UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Aplicación de Machine Learning para la Interpretación Automática de Pruebas de Presión en Pozos Petroleros

TESIS

Que para obtener el título de

Ingeniero Petrolero

P R E S E N T A

Guillermo Toledo Barrón

DIRECTOR DE TESIS

Dr. Víctor Leonardo Teja Juárez

Ciudad Universitaria, Cd. Mx., 2022

RECONOCMIENTOS

Al Dr. Víctor Leonardo Teja Juárez:

Por haberme brindado la oportunidad y el honor de trabajar con él en esta tesis, ofreciendo parte de su invaluable tiempo y recursos tanto económicos como físicos.

Al Dr. Fernando Samaniego Verduzco:

Por su apoyo durante gran parte de mi formación profesional, así como de permitirme formar parte de grandes experiencias que fueron clave en la construcción de la persona que soy hoy en día.

Al Dr. Rodolfo Gabriel Camacho Velásquez:

Por los conocimientos que me transmitió en sus clases y por las recomendaciones que aportó en la revisión de este trabajo.

Al Dr. Víctor Hugo Arana Ortiz:

Por todos los temas de novedad a los que me invita incluso después de haber terminado su curso, y por sus observaciones en la revisión de este trabajo.

Al Ing. Israel Castro Herrera:

Por aportar significativamente con los datos de pruebas de presión reales que se muestran en este trabajo, así como sus correcciones en la revisión de este trabajo.

DEDICATORIA

A mis padres, Roberto y María, que siempre han mostrado apoyo incondicional a cualquier decisión que haya tomado sobre mi futuro. Gracias por ser mis primeros verdaderos amigos, por enseñarme el valor del trabajo y ser ejemplos de como llevar una vida plena y feliz. Los amo.

A mis hermanos, Gabriela y Carlos, que siempre han escuchado las locuras que se le ocurren a su hermano mayor con una atención sobre acogedora, les debo la cordura de la que hoy dispongo. Gracias por hacerme sentir amado.

Al amor de mi vida, Sara, que jamás a puesto un dedo de presión sobre mí, que ha creído en mis capacidades incluso cuando ni yo mismo las veía claras.

A mi mejor amigo, José, que a pesar de que nuestras vidas no tienen nada que ver ya una con la otra, ha sabido ser ese defensor que separa la soledad de la compañía cuando la oscuridad de la tristeza abruma.

A Omar, mi único verdadero amigo de la prepa, que siempre me hizo sentir comprendido en medio de un mar de sombras iguales, y que ha sabido sobrellevar los lados más fríos de mi persona.

A mis compañeros de carrera, Nequiz, Fernando, David, Gaby, por expandir mi amor por el conocimiento, por apoyar a un foráneo en medio de la jungla de acero, y por tenerlo en mente cuando la oportunidad se asoma.

A mis compañeros de equipo en el petrobowl, Busti, Juan, Dieguito, Kevin, y nuestro coach Noé, por empujarme al límite, mostrarme lo que el esfuerzo es capaz de hacer y competir conmigo en el hobby más divertido que puede tener una facultad.

A mis primos Alex, Pao y Karla, por aguantar a la peor versión de mí durante mis primeros años en la ciudad, es gracias a su hospitalidad que mi vida es como es hoy, siempre los tengo en mente.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

El análisis de pruebas de presión es el proceso mediante el cual se obtiene información acerca del yacimiento, a través del estudio de la respuesta de la presión causada por un cambio en el gasto (flujo volumétrico) de producción. De estas pruebas se pueden obtener varios datos como estimaciones de la presión inicial del yacimiento o identificar regímenes de flujo que se presentan durante la producción del mismo, pero los más básicos, y posiblemente los más importantes son la permeabilidad (*k*), el daño a la vecindad del pozo (S) y el coeficiente de almacenamiento del pozo (C).

Se propone la aplicación de algoritmos de *Machine Learning* para la resolución de estos problemas. Estos algoritmos son capaces de, mediante la observación de tendencias en los datos, encontrar las reglas que controlan un fenómeno determinado y que, gracias a ello cuentan con un gran poder predictivo.

En este trabajo se optó por entrenar una red neuronal por su gran capacidad para resolver problemas de regresión de alta complejidad. Una red neuronal tiene que entrenarse y después optimizada mediante ciertos hiperparámetros (número de capas, nodos en cada capa, tasa de aprendizaje, etc.)

Para su entrenamiento, la red necesita de una gran cantidad de pruebas de presión conjuntamente con sus soluciones (permeabilidad, daño y almacenamiento), para que esta sea capaz de identificar un patrón dentro de estos datos. Estas pruebas de presión se generaron de manera sintética, partiendo de la ecuación que dio origen a la curva tipo de Gringarten, mediante la cual se crearon 10,000 pruebas a partir de distintas combinaciones de distintos rangos de permeabilidad, daño y almacenamiento, y con ello se obtuvo el conjunto de datos para entrenar nuestra red neuronal.

Se entrenaron 1458 redes distintas variando los hiperparametros que las formaban, y se utilizó R2 y el error cuadrático medio para comparar y determinar qué modelo en específico es el que mejor predecía los valores de permeabilidad, daño y almacenamiento.

Una vez obtenida la estructura de la red neuronal, y de haberla entrenado, se creó un código que se encarga de pre-procesar los datos de cualquier prueba ajena al entrenamiento de la red, para que así el modelo pueda interpretarla de manera automática, es decir, la obtención de parámetros de permeabilidad, daño y almacenamiento. Este pre-procesamiento incluye, la toma de puntos relevantes de la curva de la prueba de presión y normalización de estos datos.

El modelo cuenta con una precisión de R2 del 0.956. Este modelo es capaz de interpretar pruebas que cuentan con cierto parecido a lo “convencional”, o dicho de otra manera, con el comportamiento de flujo radial, cumpliendo con el objetivo de haber desarrollado una herramienta computacional portable y sencilla aplicando Machine Learning con el fin, de interpretar automáticamente las pruebas de presión de pozos y así obtener datos de permeabilidad, daño y almacenamiento.

ABSTRACT

Well test analysis is the process by which reservoir information is obtained, studying the pressure response caused by a change in production rate (volumetric flow). Several data can be obtained from these tests, such as estimates of the reservoir initial pressure or identifying flow regimes that occur during its production, but the most basic, and possibly the most important, are permeability (k), damage to the vicinity of the well (S) and wellbore storage (C).

The application of Machine Learning algorithms is proposed to solve these problems. These algorithms are capable of, by observing trends in the data, finding the rules that control a given phenomenon and, thanks to this, have great predictive power.

In this work, it was decided to train a neural network due to its great capacity to solve highly complex regression problems. A neural network has to be trained and then optimized using certain hyperparameters (number of layers, nodes in each layer, learning rate, etc.)

For its training, the network needs a large number of pressure tests together with their solutions (permeability, damage and storage), so that it is able to identify a pattern within this data. These pressure tests were generated synthetically, starting from the equation that gave rise to the Gringarten type curve, through which 10,000 tests were created from different combinations of different ranges of permeability, damage and wellbore storage, this let the dataset to train our neural network.

1458 different networks were trained varying the hyperparameters that formed them, and R2 and the mean square error were used to compare and determine which specific model is the one that best predicts the values of permeability, damage and wellbore storage.

