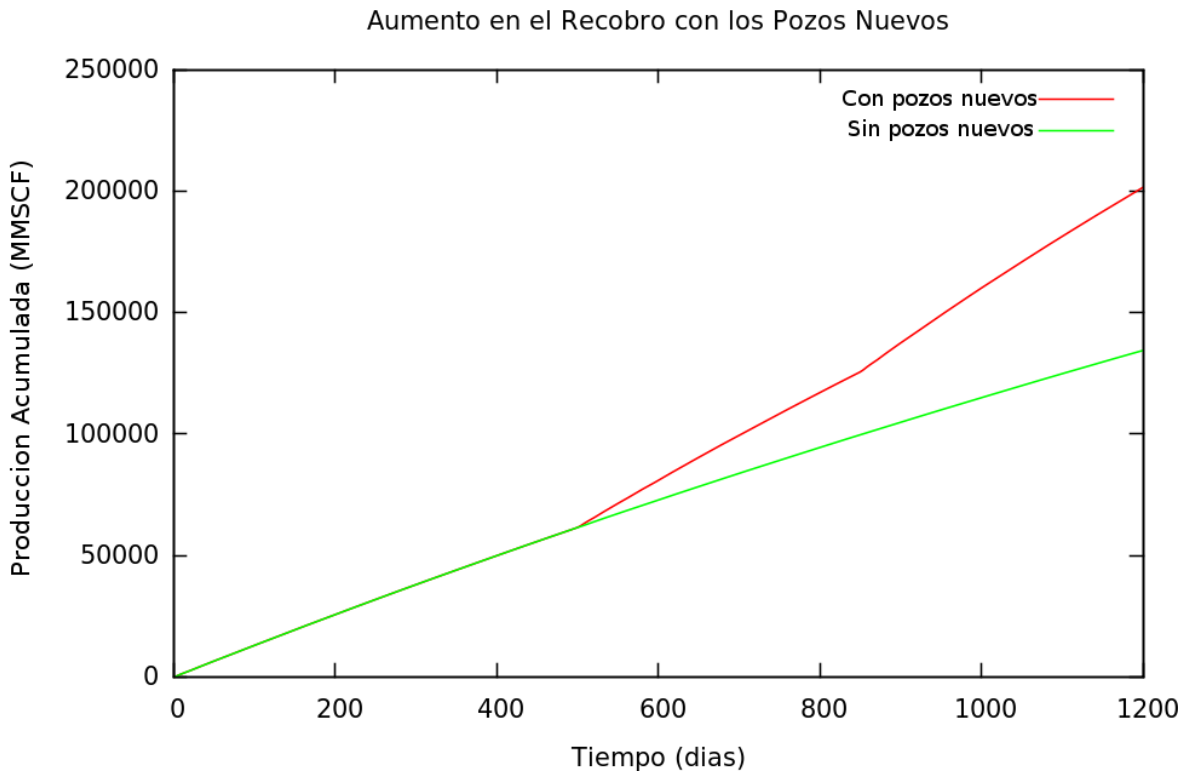


Figura 5.27: Aumento en el recobro con la perforación de los pozos nuevos, prueba 5.4.

entrenamiento, la distribución de pozos se muestra en la Figura 5.28, además se cambió la forma de normalizar la producción acumulada de cada pozo como se explica en la subsección 3.2.2 usando la Ecuación 3.6. Todo lo demás permaneció igual a la prueba anterior. En la Figura 5.29 se muestra la curva de error obtenida mientras se entrenó la red neuronal artificial, la curva negra es el error obtenido por los datos de entrenamiento, la curva roja es el error obtenido por los datos de validación, siempre que ambas curvas sean decrecientes no se presenta sobre-entrenamiento, si se continúa entrenando la red neuronal la curva de error de validación se cambiará de pendiente y empezará a tener un comportamiento creciente, cuando esto se presenta se dice que la red esta sobre-entrenada, que “memorizó en lugar de generalizar”, por lo tanto obtendrá buenos resultados para el conjunto de datos de entrenamiento pero para datos diferentes a estos los resultados serán erróneos, un ejemplo de una red neuronal artificial sobre-entrenada se muestra en la Figura 5.30.

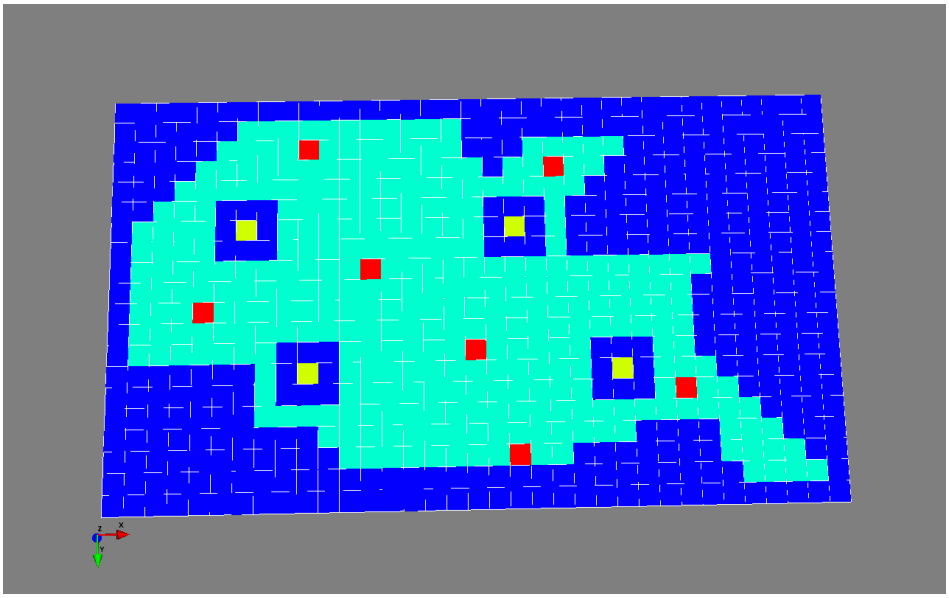


Figura 5.28: Pozos de validación para la prueba 5.5: Cuadros rojos.

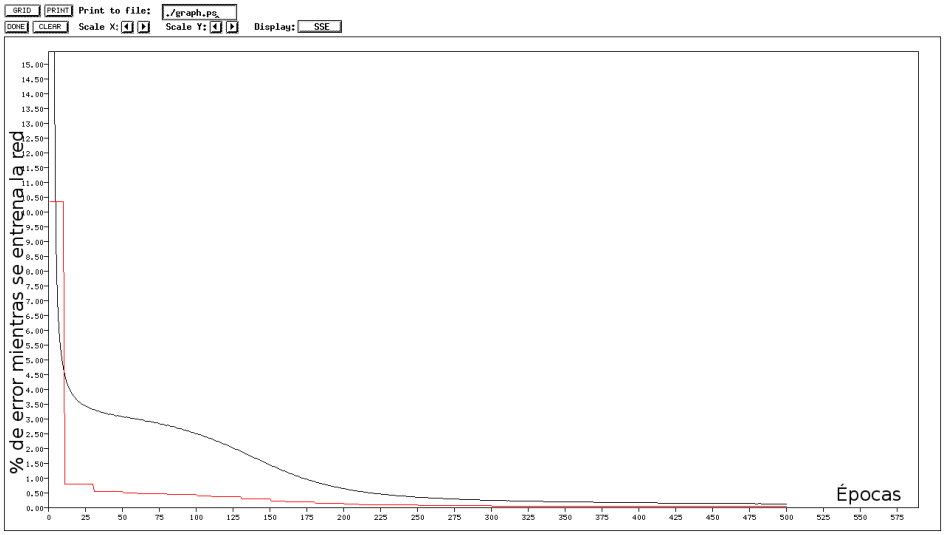


Figura 5.29: Curva de error para la prueba 5.4.

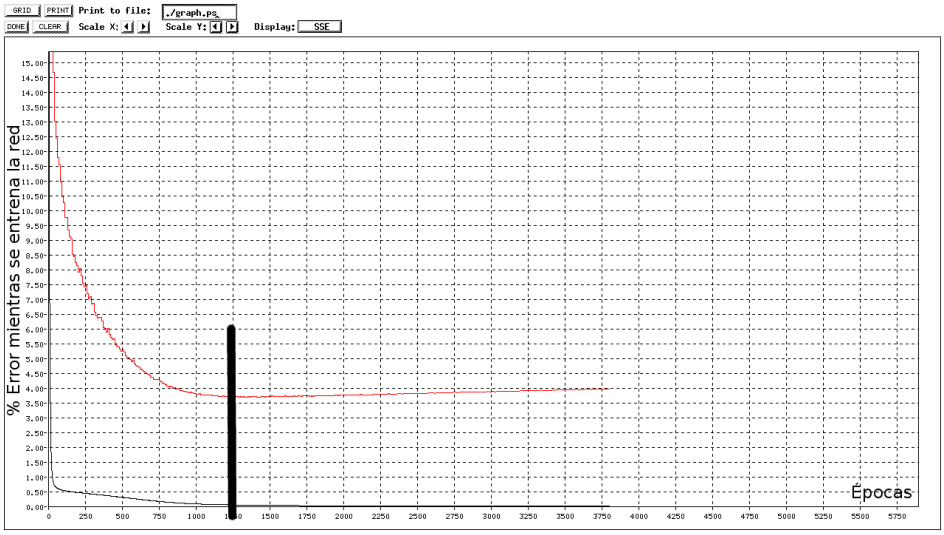


Figura 5.30: Curva de error para donde se aprecia el sobre-entrenamiento.

Los diez mejores resultados sugeridos por la red neuronal artificial se muestran en la Figura 5.31 con un porcentaje de error de $2{,}05\,\%$ mucho menor que para la prueba anterior, además se puede observar dos zonas del yacimiento muy propicias para la ubicación de pozos nuevos estas zonas se resaltan en la Figura 5.34. Cabe resaltar que al momento de realizar esta prueba ya se había programado la salida de resultados en archivos vtk y su visualización se hizo en Mayavi2 en lugar de GNUplot programa para la realización de gráficos usados en las pruebas anteriores.

En la Figura 5.33 se muestra el aumento en el recobro por la perforación de los pozos nuevos con la mejor combinación de ubicaciones.