Once the structure of the neural network was obtained, and having trained it, a code was created with the responsibility of pre-processing the data of any test outside the training of the network, so that the model can interpret it automatically, that is, obtaining parameters such as permeability, damage and wellbore storage. This pre-processing includes taking relevant data points of the pressure test curve and normalizing these data.

The model has an R2 precision of 0.956. This model is capable of interpreting tests with a certain resemblance to the "conventional" behavior or in other words with radial flow behavior, fulfilling the objective of having developed a portable and simple computational tool applying Machine Learning in order to automatically interpret the well pressure tests, obtaining on permeability, damage and wellbore storage.

Capítulo 1. Introducción

Desde los inicios de la industria petrolera se ha buscado mejorar la forma en la que se exploran y evalúan valiosos prospectos valiosos a explotar. Ciencias como la geología y la petrología eran, y son hasta el día de hoy, de suma importancia en el entendimiento de los yacimientos de hidrocarburos. La necesidad de crear herramientas que permitan analizar algo que no podemos ver va en incremento, puesto que conforme los avances tecnológicos permitieron llegar a yacimientos cada vez más profundos, herramientas como lo son los registros geofísicos, estudios de sismología y, el enfoque de este trabajo, el análisis de pruebas de presión, se volvieron rápidamente rutinarios en el flujo de trabajo de los profesionales que buscaban comprender lo que sucedía en el subsuelo.

Desde su introducción han surgido diversas técnicas para la interpretación de pruebas de presión, análisis semilogarítmicos o análisis con curvas tipo por mencionar un par, y cada una de estas se ha ido profundizando hacia la resolución de problemas cada vez más complejos (pruebas de presión en yacimientos naturalmente fracturados, en pozos hidráulicamente fracturados, pruebas de interferencia, entre muchas otras más).

Actualmente las herramientas empleadas por *softwares* de interpretación de pruebas de presión se basan en la metodología seguida por estos análisis manuales. En este trabajo se propone la aplicación de algoritmos de *Machine Learning (ML)* para la resolución de estos problemas de manera automática; más específicamente, la obtención de datos esenciales como lo son la permeabilidad, el daño en la vecindad del pozo y el almacenamiento del pozo. Estos algoritmos han probado ser capaces de predecir con éxito resultados de problemas de alta complejidad, como lo pueden ser la detección de enfermedades en el ámbito clínico, o incluso la propia clasificación de yacimientos con el objetivo de determinar qué tipo de yacimiento se está a punto de analizar.

Durante el desarrollo de esta investigación, en el capítulo 2 se proporciona primero una introducción de lo que es la interpretación de pruebas de presión, discutiendo sobre conceptos básicos que rodean la ingeniería de yacimientos, mostrando a la par distintas técnicas empleadas para realizar esta tarea, así como ejemplos de la aplicación de dichas técnicas. Después en el capítulo 3 se presentan los conceptos básicos que abren las puertas del *ML*, específicamente se detalla el funcionamiento de las redes neuronales. En el capítulo 4 se muestra a detalle cuál es el flujo de trabajo que se siguió para el desarrollo de esta herramienta computacional, así como una explicación relacionada con su funcionamiento. Finalmente, en el capítulo 5 se expondrán los resultados obtenidos, del propio modelo, así como en comparación con otros algoritmos de *ML* distintos a las redes neuronales, para así poder concluir en el capítulo 6 donde se plantean posibles pasos a seguir en trabajos futuros con el objetivo de profundizar esta investigación.

1.1 OBJETIVO GENERAL

Desarrollar una herramienta computacional portable y sencilla aplicando *Machine Learning (ML)* con el objetivo de interpretar y obtener valores de permeabilidad, daño y almacenamiento de manera automática

a partir de las pruebas de presión de pozos, para aplicarlo a yacimientos de hidrocarburos mexicanos en proceso de explotación.

1.2 OBJETIVOS PARTICULARES

- Investigar los conceptos básicos que intervienen para la interpretación adecuada de pruebas de presión.

- Desarrollar una herramienta aplicando *Machine Learning* capaz de interpretar las pruebas de presión mediante parámetros seleccionados de entrada, obtenidos de la literatura y de una generación sintética de estos mismos modelos del tipo homogéneo.

- Analizar el comportamiento del código desarrollado mediante datos o pruebas basados en la literatura.

Capítulo 2. Interpretación de las Pruebas de Presión

Antes de comenzar con los conceptos fundamentales de la interpretación de pruebas de presión, cabe mencionar que esta tesis estará enfocada en el escenario más simple de todos: un único pozo produciendo a una sola fase liquida a un gasto constante, de una única capa de espesor constante en un yacimiento homogéneo e isotrópico, en donde el flujo es primordialmente radial. Esto debido a que en la literatura es este problema el primero en solucionarse, además de que problemas más complejos parten de este mismo caso como base para extender una metodología más compleja.

2.1 INTRODUCCIÓN A LA INTERPRETACIÓN DE PRUEBAS DE PRESIÓN.

El análisis de pruebas de presión es el proceso mediante el cual se obtiene información acerca del yacimiento, a través del estudio de la respuesta de la presión en el fondo del pozo causada por un cambio en el gasto de producción en la superficie.

Es importante entender que la interpretación de pruebas de presión es un problema inverso, es decir, no hay una solución única. En un problema inverso podemos conocer los datos de entrada (el gasto de producción en este caso) y los datos de salida (la presión), y al querer encontrar el modelo desconocido que se ajuste a dichos datos se concluiríaque hay varios modelos que lo logran.

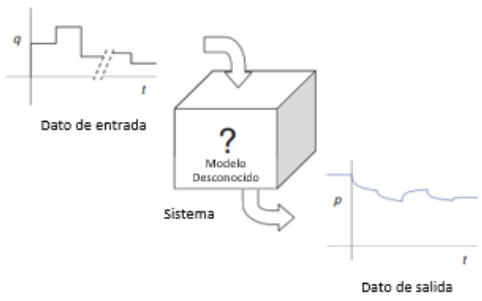

Ilustración 1 Problema Inverso (de Applied Well Test Interpretation, 2013).

Los métodos para interpretar pruebas de presión pueden clasificarse en las 3 categorías siguientes:

- Métodos de línea recta.
- Análisis de curva tipo.
- Simulación numérica con "*history match*"

En esta tesis es de interés realizar un flujo de trabajo en el que los tres métodos se complementen, con el objetivo de obtener una interpretación lo más acertada posible, este flujo de trabajo ayudará como punto de comparación entre lo "convencional" y el modelo de ML aquí presentado más adelante.

2.2 FLUJO DE FLUIDOS EN MEDIOS POROSOS

Antes de tratar directamente con la interpretación de pruebas es conveniente revisar algunos conceptos importantes sobre el flujo de fluidos en medios porosos.

Ley de Darcy: describe el movimiento horizontal de los fluidos a través de un medio poroso bajo la influencia de un gradiente de presión y esta expresada como:

$$q = \frac{kA}{\mu}\frac{\Delta p}{L} \tag{1}$$

donde q se refiere al gasto, k es la permeabilidad, A es el área transversal al flujo, μ se refiere a la viscosidad del fluido, y Δp y L se refieren al cambio de presión y la longitud respectivamente.

Ecuación de difusividad: Describe el flujo transiente de un fluido ligeramente compresible a través de un medio poroso. Esta ecuación es derivada de la combinación de tres principios físicos: La ley de Darcy, una ecuación de continuidad que suele ser un balance de materia, y una ecuación de estado para tomar en cuenta los efectos de trabajar con un fluido ligeramente compresible. Esta ecuación puede expresarse en coordenadas radiales como:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial p}{\partial r}\right) = \frac{\Phi\mu c_t}{k}\frac{\partial p}{\partial t} \tag{2}$$

donde c_t representa la compresión total entre el yacimiento y el fluido dentro de este.

Solución de línea fuente: Es una de las soluciones más importantes a la ecuación de difusividad, la cual se basa en la integral exponencial E_i; describe el flujo radial transitorio (sin estar afectado por ninguna frontera (IARF)). Esta solución puede expresarse matemáticamente como:

$$p(r,t) = p_i + \frac{70.6qB\mu}{kh}E_i\left(-\frac{948\varphi\mu c_t r^2}{kt}\right) \tag{3}$$

donde r y t son el radio y el tiempo, respectivamente, al cual se requiere calcular la presión.

Radio de investigación: Es un concepto que nos permite entender el movimiento del transiente de presión a través del yacimiento. Se deriva de la solución de línea fuente, asumiendo un valor de la función E_i igual a la unidad (1) y resolviendo para la distancia:

$$r_i = \sqrt{\frac{kt}{948\varphi\mu c_t}} \tag{4}$$

Daño o estimulación (skin factor): Como resultado de las operaciones llevadas a cabo para la perforación y/o terminación de un pozo, la región cercana a este se puede ver alterada. El concepto de daño se refiere a que hay un decremento de la permeabilidad en la vecindad del pozo en comparación con el resto del

yacimiento como resultado de las actividades de perforación. El caso contrario es el de un proyecto de estimulación, el cual busca remediar el daño, e incluso mejorar la permeabilidad a un valor mayor que se tenía previo a su afectación.

$$s = \left(\frac{k}{k_a} - 1\right) ln\ \left(\frac{r_a}{r}\right) \tag{5}$$

donde el sufijo a se refiere a las propiedades la zona alterada del yacimiento. Es de la expresión anterior que podemos obtener la ecuación para calcular la caída de presión adicional debida al daño:

$$\Delta p_s = \frac{141.2qB\mu}{kh} s \tag{6}$$

Almacenamiento: Este término toma en cuenta el volumen finito del pozo entre la cara del yacimiento y el punto de medición del gasto; es decir, sin este término se consideraría que el flujo sube instantáneamente de cero al gasto constante final al inicio de una prueba de decremento. Es importante mencionar que existen dos situaciones principales donde se presenta el efecto de almacenamiento. En un pozo lleno de un fluido de una sola fase, y en un pozo con un nivel de fluido ascendente o descendente.

2.3 ANÁLISIS SEMILOG DE FLUJO RADIAL

En una **prueba de decremento** a gasto constante, el pozo es previamente cerrado; en este punto el yacimiento se encuentra a una presión uniforme. El pozo después se produce a un gasto constante y se mide la presión de fondo como una función del tiempo conforme la presión decae.

La respuesta de presión de un pozo produciendo a un gasto constante en un yacimiento “infinito” (pozo el cual no ha sufrido los efectos ocasionados porque el transiente de presión haya tocado las fronteras externas del yacimiento) está descrita por la ecuación:

$$p_{wf}(t) = p_i - \frac{162.6qB\mu}{kh}\left[\log t + \log\left(\frac{k}{\varphi\mu c_t r_w^2}\right) - 3.23 + 0.869s\right] \tag{7}$$

donde p_i es presión inicial a la que se encuentra el yacimiento.

Si se compara dicha ecuación con la de una línea recta en una gráfica de presión contra el logaritmo del tiempo, se tiene que:

$$m = \frac{162.6qB\mu}{kh} \tag{8}$$

$$b = p_{1hr} = p_i - \frac{162.6qB\mu}{kh}\left[\log\left(\frac{k}{\varphi\mu c_t r_w^2}\right) - 3.23 + 0.869s\right] \quad (9)$$

donde m es la pendiente de la recta y b la ordenada al origen.

Es a partir de esta representación de la ecuación en línea recta que se puede resolver para la permeabilidad y el daño, y con la ayuda de una gráfica de presión contra tiempo en escala semilogarítmica es posible obtener estos valores. Por lo que el flujo de trabajo para la interpretación de una prueba de decremento con análisis semilog es:

1. Graficar la presión de fondo fluyente contra el tiempo en escala semilogarítmica.
2. Identificar los datos que caen en IARF con una línea recta.
3. Encontrar la pendiente y la ordenada al origen.
4. Calcular permeabilidad, daño y radio de investigación:

$$k = -\frac{162.6qB\mu}{mh} \quad (10)$$

$$S = 1.151\left[\frac{p_i - p_{1hr}}{|m|} - \log\left(\frac{k}{\varphi\mu c_t r_w^2}\right) + 3.23\right] \quad (11)$$

$$r_i = \sqrt{\frac{kt}{948\varphi\mu c_t}} \quad (12)$$

Nota: Si el gasto varía poco y de manera suave se puede realizar una normalización de la presión con respecto del gasto, el procedimiento sigue siendo el mismo.

En una **prueba de incremento** el pozo ha estado produciendo a gasto constante por un tiempo determinado y después ocurre el cierre. La presión de fondo de cierre se mide como una función del tiempo de cierre conforme la presión va en aumento. Este tipo de análisis requiere del uso del principio de superposición en tiempo y existen varios métodos en la literatura para hacerlo.

El método Miller-Dyes-Hutchinson (MDH) asume que el tiempo de producción es mucho mayor que el de cierre, por lo que obtiene la ecuación (13):

$$p_{ws}(\Delta t) = p_{wf} + \frac{162.6qB\mu}{kh}\left[\log\Delta t + \log\left(\frac{k}{\varphi\mu c_t r_w^2}\right) - 3.23 + 0.869s\right] \quad (13)$$

donde p_{wf} se refiere a la presión de fondo fluyendo al momento del cierre.

En este método se puede repetir la comparativa con la ecuación de la línea recta llevada a cabo en la prueba de decremento. Entonces es fácil deducir que un flujo de trabajo quedaría como:

1. Graficar la presión de fondo de cierre contra el tiempo de cierre en una escala semilogarítmica.
2. Identificar una línea recta que representa el IARF, extenderla y encontrar su pendiente y ordenada al origen.
3. Calcular la permeabilidad y el daño:

$$k = \frac{162.6qB\mu}{|m|h} \tag{14}$$

$$S = 1.151\left[\frac{p_{ihr} - p_{wf}}{|m|} - \log\left(\frac{k}{\varphi\mu c_t r_w^2}\right) + 3.23\right] \tag{15}$$

Por otro lado, el método de Horner introduce un término conocido como el "tiempo de Horner", que sustituye al término del tiempo de cierre en la ecuación (7) por lo que el flujo de trabajo no cambia realmente en comparación con una prueba de decremento.

$$p_{ws}(\Delta t) = p_i - \frac{162.6qB\mu}{kh}\log\left(\frac{t_p + \Delta t}{\Delta t}\right) \tag{16}$$

Nota: Dada la definición del tiempo de Horner, en un gráfico semilog, el tiempo incrementa de la derecha hacia la izquierda. Además, si se considera que el yacimiento se comporta como "infinito" desde el comienzo del flujo hasta el final de la prueba, la ordenada al origen proporciona una estimación de la presión inicial del yacimiento.