En la Tabla 5.9 se muestra en resumen todas las pruebas realizadas con sus respectivos resultados y porcentaje de error respecto al simulador numérico de yacimientos, se puede observar que a medida que aumenta la complejidad en los modelos el error respecto al simulador se mantiene o aumenta hasta llegar a un valor de $24\,\%$ en la prueba 5.4 segunda etapa, pero cuando se aplican todos los cambios a la metodología (prueba 5.5) que es la misma que la 5.4 primera etapa, se observa una disminución del error de $10.03\,\%$ a $2.05\,\%$ lo que hace mucho más aceptables los resultados.

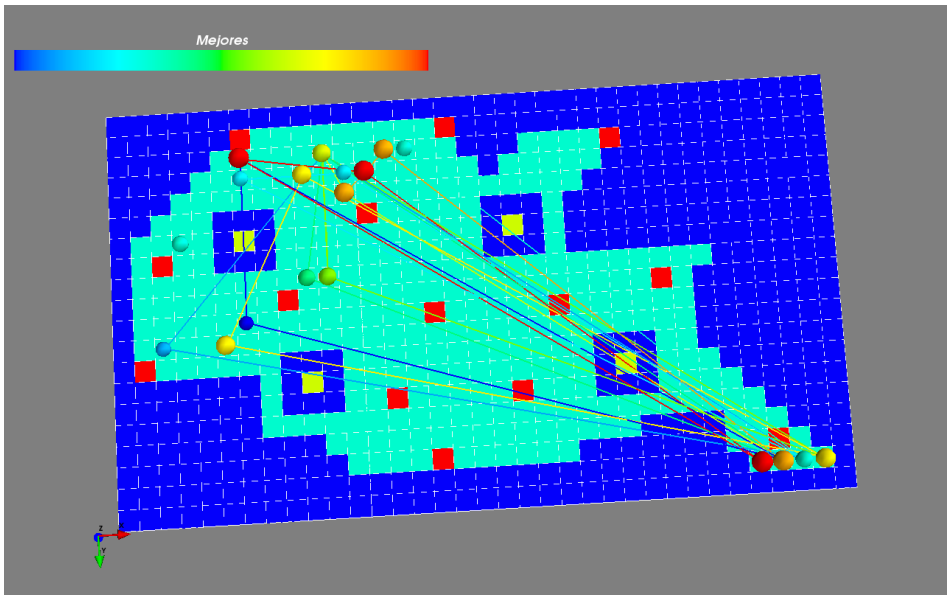


Figura 5.31: Diez mejores resultados para la prueba 5.5, mostrados con círculos.

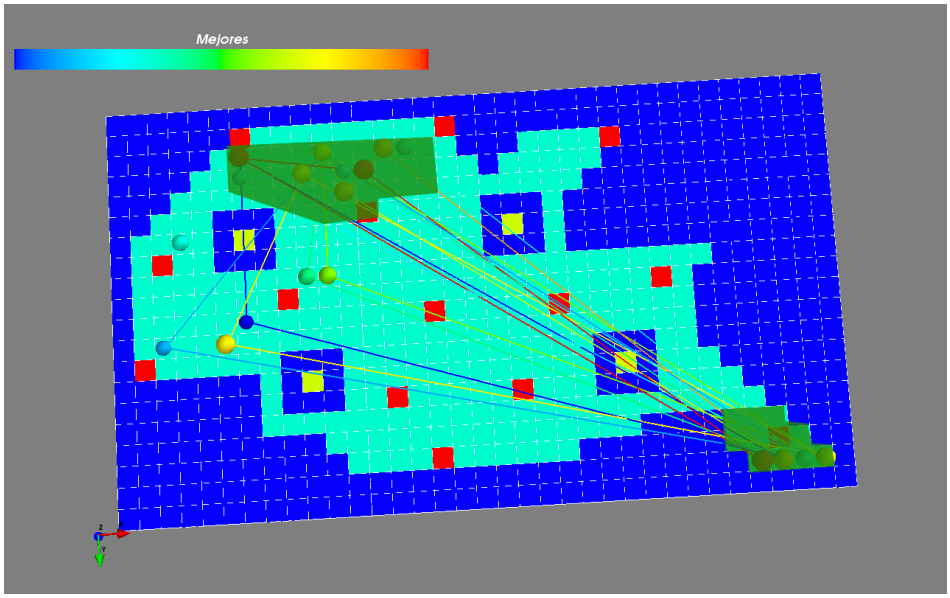
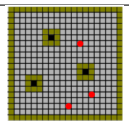
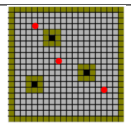
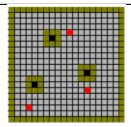
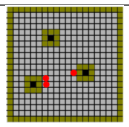
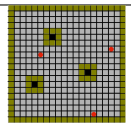
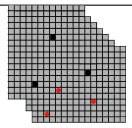
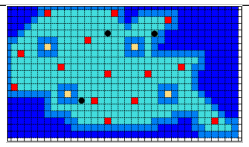
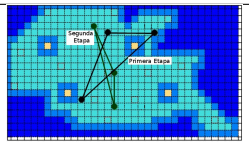
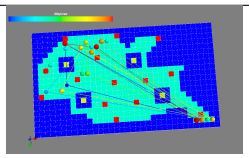


Figura 5.32: Zonas propicias para la ubicación de nuevos pozos marcadas con polígonos verdes.

Tabla 5.9: Resultados de todas las pruebas

Prueba	Yacimiento	Arch	Producción (MMSCF)		% Error	N.Épocas
		RNA	SIM			
5.1 a		25:13:6	41358.75	43835	5.99	100
5.1 b		27:13:6	43761.26	43693	0.16	200
5.1 c		28:13:6	38319.82	43668	13.96	200
5.1 d		37:13:6	37866.47	43386	12.72	300
5.2		32:19:6	140048.40	124827	12.19	1000
5.3		32:19:6	137548.19	124310	10.6	1050
5.4 a		38:18:6	138266.41	125654	10.03	1200
5.4 b		38:18:9	250199.88	201030	24.45	1200
5.5		38:18:7	128413.16	125825	2.05	1200

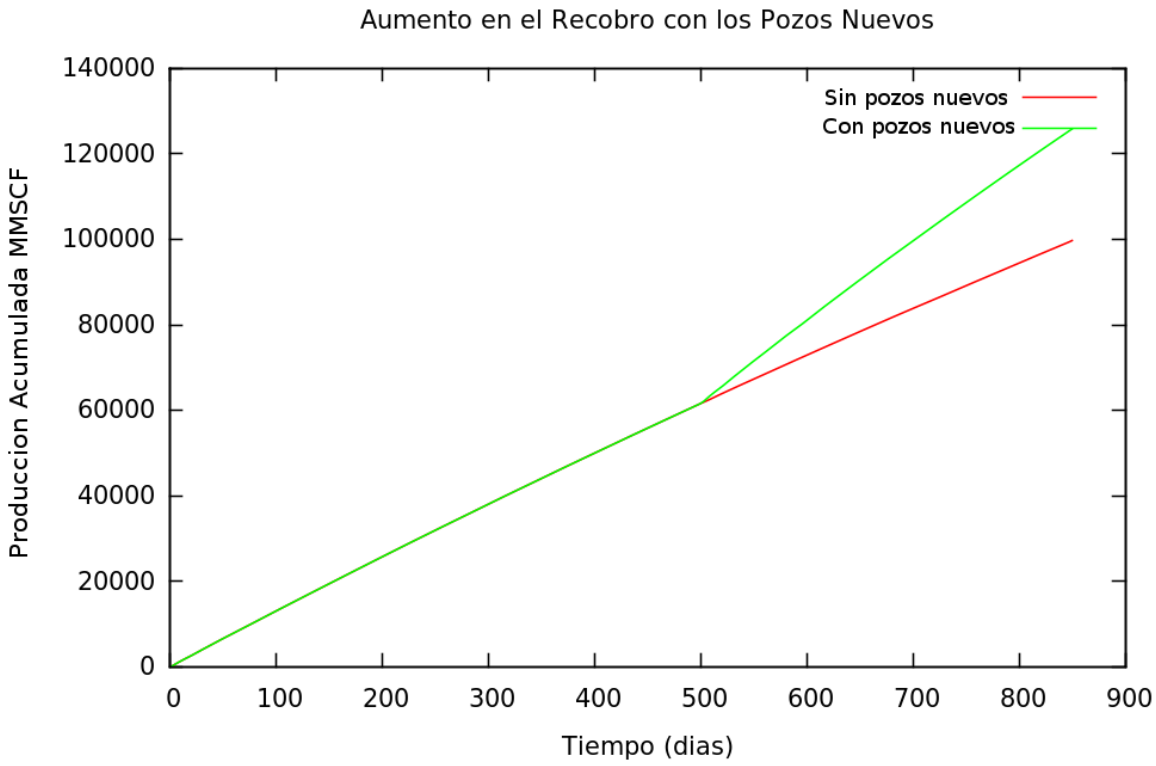


Figura 5.33: Aumento en el recobro con la perforación de los pozos nuevos, prueba 5.5.

5.6. USO DE REDES NEURONALES RECURRENTES

Esta prueba difiere con la anterior en que se cambió el tipo de red neuronal por una red neuronal recurrente, con algoritmo de entrenamiento Backpropagation Through Time, la arquitectura de la red usada se muestra en la Figura 2.6, basada en [7], en la cual el número de neuronas recurrentes en la capa de entrada equivale a el número de neuronas no recurrentes en la capa oculta; de la misma forma, el número de neuronas recurrentes en la capa oculta equivale a el número de neuronas en la capa de salida. Como se puede ver en la Figura 5.35 el error de la red neuronal mientras se entrenaba no disminuyó, lo que indica que para estos casos este tipo de red neuronal no es la adecuada.

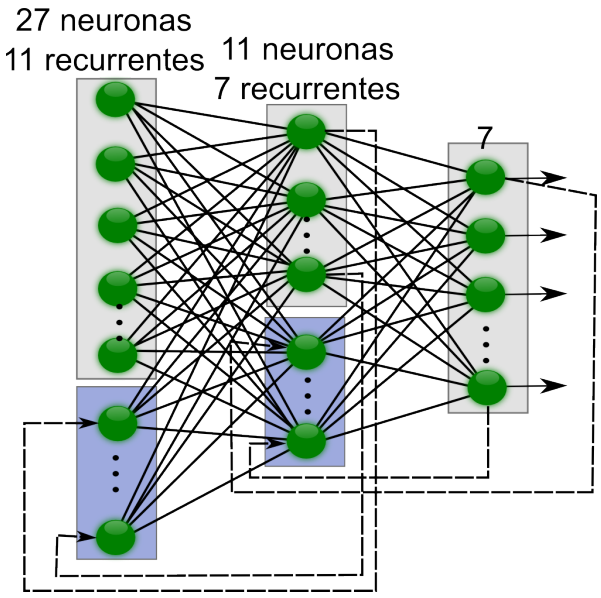


Figura 5.34: Arquitectura de la red neuronal artificial usada las neuronas dentro de los cuadrados azules representan las neuronas recurrentes.

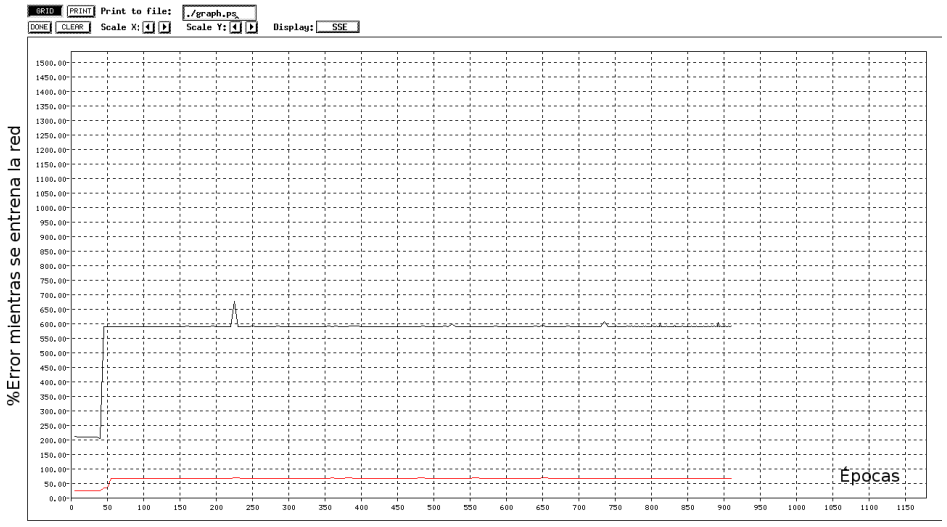


Figura 5.35: Curva de error asociado a la prueba 5.5.

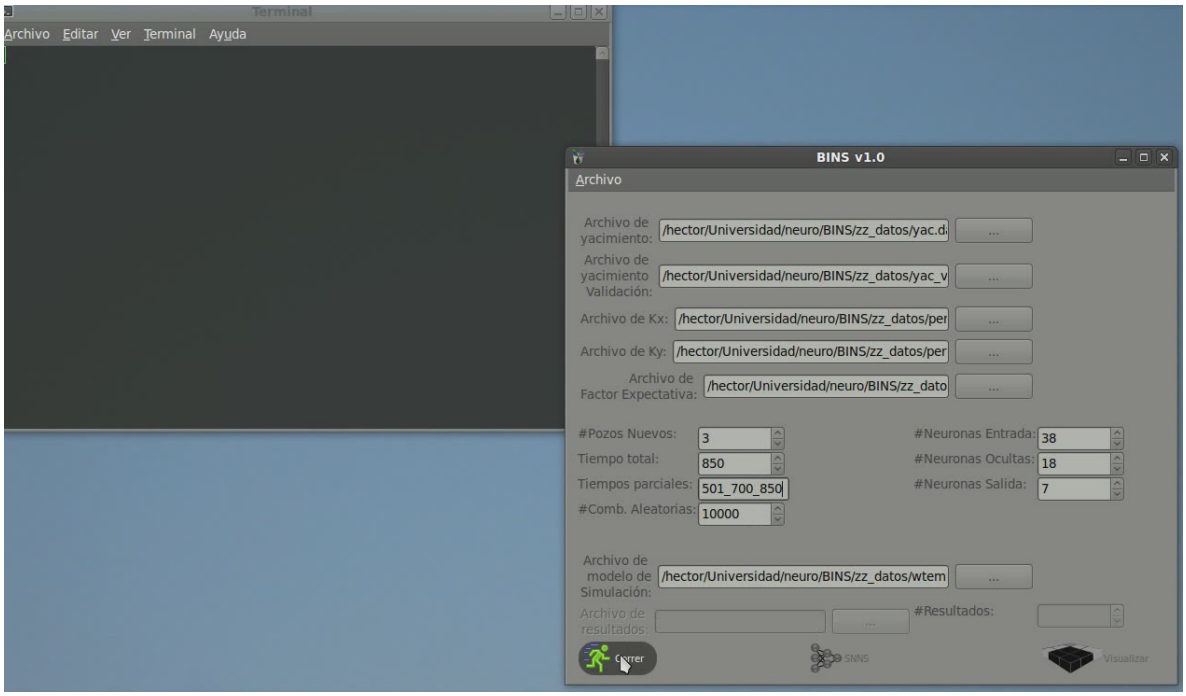


Figura 5.36: Ventana principal con el ingreso de los datos iniciales.

5.7. HACIENDO USO DE LA HERRAMIENTA SOFTWARE

En esta prueba lo que se busca es mostrar visualmente la herramienta, por lo tanto las Figuras 5.36 a 5.43 muestran la ejecución de la herramienta, el caso ejecutado es la prueba 5.5 con un menor número pozos de entrenamiento, por lo tanto los resultados no son los mejores, pero para este caso lo que se busca es mostrar la herramienta.

En la Figura 5.44 se muestra un video de la ejecución de la herramienta, para poder visualizarlo es necesario abrir el archivo en el sistema operativo Windows con el lector de archivos PDF adove reader y tener instalado el reproductor de medios windows media player junto con los codecs para reproducción de Xvid.

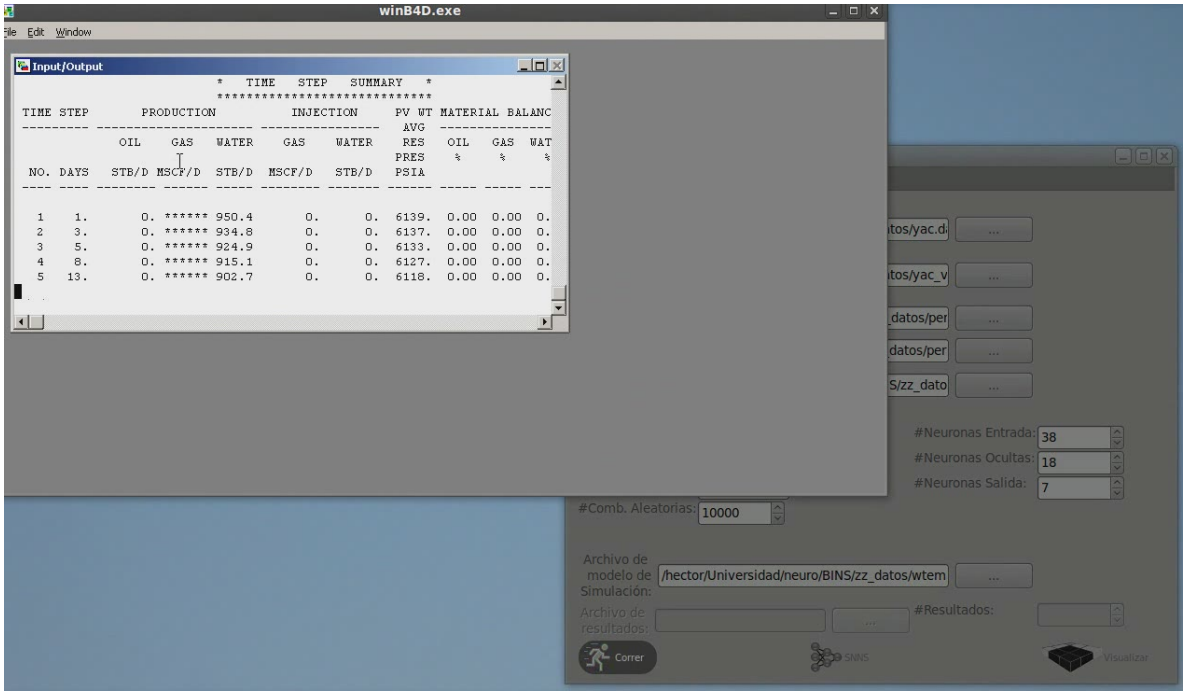


Figura 5.37: Automatización de la corrida de simulación.

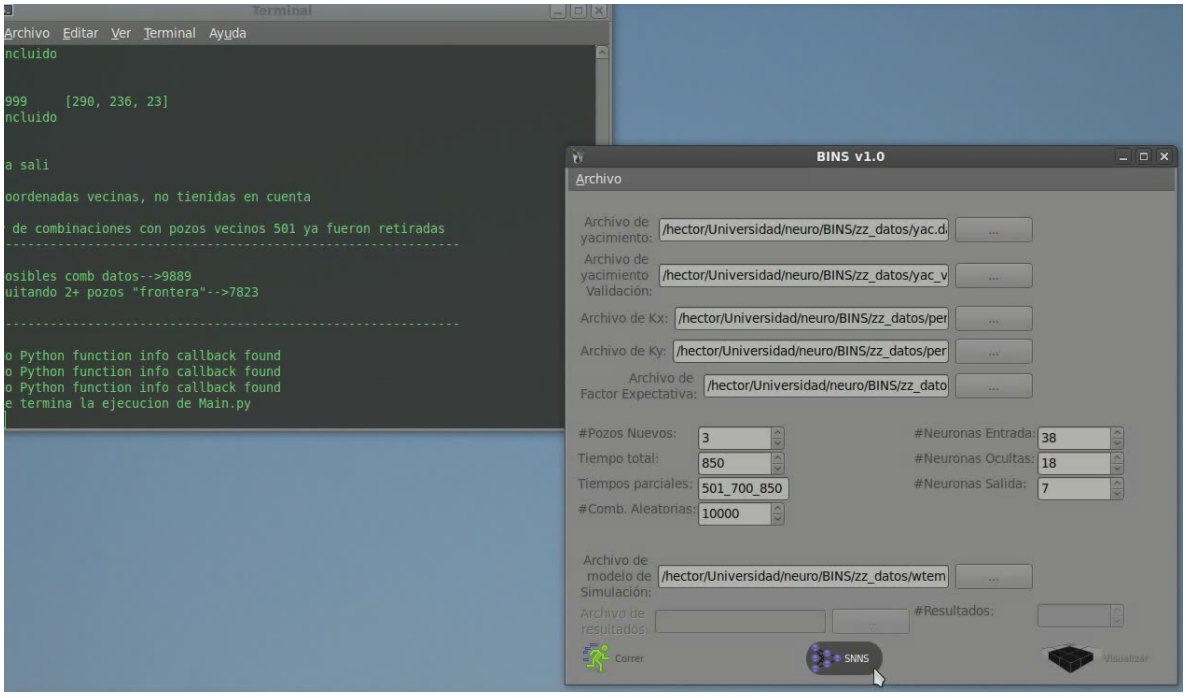


Figura 5.38: Ventana principal antes de abrir el simulador de redes neuronales.