Otro método es el de Agarwal, quien introdujo el tiempo equivalente para el análisis de pruebas de incremento, obteniendo la ecuación siguiente:

$$p_{ws}(\Delta t) = p_{wf} + \frac{162.6qB\mu}{kh}\left[\log \Delta t_e + \log\left(\frac{k}{\varphi\mu c_t r_w^2}\right) - 3.23 + 0.869s\right] \tag{17}$$

donde Δt_e es el tiempo equivalente y se expresa como:

$$\Delta t_e = \frac{t_p \Delta t}{t_p + \Delta t} \tag{18}$$

De la misma manera que se discutió para el método de Horner, en esta técnica se sigue el mismo flujo de trabajo que en el método de MDH, únicamente sustituyendo el tiempo de cierre por el tiempo equivalente de Agarwal, las ecuaciones (10) a (12) siguen aplicando.

Aunque ya se mencionó cómo obtener un valor estimado de la presión inicial del yacimiento en el método de Horner, existen otros dos métodos para cuando el pozo alcanza el flujo pseudo-estacionario antes de que empiece el incremento, además de que se requiere conocer el factor de forma del yacimiento C_A.

Dos de estos métodos siguen el mismo procedimiento, pero su aplicación depende con que grafica se esté trabajando. El método de Dietz se utiliza en gráficas de MDH, mientras que el método de Ramey-Cobb se usa en gráficas de Horner. Ambos métodos usan el flujo de trabajo siguiente:

1. Calcular el tiempo al que la presión promedio del área de drene puede leerse directamente de la gráfica semilog:

$$\Delta t_{\bar{p}} = \frac{\varphi \mu c_t A}{0.0002637 k C_A} \tag{19}$$

$$\left(\frac{t_p + \Delta t}{\Delta t}\right)_{\bar{p}} = \frac{0.0002637 k C_A t_p}{\varphi \mu c_t A} \tag{20}$$

2. Extrapolar la línea recta en el gráfico correspondiente al tiempo calculado, y leer dicha presión.

Y por último existe un método extra de MDH que hace uso de gráficas, y que no tiene la limitación de haber tenido que alcanzar el flujo pseudo-estacionario.

2.4 ANÁLISIS DE CURVAS TIPO

Las curvas tipo son soluciones a la ecuación de difusividad en su forma adimensional mostradas de manera gráfica, y que generalmente están en una escala log-log del cambio de presión contra el tiempo.

Su uso se resume en que una gráfica conteniendo los datos de presión de pozo a estudiar es colocada encima de la curva tipo, y moviéndola horizontal y verticalmente se encuentra una semejanza, un “match”. Esos puntos en los que se encuentra la semejanza son después utilizados para el cálculo de permeabilidad, coeficiente de almacenamiento, factor de daño, entre otras propiedades del yacimiento.

Diferentes curvas tipo han sido desarrolladas para tomar en cuenta diferentes condiciones internas de frontera, modelos de pozo, tipos de yacimiento, y diferentes condiciones de frontera externa.

Pero antes, central al uso de las curvas tipo está el concepto de las variables adimensionales, estas permiten que soluciones generales a la ecuación de difusividad puedan ser definidas independientemente de propiedades específicas del yacimiento.

Por poner un ejemplo, si se toma la solución línea fuente:

$$p(r,t) = p_i + \frac{70.6qB\mu}{kh} E_i\left(-\frac{948\varphi\mu c_t r^2}{kt}\right) \tag{21}$$

Y se aplican las definiciones de las siguientes variables adimensionales, donde el sufijo D señala que se trata de una variable adimensional:

$$r_D = \frac{r}{r_w} \tag{22}$$

$$t_D = \frac{0.0002637k}{\varphi\mu c_t r_w^2} t \tag{23}$$

$$p_D = \frac{0.00708kh}{qB\mu}(p_i - p) \tag{24}$$

Se obtiene la siguiente ecuación:

$$p_D = -\frac{1}{2} E_i\left(-\frac{r_D^2}{4t_D}\right) \tag{25}$$

Esta ecuación es mucho más elegante y da luz a una de las curva tipo más simples que hay.

Sin embargo, la curva tipo más utilizada es la de Gringarten-Bourdet, que describe la respuesta de presión de un pozo vertical con un almacenamiento y daño constantes en un yacimiento infinito, produciendo una sola fase liquida a gasto constante, de una capa de espesor constante en un yacimiento homogéneo e isotrópico. Como se mencionó al inicio del capítulo, es este tipo de problema en el cual centraremos nuestros esfuerzos.

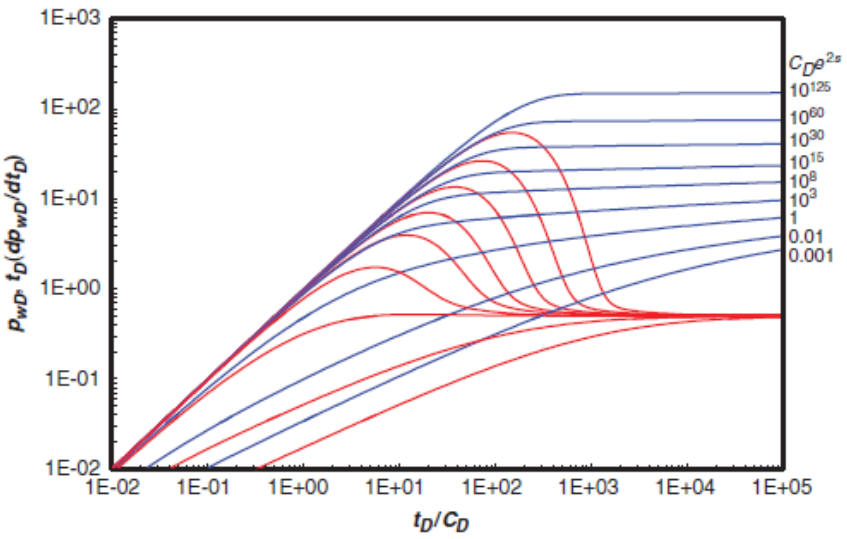

Ilustración 2 Curva tipo de Gringarten-Bourdet para un pozo vertical con un alamcenamiento y daño constante en un yacimiento infinito. (de Bourdet et al. 1983)

Para analizar una **prueba de decremento** a gasto constante con este método simplemente se tiene que:

1. Graficar el cambio de presión y la derivada logarítmica contra el tiempo.

$$t\frac{dp}{dt} = \frac{dp}{d\ln(t)} \tag{26}$$

2. Identificar el periodo dominado por el almacenamiento, indicado por la porción con una pendiente de 1, y el periodo IARF indicado por una porción horizontal.
3. Buscar el "match" entre esta gráfica y la curva tipo.
4. Seleccionar los puntos convenientes en los cuales leer tanto los valores del eje y como del eje x para ambos gráficos.
5. Calcular la permeabilidad, coeficiente adimensional de almacenamiento, coeficiente de almacenamiento y daño utilizando las ecuaciones (27) a (30). (Los sufijos MP se refiere a que esos datos se obtienen del punto de "match"):

$$k = \frac{qB\mu}{0.00708h}\left(\frac{p_D}{\Delta p}\right)_{MP} \tag{27}$$

$$C_D = \frac{0.0002637k}{\varphi\mu c_t r_w^2}\left(\frac{t}{t_D/C_D}\right)_{MP} \tag{28}$$

$$C = \frac{\varphi c_t r_w^2}{0.894}C_D \tag{29}$$

(30)

$$S = 0.5 \ln\left(\frac{C_D e^{2s}}{C_D}\right)$$

El mismo procedimiento puede ser aplicado a **pruebas de incremento** si el tiempo de la prueba es mucho más pequeño que el del periodo de flujo que le precede. O si el yacimiento actúa como yacimiento infinito durante la totalidad del periodo de flujo entonces al final de la prueba se puede utilizar el método con la sustitución del tiempo por el tiempo equivalente de Agarwal.