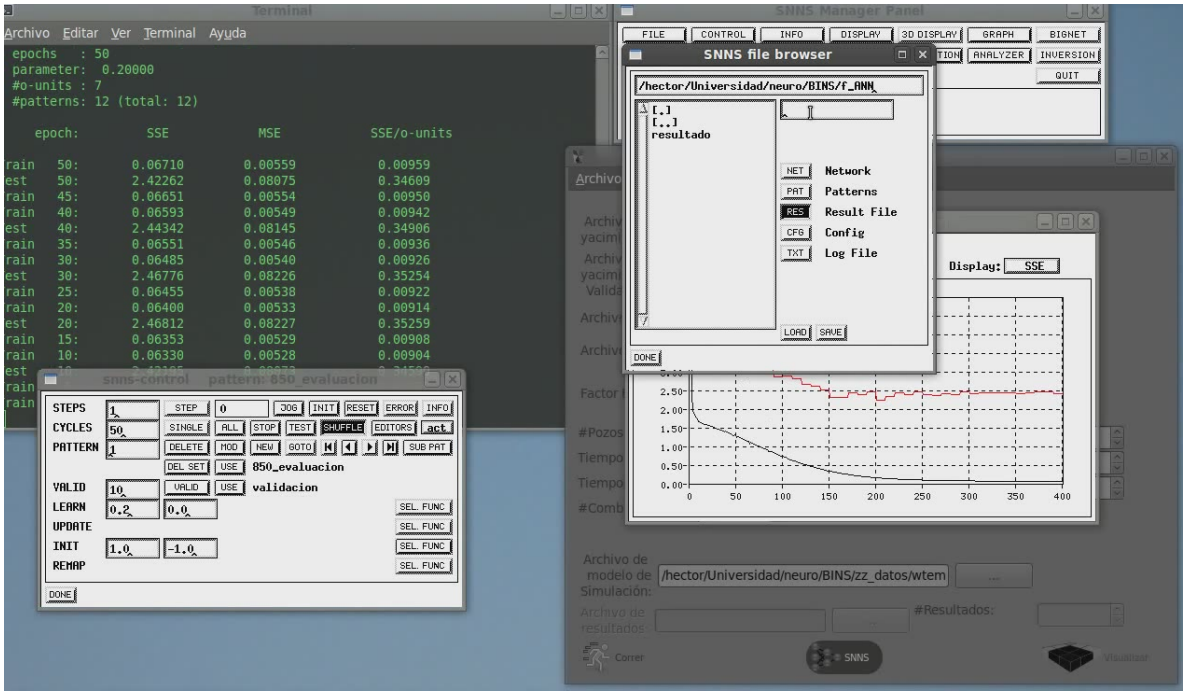


Figura 5.39: Entrenando la red neuronal artificial.

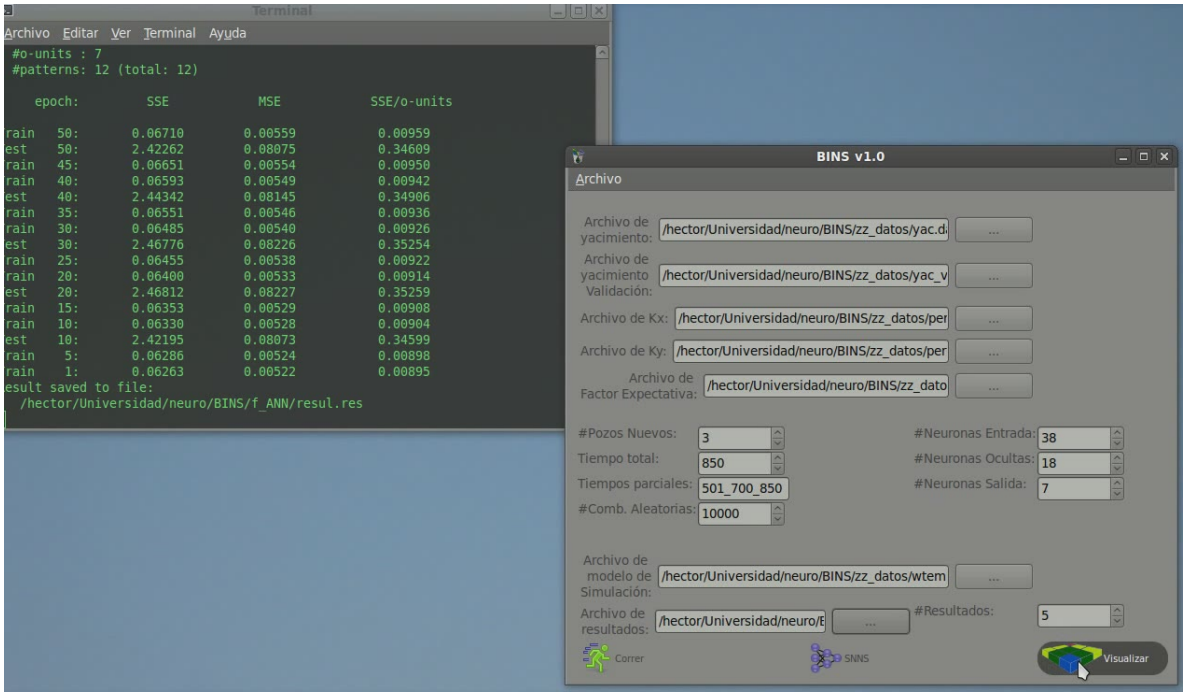


Figura 5.40: Ventana principal antes de visualizar resultados.

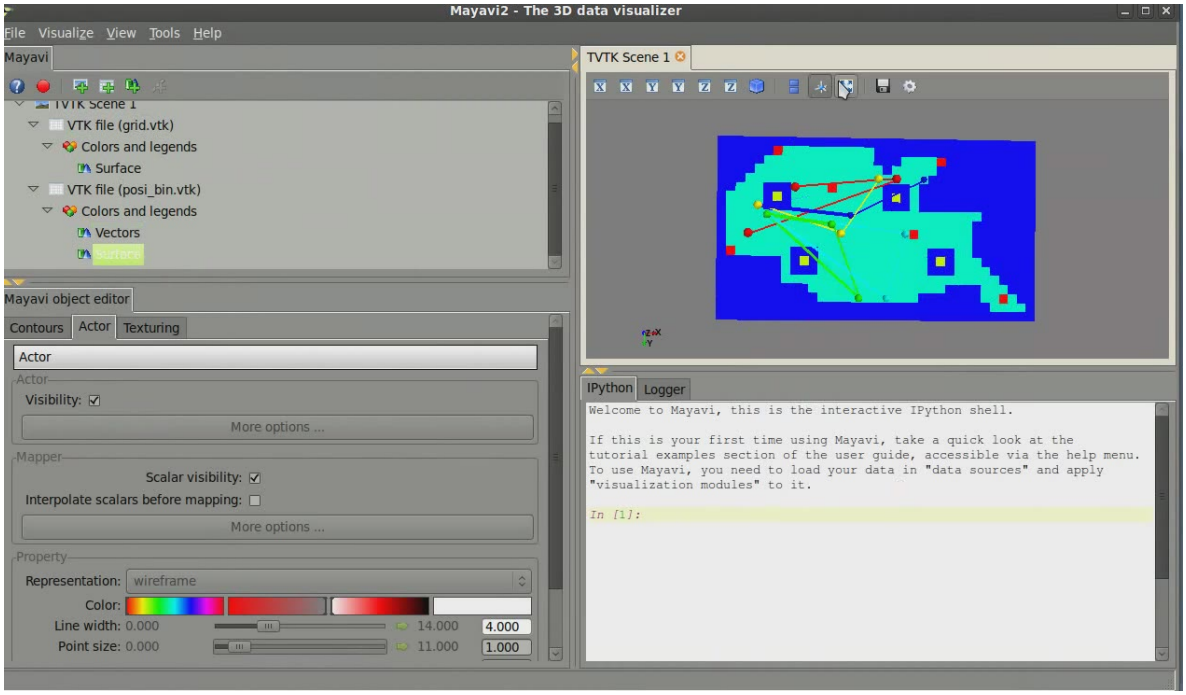


Figura 5.41: Visualizando los resultados.

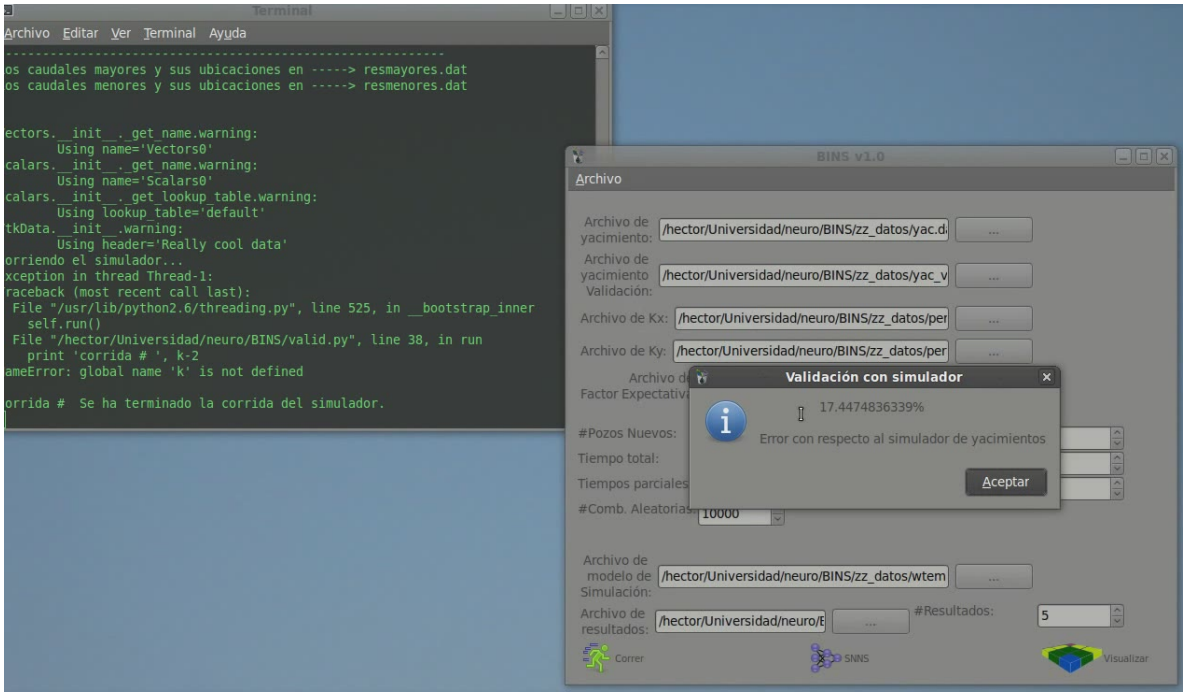


Figura 5.42: Validación de resultados.

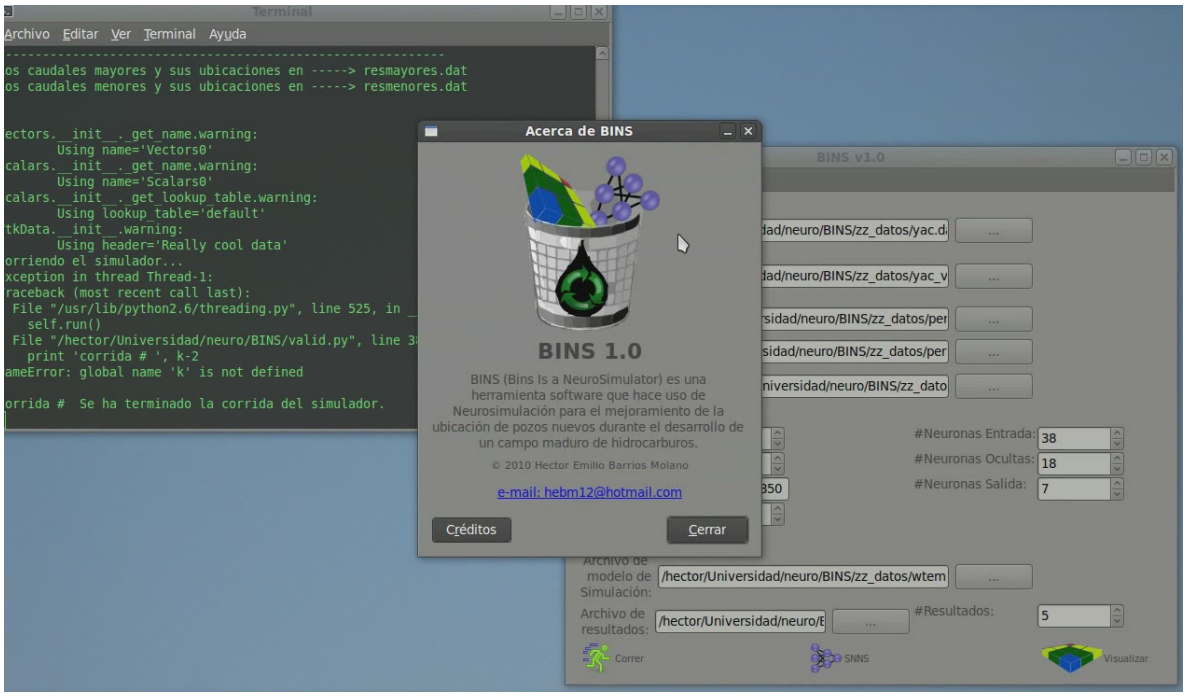


Figura 5.43: Acerca de.

(cargando bins.avi)

Figura 5.44: Video, ejecución de la herramienta. Nota: Para poder visualizar el video es necesario abrir el archivo en el sistema operativo Windows con el lector de archivos PDF Adove Reader y tener instalado el reproductor de medios Windows Media Player junto con los codecs para reproducción de Xvid.

6 CONCLUSIONES

- El presente trabajo presenta una técnica que toma lo mejor de dos “mundos” la precisión de los simuladores numéricos de yacimientos con el potencial de las redes neuronales artificiales para determinar la mejor posición de nuevos pozos en el desarrollo de campos de hidrocarburos con diferentes niveles de complejidad.
- Se parte de una metodología para realizar neurosimulación, se modifica, se propone una nueva metodología con los cambios realizados y se comprueba su eficacia por medio de varias pruebas de diferente complejidad.
- El mejor tipo de redes neuronales artificiales, es la de propagación hacia adelante, con una capa de entrada, una intermedia y una de salida, con función de transferencia sigmoidal y algoritmo de entrenamiento backpropagation.
- La introducción de heterogeneidad e irregularidadd en los modelos requiere la introducción en los datos de entrada mayor información acerca del medio o variables que den alguna clase de conocimiento “experto”, además de un mejoramiento en la rutina de entrenamiento de la red neuronal artificial.
- En este proyecto se ha podido comprobar la eficiencia y capacidades del software libre aplicado en la ingeniería de petróleos, propicia un ambiente colaborativo, se crea conocimiento, se evita la redundancia a la hora de realizar aplicaciones.
- Para el desarrollo de la herramienta software se hizo uso de la metodología de Desarrollo Rápido de Aplicaciones lo que redujo los tiempos de construcción de la herramienta, ya que a medida que se escribía el código se planeaba la herramienta, además se reutilizaron herramientas de software libre, lo que ayudó a tener una buena calidad de los resultados finales.
- Este trabajo se presenta como el primer paso en la incursión de una nueva forma de ubicar pozos en el desarrollo de un campo de hidrocarburos, en el cual se desarrolla una herramienta software en forma de prototipo, un proyecto experimental con el cual se obtuvieron buenos resultados.

7 RECOMENDACIONES

- Extender el análisis a más dimensiones (variación vertical).
- Migrar la herramienta a otras plataformas como Windows, Mac OSX, Solaris.
- En este proyecto se evaluaron redes neuronales de propagación hacia adelante con backpropagation y redes neuronales recurrentes con backpropagation through time, se recomienda que trabajos posteriores evalúen otros tipos de redes neuronales con diferentes algoritmos de entrenamiento.
- Utilizar simuladores de yacimientos y de redes neuronales artificiales diferentes a los usados en este proyecto.
- La neurosimulación no solo sirve para ubicar pozos durante el desarrollo de un campo de hidrocarburos, es por esto que se recomienda realizar pruebas con otros procesos en la industria de los hidrocarburos.

Bibliografía

[1] M. Al-Wadahi, A. Grader, and T. Ertekin. An investigation of three-phase counter-current flow using X-Ray computerized tomography and neuro-simulation modeling. In *SPE Annual Technical Conference and Exhibition*, 2000.

[2] J. Ali. Neural Networks: A New Tool for the Petroleum Industry? In *European Petroleum Computer Conference*, 1994.

[3] L. Ayala and T. Ertekin. Analysis of Gas-Cycling Performance in Gas/Condensate Reservoirs Using Neuro-Simulation. In *SPE Annual Technical Conference and Exhibition*, 2005.

[4] O. Badru. *Well-Placement Optimization Using the Quality Map Approach*. PhD thesis, STANFORD UNIVERSITY, 2003.

[5] G. Baris, R. Horne, R. Leah, and J. Rosenzweig. Optimization of well placement in a Gulf of Mexico Waterflooding Project. *SPE Reservoir Evaluation & Engineering*, 5(3):229–236, 2002.

[6] A. Centilmen, T. Ertekin, and A. Grader. Applications of neural networks in multiwell field development. In *SPE Annual Technical Conference and Exhibition*, 1999.

[7] H. Doraisamy. Methods of neuro-simulation for field development. In *SPE Rocky Mountain/Low Permeability Reservoir Regional Meeting*. SPE, 1997.

[8] H. Doraisamy, T. Ertekin, and A. Grader. Key parameters controlling the performance of neuro-simulation applications in field development. In *SPE Eastern Regional Meeting*, 1998.

[9] J. Fanchi. *Principles of applied reservoir simulation*. Gulf Professional Publishing, 2001.

[10] J. Flores, W. Gaviria, J. Lorenzon, J. Alvarez, and A. Presser. New Life for a Mature Oil Province via a Massive Infill Drilling Program. In *First International Oil Conference and Exhibition in Mexico*, 2006.

[11] F. Gorucu, T. Ertekin, G. Bromhal, D. Smith, W. Sams, and S. Jikich. A Neurosimulation Tool for Predicting Performance in Enhanced Coalbed Methane

and CO2, Sequestration Projects. In *SPE Annual Technical Conference and Exhibition*, 2005.

[12] E. Idrobo, N. Santos, and H. Vega. Aplicaciôn de algoritmos genéticos como herramienta de optimizaciôn en la ubicaciôn de pozos de desarrollo y en el trazado de los canales en yacimientos de depositaciôn fluvial. *C. T. & F Ciencia, Tecnología, Futuro*, 3(1):139–149, 2005.

[13] D. Rian and A. Hage. Automatic optimization of well locations in a north sea fractured chalk reservoir using a front tracking reservoir simulator. In *International Petroleum Conference and Exhibition of Mexico*, 1994.

[14] T. Russell and M. Wheeler. The mathematics of reservoir simulation. *SIAM, Philadelphia*, 1983.

[15] P. S. Teknica. Reservoir Simulation, 2001.

[16] A. Zell, G. Mamier, M. Vogt, N. Mache, R. H
übner, S. D
.oring, K. Herrmann, T. Soyez, M. Schmalzl, T. Sommer, et al. SNNS-Stuttgart Neural Network Simulator, User Manual, Version 4.2, 2000.