Ahora bien, para calcular la derivada logarítmica existen varios métodos, unos más complejos que otros. Entre más simple sea el método mayor será su incertidumbre y presentará mayores problemas a la hora de llevar a cabo el análisis.

Método de los dos puntos: Calculado el punto de la derivada con las siguientes dos ecuaciones para los ejes "y" y "x" respectivamente:

$$\left(t\frac{dp}{dt}\right) = \frac{p_R - p_L}{X_R - X_L} \tag{31}$$

$$t_c = \sqrt{t_L t_R} \tag{32}$$

Método de Bourdet:

1. Calcular el logaritmo natural del tiempo del respectivo punto $(X_c = ln(t))$.
2. Sustraer el parámetro de suavizado L (generalmente se usa $L = 0.1$) y encuentra el punto $X = ln(t)$ más grande que cumpla con $X < X_c - L$ y con este punto calcula una derivada de diferencias finitas hacia atrás:

$$\left(t\frac{dp}{dt}\right)_L = \frac{p_c - p_L}{X_c - X_L} \tag{33}$$

3. Añadir el parámetro de suavizado L y encuentra el punto $X = ln(t)$ más pequeño que cumpla con $X < X_c - L$ y con este punto calcula una derivada de diferencias finitas hacia delante:

$$\left(t\frac{dp}{dt}\right)_R = \frac{p_R - p_c}{X_R - X_c} \tag{34}$$

4. Calcula la derivada como un promedio ponderado de ambas derivadas:

$$\left(t\frac{dp}{dt}\right)_c = \frac{(X_R - X_c)\left(t\frac{dp}{dt}\right)_L + (X_c - X_L)\left(t\frac{dp}{dt}\right)_R}{X_R - X_L} \quad (35)$$

También se pueden usar diferencias finitas convencionales y algoritmos de spline.

2.5 REGÍMENES DE FLUJO Y GRÁFICA DE DIAGNÓSTICO

La identificación de los regímenes de flujo presentes durante una prueba de presión es uno de los pasos más importantes en la identificación de un modelo para el yacimiento. Cada régimen de flujo está caracterizado por un cierto comportamiento en tres diferentes gráficas: log-log, diagnóstico especializada y gráfica de línea recta especializada.

La gráfica diagnostico log-log convencional es muy parecida a una curva tipo en el sentido de que está construido por tener el cambio de la presión y la derivada logarítmica contra el tiempo graficados. En este tipo de gráfica podemos identificar regímenes de flujo mediante la observación de las pendientes de los distintos periodos como lo pueden ser el lineal con una pendiente de 0.5, bilinear y esférico con 0.25 y -0.5 de pendiente respectivamente.

2.6 FLUJO DE TRABAJO PARA LA INTERPRETACIÓN DE PRUEBAS DE PRESIÓN

En resumen, el flujo de trabajo general para una prueba de presión es:

1. Recolectar los datos necesarios para la interpretación.
2. Revisar y aplicar un control de calidad para preparar los datos para su interpretación.
3. Deconvolucionar los datos.
4. Identificar los regímenes de flujo presentes en los datos.
5. Seleccionar el modelo de yacimiento que se utilizará para la interpretación.
6. Estimar los parámetros que caracterizan al modelo utilizando métodos de línea recta o curvas tipo.
7. Simular o hacer un "*history match*" de la respuesta de presión.
8. Interpretar los parámetros estimados.
9. Validar resultados.

2.7 EJEMPLOS

Problema 1. Dadas las propiedades de la roca y del fluido en la tabla 2.1, analizar la prueba de decremento que se muestra en la tabla 2.2:

Tabla 2.1 Propiedades de la roca y el fluido para en el ejemplo de análisis de pruebas de decremento.

q	125 STB/D	p_i	2,750 psi
ϕ	22 %	B	1.152 bbl/STB
h	32 ft	μ	2.122 cp
r_w	0.25 ft	c_t	10.9 x10^{-6} psi^{-1}

Tabla 2.2 Datos de la prueba de decremento para el ejemplo de análisis.

t hr	p_{wf} psia	Δp psi	$t\frac{dp}{dt}$ psi	t hr	p_{wf} psia	Δp psi	$t\frac{dp}{dt}$ psi
0.001	2748.95	0.0869	2655.67	0.993	2196.92	10.545	1995.75
0.0021	2745.62	0.0988	2642.29	1.118	2170.7	11.865	1991.15
0.0034	2744.63	0.1121	2627.5	1.259	2148.33	13.349	1988.67
0.0048	2745.49	0.1271	2614.76	1.417	2126.44	15.018	1984.74
0.0064	2741.7	0.144	2598.79	1.595	2108.5	16.897	1979.34
0.0082	2742	0.163	2582.16	1.795	2090.87	19.01	1981.14
0.0102	2736.69	0.1844	2564.54	2.021	2080.73	21.387	1973.78
0.0125	2737.26	0.208	2545.27	2.275	2066.59	24.061	1970.58
0.0151	2733.72	0.236	2523.21	2.56	2054.29	27.07	1967.59
0.018	2729.13	0.266	2501.07	2.881	2048.25	30.455	1965.5
0.0212	2724.23	0.3	2475.93	3.242	2039.49	34.262	1961.64
0.0249	2720.57	0.339	2451.83	3.648	2035.32	38.546	1957.61
0.029	2715.83	0.382	2422.8	4.105	2029.91	43.366	1955.9
0.0336	2710.7	0.431	2397.61	4.619	2025.01	48.787	1951.21
0.0388	2706.63	0.486	2367.5	5.198	2018.87	54.787	1949.05
0.0447	2698.17	0.547	2338.18	5.848	2016.4	60.787	1945.7
0.0512	2692.75	0.617	2309.21	6.58	2011.11	66.787	1942.51
0.0587	2684.56	0.695	2277.84	7.404	2007.46	72	1941.14
0.067	2676.82	0.783	2251.46	8.331	2003.24		
0.0764	2665.33	0.882	2222.09	9.373	2000.53		

Solución.

Se grafica le presión de fondo fluyente p_{wf} contra el tiempo de prueba t en una escala semilogarítmica.

Se identifican los datos que se encuentren dentro del periodo de flujo radial. En este caso en particular los datos de tiempos tempranos están distorsionados por efectos de almacenamiento. Pero los datos en el tiempo tardío caen en una tendencia de línea recta, por lo que se seleccionan estos datos para continuar con el análisis.

Se dibuja una línea recta a través de dichos datos, y se busca tanto la pendiente de esta línea, así como la intersección con $t = 1$. Para encontrar la pendiente se lee dos puntos los más lejanos posible uno del otro para mejor precisión.

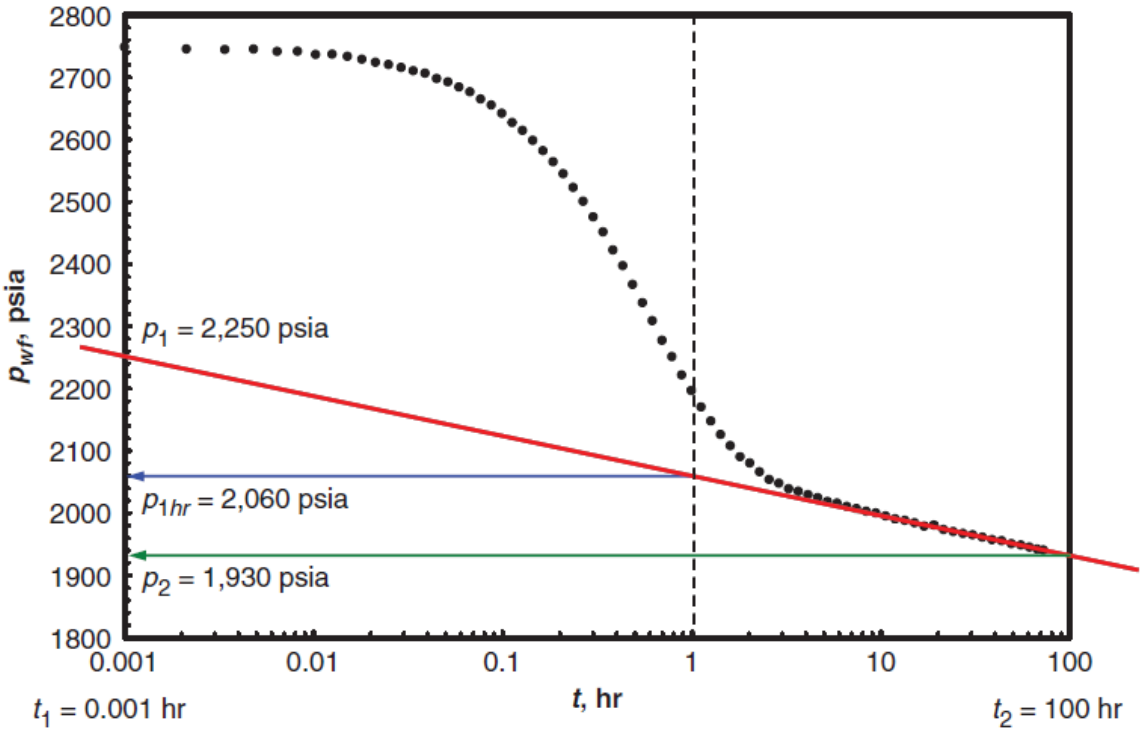

Ilustración 3 Análisis semilogarítmica de una prueba de decremento. (de Applied Well Test Interpretation, 2013)

$$m = \frac{2{,}250 - 1{,}930}{\log(0.001) - \log(100)} = \frac{320}{-3 - 2} = -64 \left[\frac{psi}{cycle}\right]$$

Calculamos la permeabilidad con ayuda de la ecuación 14:

$$k = \frac{162.6qB\mu}{|m|h} = \frac{162.6(125)(1.152)(2.122)}{|-64|(32)} = 24.3\ [md]$$

Cabe resaltar que la permeabilidad es convencionalmente calculada con el valor absoluto de la pendiente, para una prueba de decremento, la pendiente generalmente es negativa dependiendo del tipo de gráfica.

Ahora se calcula el factor de daño con ayuda del valor que leímos de presión en la intersección con $t = 1$ y la ecuación 15:

$$S = 1.151\left[\frac{p_{ihr} - p_{wf}}{|m|} - \log\left(\frac{k}{\varphi\mu c_t r_w^2}\right) + 3.23\right]$$

$$S = 1.151\left[\frac{2{,}750 - 2{,}060}{|-64|} - \log\left(\frac{24.26}{0.22(2.122)(10.9\,x10^{-6})(0.25)^2}\right) + 3.23\right] = 7.06$$

También se puede calcular el radio de investigación al final de la aparente línea recta en la gráfica semilogarítmica con ayuda de la ecuación 12. La línea recta en dicha gráfica termina alrededor de $t = 72$:

$$r_i = \sqrt{\frac{kt}{948\varphi\mu c_t}} = \sqrt{\frac{24.3(72)}{948(0.22)(2.122)(10.9x10^{-6})}} = 602\ [ft]$$

Problema 2. Dadas las propiedades de la roca y el fluido en la tabla 2.3, analizar los datos en la prueba de presión de decremento que se muestra en la tabla 2.4 usando la curva tipo de Gringarten-Bourdet:

Tabla 2.3 Propiedades de la roca y el fluido para en el ejemplo de análisis de pruebas de decremento.

q	125 STB/D	p_i	2,750 psi
ϕ	22 %	B	1.152 bbl/STB
h	32 ft	μ	2.122 cp
$\boldsymbol{r_w}$	0.25 ft	c_t	10.9 x10^{-6} psi^{-1}

Tabla 2.4 Datos de la prueba de decremento para el ejemplo de análisis.

t hr	$\boldsymbol{p_{wf}}$ psia	**Δp** psi	$\boldsymbol{t\frac{dp}{dt}}$ psi	**t** hr	$\boldsymbol{p_{wf}}$ psia	**Δp** psi	$\boldsymbol{t\frac{dp}{dt}}$ psi
0.001	2743.96	6.04	5.81	0.882	2341.74	408.26	21.07
0.0021	2737.52	12.48	12.23	0.993	2339.53	410.47	18.78
0.0034	2730.16	19.84	19.2	1.118	2337.28	412.72	20.48
0.0048	2722.32	27.68	26.01	1.259	2334.67	415.33	19.94
0.0064	2713.83	36.17	33.68	1.417	2332.55	417.45	17.67
0.0082	2704.46	45.54	42.11	1.595	2330.48	419.52	18.57
0.0103	2694.23	55.77	51.02	1.795	2328.15	421.85	19.37