A PARTICIPACIONES

Durante el desarrollo del presente trabajo se tuvo la oportunidad de participar en los siguientes congresos nacionales e internacionales y publicaciones:

1. BARRIOS MOLANO, Hector Emilio; SANTAFÉ RANGEL Elkin Rodolfo. (Grupo de Investigación en Tecnologías Alternativas para Hidrocarburos) *Uso de la Neurosimulación en el Posicionamiento de Pozos para el Desarrollo de un Campo de Hidrocarburos*. XI Semana Técnica Internacional de Ingeniería de Petróleos. Bucaramanga. Colombia. Octubre 2008.

2. BARRIOS MOLANO, Hector Emilio. (Grupo de Investigación en Tecnologías Alternativas para Hidrocarburos) *Use of Neurosimulation in Well Placement for the Development of a Hydrocarbon Field*. SPE Latin American Regional Student Paper Contest. Cartagena. Colombia. Mayo 2009.

3. BARRIOS MOLANO, Hector Emilio. (Grupo de Investigación en Tecnologías Alternativas para Hidrocarburos) *Use of Neurosimulation in Well Placement for the Development of a Hydrocarbon Field*. (Poster) X Simposio Bolivariano Exploración Petrolera en las Cuencas Subandinas. Cartagena. Colombia. Julio 2009.

4. BARRIOS MOLANO, Hector Emilio; SANTAFÉ RANGEL, Elkin Rodolfo. (Grupo de Investigación en Tecnologías Alternativas para Hidrocarburos) *Uso de Neurosimulación en el Posicionamiento de Pozos Durante el Desarrollo de un Campo de Hidrocarburos*. Fuentes, El Reventón Energético. Segundo semestre del 2009.

B REQUISITOS PARA CORRER LA HERRAMIENTA

A continuación se listan los requisitos para poder correr la herramienta software desarrollada en este proyecto.

- Sistema operativo: GNU/Linux
- Python 2.6[a]
- wine[b]
- numpy y scipy[c]
- Gnuplot[d]
- Gnuplot.py[e]
- pyVTK[f]
- wxpython[g]
- Mayavi2[h]

[a] http://www.python.org/download/
[b] http://www.winehq.org/download/
[c] http://numpy.scipy.org/
[d] http://www.gnuplot.info/
[e] http://gnuplot-py.sourceforge.net/
[f] http://cens.ioc.ee/projects/pyvtk/
[g] http://www.wxpython.org/
[h] http://code.enthought.com/projects/mayavi/

C MANUAL DE LA HERRAMIENTA

El software desarrollado en este proyecto se llama BINS (Bins Is a NeuroSimulator) en este anexo se muestra su uso. En la Figura C.1 se muestra la ventana principal del programa.

El primer paso es cargar los archivos iniciales, estos son:

- Archivo de información del yacimiento con pozos de entrenamiento.
- Archivo de información del yacimiento con pozos de validación.
- Archivo de permeabilidad en X.
- Archivo de permeabilidad en Y.
- Archivo de factor de expectativa.
- Archivo de modelo de simulación

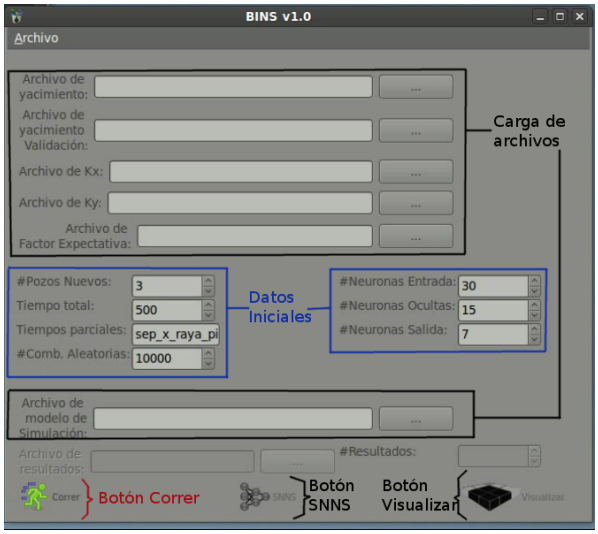


Figura C.1: Ventana principal de BINS

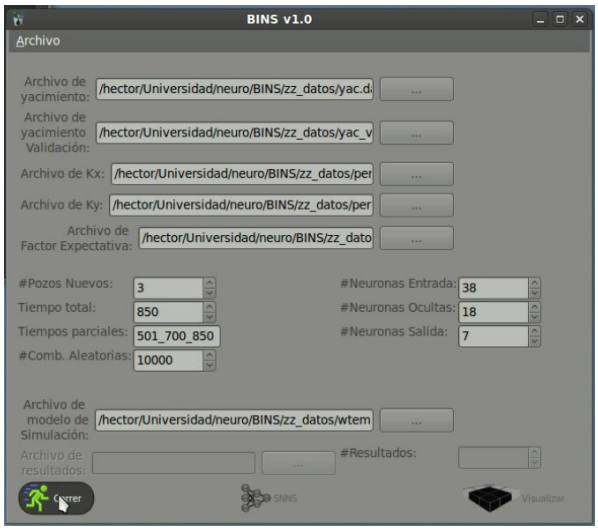


Figura C.2: Ventana principal con datos iniciales

Luego se ingresan los siguientes datos adicionales:

- Número de pozos nuevos (por ahora solo sirve para tres).
- Tiempo total en días.
- Tiempos parciales en días, separados por raya piso (_)
- Número de combinaciones aleatorias para ser evaluadas por la red neuronal artificial.
- Número de neuronas de entrada.
- Número de neuronas intermedias.
- Número de neuronas de salida.

En este punto se oprime el botón correr, la Figura C.2 se muestra la ventana principal con los datos iniciales.

Cuando se oprime el botón correr es importante no mover el cursor o el teclado mientras el simulador numérico de yacimientos está corriendo, ya que el programa mueve el cursor y el teclado para automatizar las corridas. Cuando se termina de realizar las corridas, el botón SNNS se habilita (Figura C.3), al oprimir éste botón se abre el simulador de redes neuronales SNNS (Figura C.4).

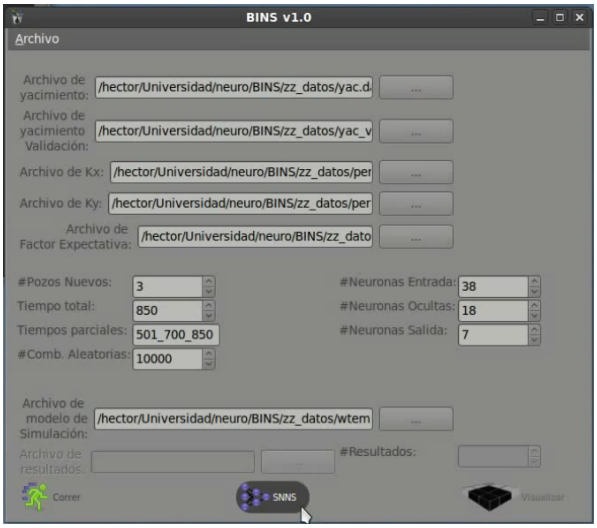


Figura C.3: Ventana principal luego de las simulaciones

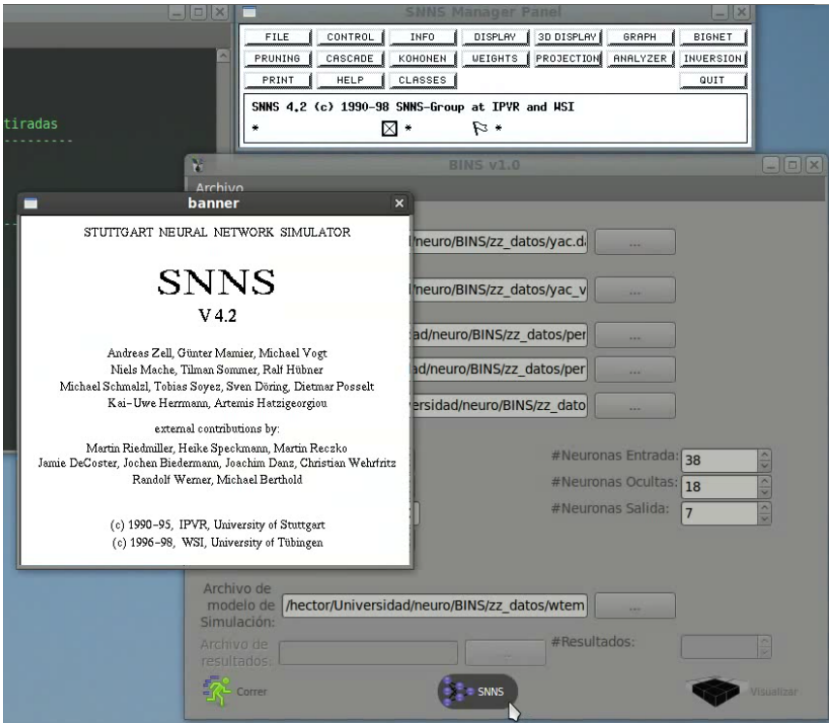


Figura C.4: SNNS

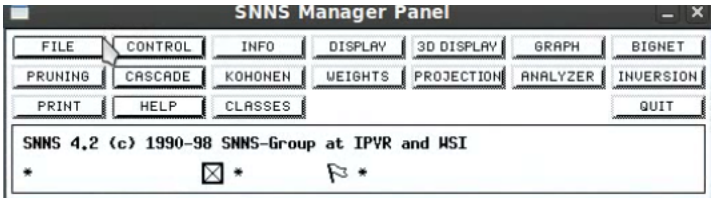


Figura C.5: Oprimir el botón File

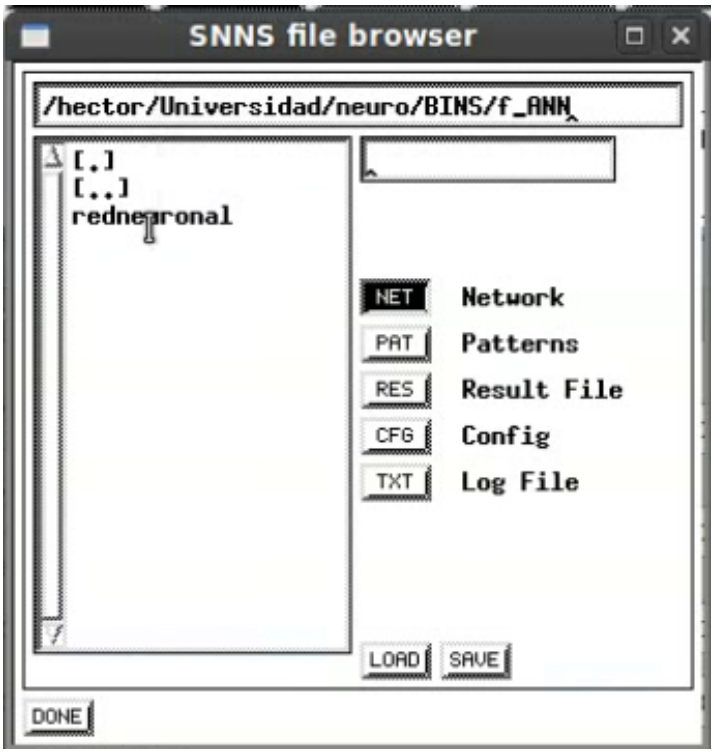


Figura C.6: Cargar la red neuronal, click en load

En las Figuras C.5 a C.11 se muestra como cargar los datos a SNNS, entrenar la red neuronal artificial y guardar los resultados evaluados. Para mayor información acerca del manejo de SNNS leer su manual [16]