0.0125	2683.03	66.97	59.53	2.021	2325.9	424.1	19
0.0151	2671.26	78.74	68.31	2.275	2323.66	426.34	18.95
0.018	2658.36	91.64	78.01	2.56	2321.42	428.58	18.24
0.0212	2644.71	105.29	86.98	2.881	2319.35	430.65	17.34
0.0249	2630.14	119.86	95.96	3.242	2317.33	432.67	18.29
0.029	2614.84	135.16	104.5	3.648	2315.03	434.97	18.25
0.0336	2598.72	151.28	112.2	4.105	2313.02	436.98	16.84
0.0388	2582.14	167.86	118.9	4.619	2311.06	438.94	17.24
0.0447	2564.96	185.04	125.1	5.198	2308.95	441.05	18.1
0.0513	2547.42	202.58	129	5.848	2306.79	443.21	17.31
0.0587	2529.89	220.11	131.7	6.58	2304.87	445.13	18.94
0.067	2512.19	237.81	133	7.404	2302.32	447.68	18.17
0.0764	2494.89	255.11	131	8.331	2300.59	449.41	16.37
0.0869	2478.16	271.84	127.5	9.373	2298.46	451.54	17.4
0.0988	2462.14	287.86	122.7	10.55	2296.49	453.51	17.53
0.1121	2446.97	303.03	115.8	11.86	2294.32	455.68	17.83
0.1271	2432.97	317.03	107.1	13.35	2292.28	457.72	17.63
0.144	2420.21	329.79	96.45	15.02	2290.17	459.83	17.7
0.163	2409.04	340.96	86.94	16.9	2288.11	461.89	17.42
0.1844	2398.75	351.25	76.8	19.01	2286.06	463.94	17.51
0.209	2390.2	359.8	65.3	21.39	2283.99	466.01	17.93
0.236	2382.79	367.21	56.91	24.06	2281.84	468.16	17.93
0.266	2376.35	373.65	48.83	27.07	2279.76	470.24	16.5
0.3	2370.96	379.04	40.93	30.45	2277.95	472.05	17.8
0.339	2366.46	383.54	35.72	34.26	2275.56	474.44	17.7
0.382	2362.35	387.65	32.13	38.55	2273.78	476.22	17.07
0.431	2358.74	391.26	29.35	43.37	2271.54	478.46	17.02
0.486	2355.31	394.69	26.09	48.79	2269.77	480.23	16.49
0.547	2352.5	397.5	23.34	54.79	2267.69	482.31	18.41
0.617	2349.73	400.27	23.13	60.79	2265.73	484.27	17.83
0.695	2346.98	403.02	21.79	66.79	2264.46	485.54	18.66
0.783	2344.54	405.46	22.01	72	2262.99	487.01	16.28

Solución.

Se grafica el cambio de la presión, definido como $\Delta p = p_i - p_{wf}(t)$ junto con su derivada logarítmica definida en la ecuación 26 contra el tiempo de la prueba t teniendo los mismos ciclos logarítmicos que la curva tipo, a esta gráfica se le referirá como gráfica de datos de campo:

En esta misma gráfica es posible identificar el periodo dominado por el efecto del almacenamiento que es característico por una pendiendo de una unidad en la derivada durante tiempos tempranos. Así como

también es factible identificar el periodo de flujo radial (*IARF)* indicado por la porción horizontal de la derivada en tiempos tardíos.

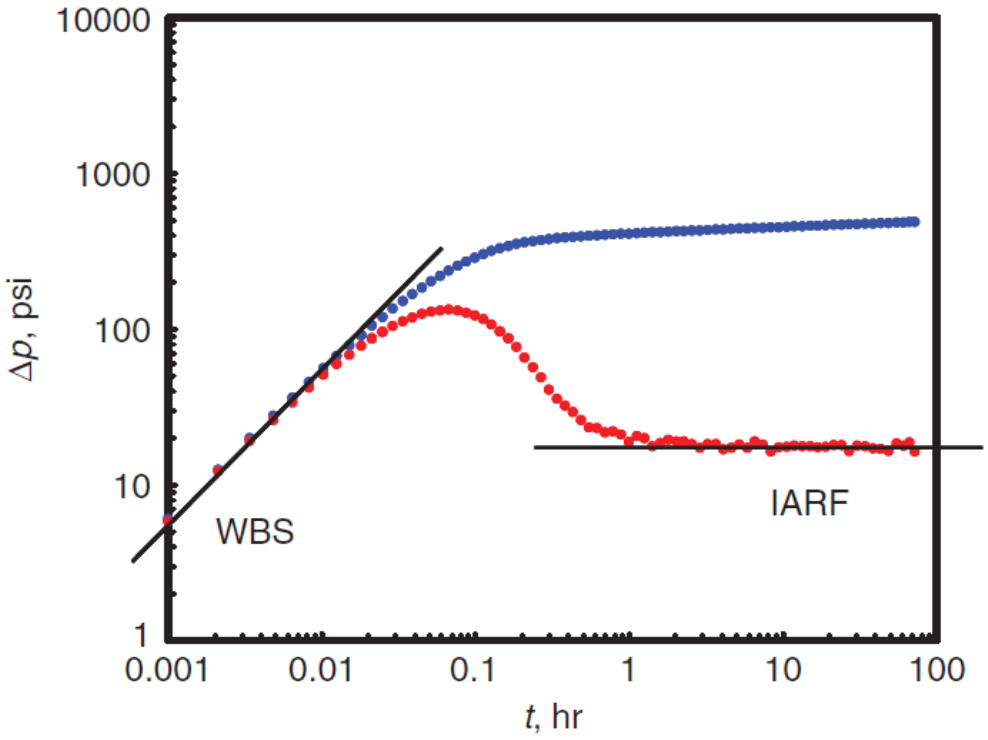

Ilustración 4 Gráfica de datos de campo. (de Applied Well Test Interpretation, 2013)

Se coloca la gráfica de datos de campo sobre la gráfica de la curva tipo:

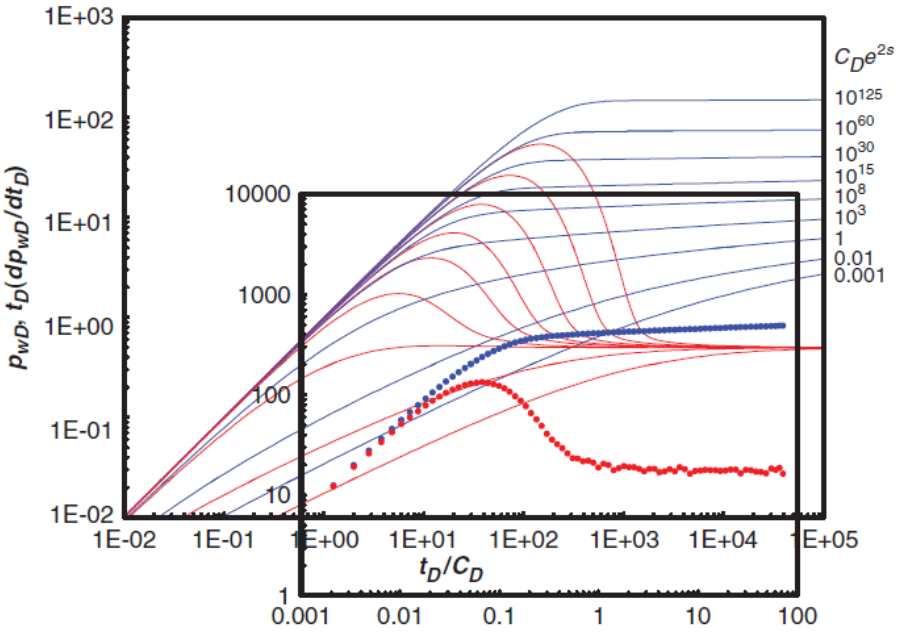

Ilustración 5 Gráfica de datos de campo sobre gráfica de curva tipo. (de Applied Well Test Interpretation, 2013)

Manteniendo los ejes de ambas gráficas paralelos, se mueve la gráfica de los datos de campo verticalmente hasta que la parte horizontal del periodo del flujo radial se alinea con la parte horizontal de la curva tipo.