Una vez se cierra SNNS se habilita la entrada de datos para visualizar, se ingresa el archivo de resultados generado por SNNS y el número de las mejores combinaciones que se quieran visualizar, cuando se oprime el botón Visualizar se abre Mayavi2.

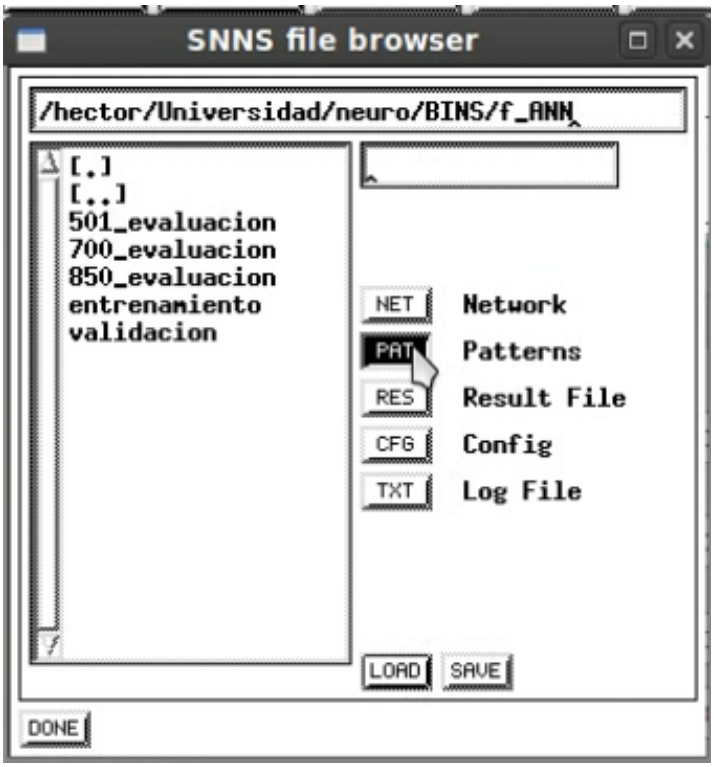


Figura C.7: Cargar los archivos .pat para entrenamiento, validación y evaluación

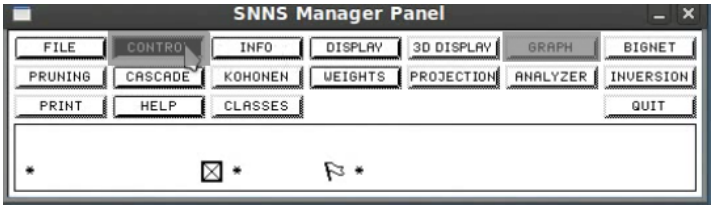


Figura C.8: Oprimir los botones Control y Graph

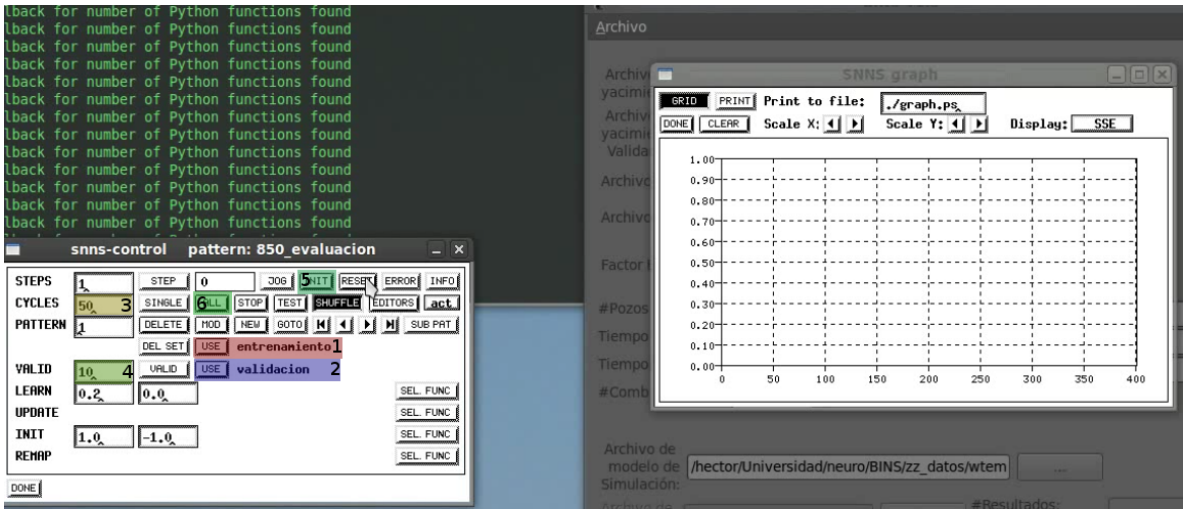


Figura C.9: Para entrenar la RNA escoger el archivo de entrenamiento en (1) haciendo click en Use, de la misma forma se escoge el archivo de validación en (2), se escribe el número de ciclos de entrenamiento (3), y de validación (4), se da click en Init (5) para inicializar aleatoriamente los pesos sinápticos, luego se oprime el botón All (6) tantas veces como el error de aprendizaje de la RNA sea tan bajo como se requiera.

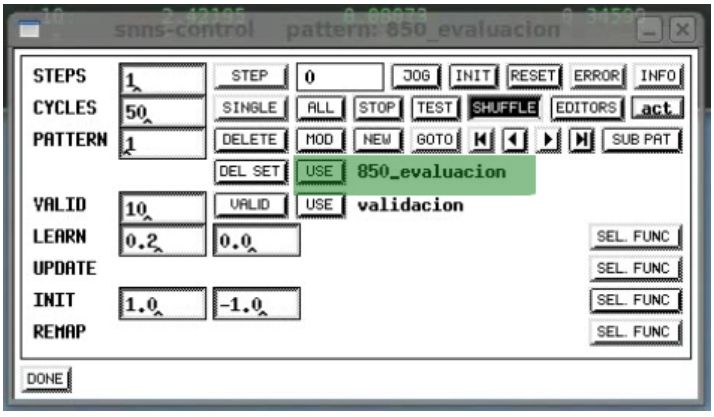


Figura C.10: Para evaluar los resultados, escoger el archivo de evaluación en el cuadro verde por medio del botón Use

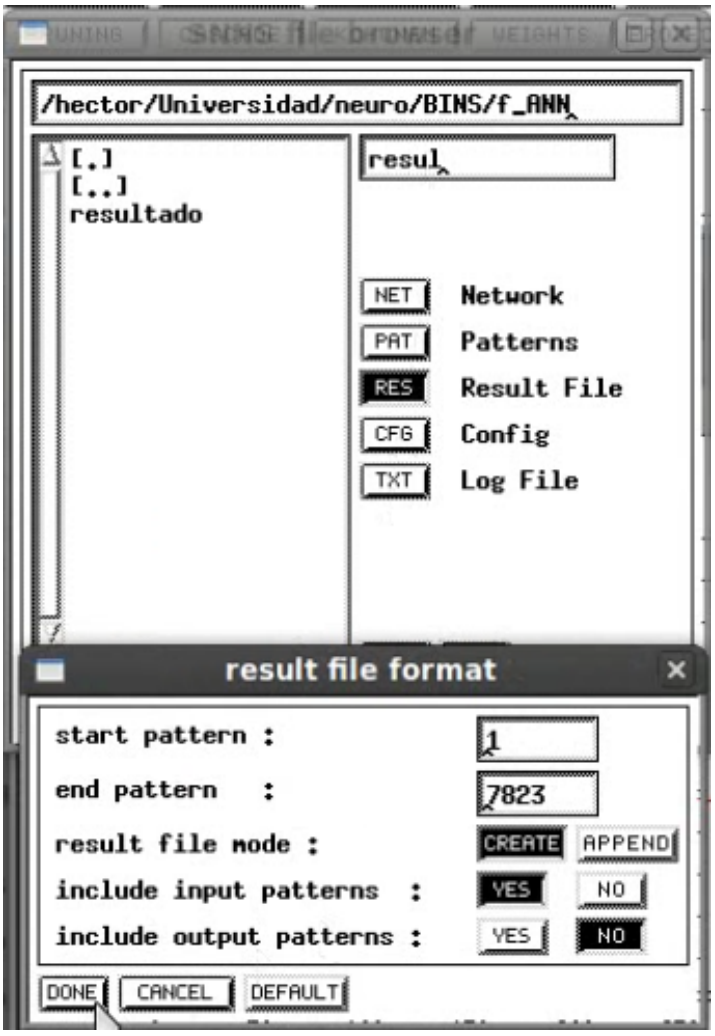


Figura C.11: Oprimir el botón File, oprimir el botón Res, oprimir el botón Done y oprimir el botón Done en la ventana emergente. Cerrar SNNS

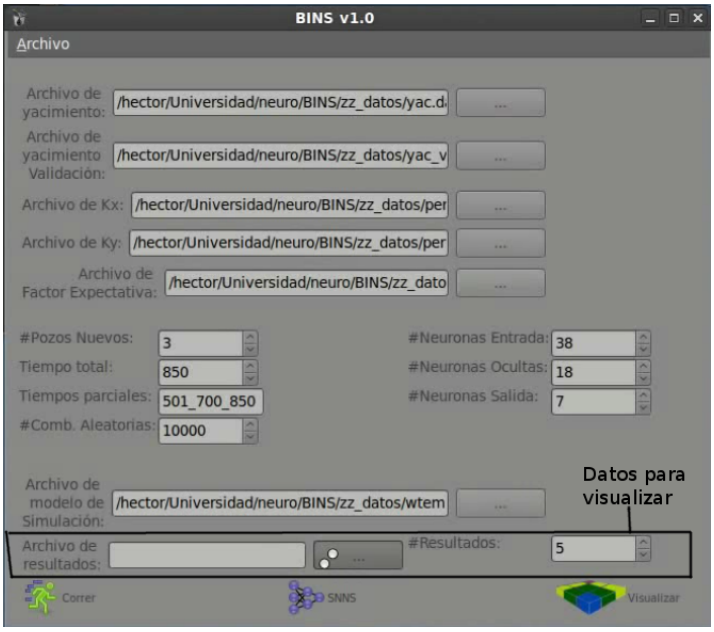


Figura C.12: Datos para visualizar.

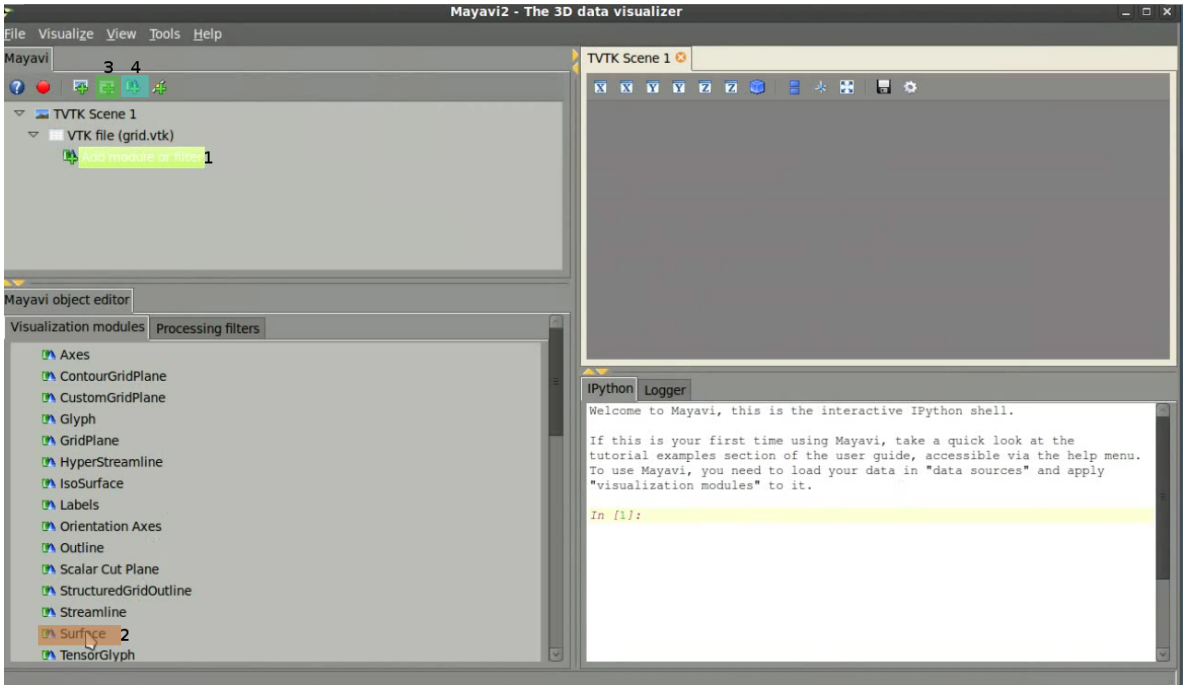


Figura C.13: Mayavi2. Click en (1), doble click en (2), luego abrir el archivo generado oprimiendo el botón en (3), seleccionar el archivo “posi_bin.vtk”, oprimir (4) y seleccionar vectores.

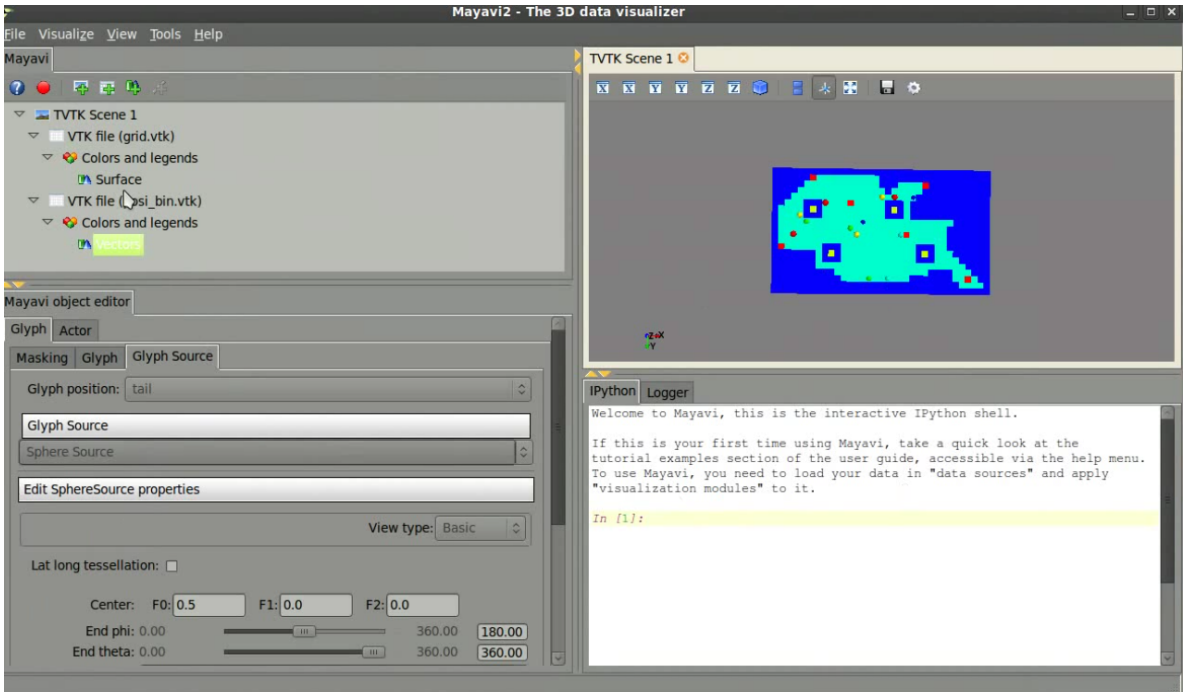


Figura C.14: Mayavi2, visualización completa.

Luego de cerrar Mayavi2 el simulador numérico de yacimientos es ejecutado una vez más, de nuevo no se debe mover el cursor o el teclado, luego un diálogo aparece mostrando el error de los resultados respecto del simulador numérico de yacimientos (Figura C.15).

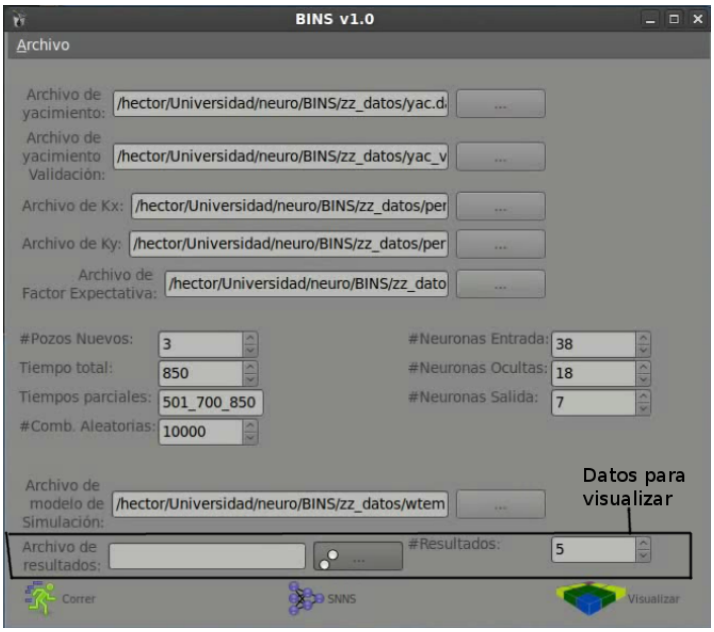


Figura C.15: Verificación.

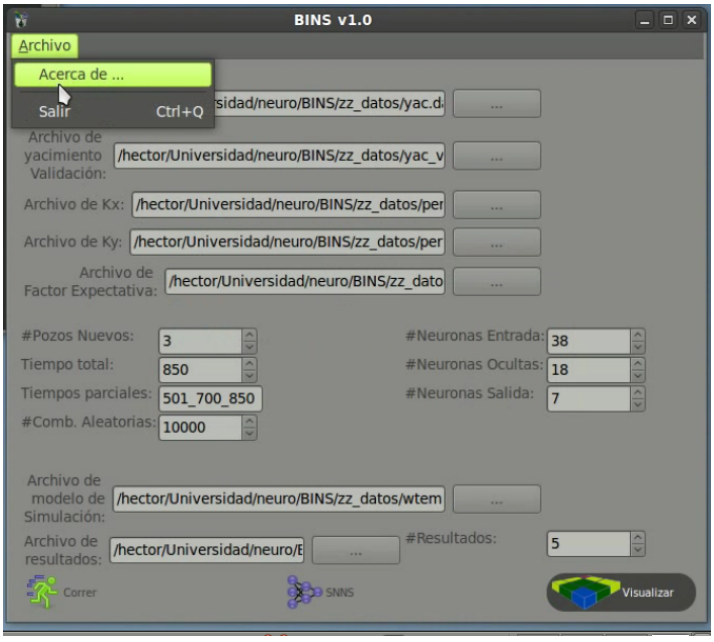


Figura C.16: Menú archivo.

Figura C.17: Acerca de.