Después se desplaza la gráfica de datos de campo horizontalmente hasta que la pendiente de una unidad que indica los efectos de almacenamiento se alinee con la gráfica de la curva tipo.

En la gráfica de la curva tipo, se busca la curva de presión y de derivada de presión que mejor concuerde con la de la gráfica de datos de campo. En ciertas ocasiones quizás sea necesario realizar una interpolación para conseguir una buena representación en la curva tipo. Se obtiene el valor de C_De^{2s} correspondiente a la curva tipo seleccionada.

Se selecciona un punto conveniente del eje x, y se obtiene t_D/C_D de la gráfica de la curva tipo y t de la curva de los datos de campo.

De manera similar, se selecciona un punto conveniente en el eje y, y se obtiene p_D de la gráfica de la curva tipo y Δp de la curva de los datos de campo.

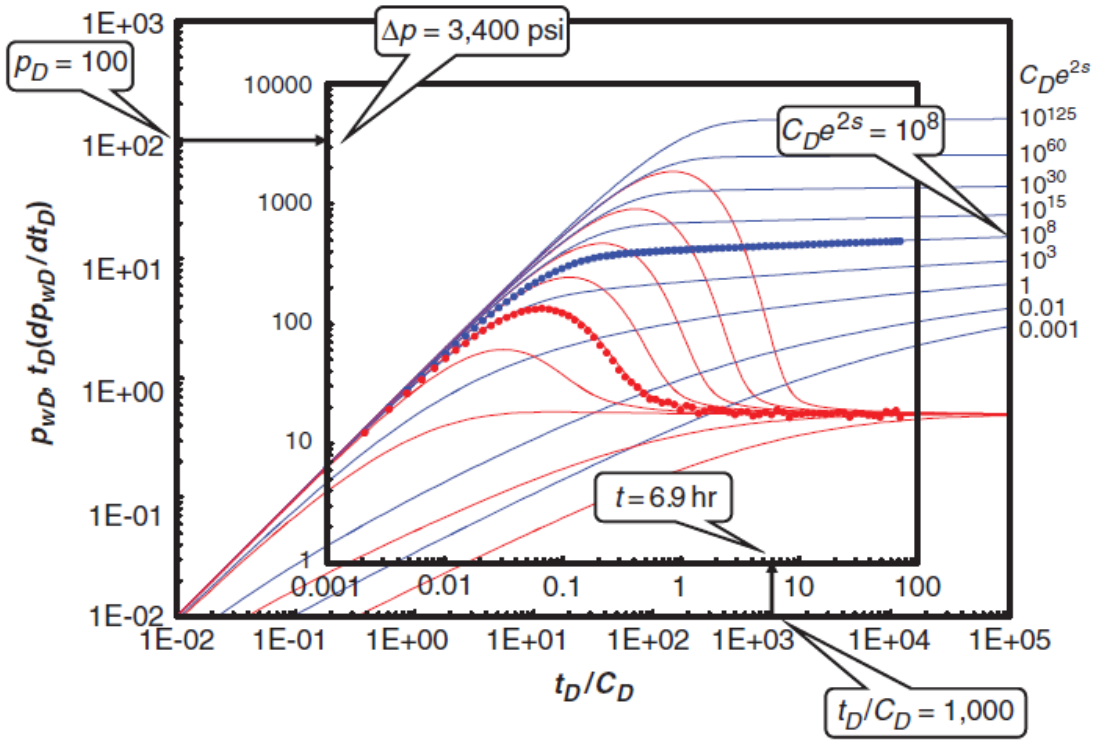

Ilustración 6 Obtención de datos que se alinean entre ambas gráficas. (de Applied Well Test Interpretation, 2013)

Se calcula la permeabilidad con los datos de presión que se obtienen con la ayuda de la ecuación 27:

$$k = \frac{qB\mu}{0.00708h}\left(\frac{p_D}{\Delta p}\right)_{MP} = \frac{125\ (1.152)(2.122)}{0.00708(50)}\left(\frac{100}{3{,}400}\right) = 25.4\ [md]$$

Y se calcula el coeficiente de almacenamiento adimensional con los datos de tiempo y la ayuda de la ecuación 28:

$$C_D = \frac{0.0002637k}{\varphi\mu c_t r_w^2}\left(\frac{t}{t_D/C_D}\right)_{MP} = \frac{0.0002637(25.4)}{0.22(2.122)(1.094\ x10^{-5})(0.25)^2}\left(\frac{6.9}{1{,}000}\right) = 145$$

Para así calcular el coeficiente de almacenamiento con la ayuda de la ecuación 29:

$$C = \frac{\varphi c_t r_w^2}{0.894} C_D = \frac{0.22(1.094\,x10^{-5})(50)(0.25)^2}{0.894}(145) = 0.00122\ \left[\frac{bbl}{psi}\right]$$

Y finalmente se calcula el factor de daño con el valor de $C_D e^{2s}$ y la ecuación 30:

$$S = 0.5 \ln\left(\frac{C_D e^{2s}}{C_D}\right) = 0.5 \ln\left(\frac{10^8}{145}\right) = 6.7$$

Problema 3. Dadas las propiedades de la roca y el fluido en la tabla 2.5, analizar la prueba de incremento de la tabla 2.6 usando la curva tipo de Gringarten-Bourdet. Hacer el análisis primero usando el tiempo de cierre como como la variable del tiempo, y después usando el tiempo equivalente de Agarwal.

Tabla 2.5 Propiedades de la roca y el fluido para en el ejemplo de análisis de pruebas de incremento.

q	100 STB/D	t_p	2160 hour
ϕ	20 %	B	1.17 bbl/STB
h	25 ft	μ	2.24 cp
$\boldsymbol{r_w}$	0.25 ft	c_t	10.92x10^{-6} psi^{-1}
A	40 acre		

Tabla 2.6 Datos de la prueba de incremento para el ejemplo de análisis.

Δt hr	$\frac{t_p+\Delta t}{\Delta t}$ Δt	$\boldsymbol{p_{wf}, p_{ws}}$ psia	Δt **hr**	$\frac{t_p+\Delta t}{\Delta t}$ Δt	$\boldsymbol{p_{wf}, p_{ws}}$ psia	Δt **hr**	$\frac{t_p+\Delta t}{\Delta t}$ Δt	$\boldsymbol{p_{wf}, p_{ws}}$ psia
0		2605.93	0.1844	11715	2849.77	4.619	469	2970.5
0.001	2160001	2610.68	0.2085	10361	2857.15	5.198	417	2973.92
0.0021	1028572	2615.82	0.2355	9173	2864.01	5.848	370	2977.3
0.0034	635295	2621.38	0.266	8121	2870.4	6.581	329	2980.65
0.0048	450001	2627.4	0.3002	7196	2876.36	7.404	293	2983.97
0.0064	337501	2633.91	0.3387	6378	2881.96	8.331	260	2987.24
0.0082	263416	2640.91	0.3821	5654	2887.24	9.373	231	2990.48
0.0102	211766	2648.43	0.4308	5015	2892.26	10.55	206	2993.66
0.0125	172801	2656.46	0.4857	4448	2897.05	11.87	183	2996.79
0.0151	143047	2665.02	0.5474	3947	2901.66	13.35	163	2999.87
0.018	120001	2674.09	0.6168	3503	2906.11	15.02	145	3002.89
0.0212	101888	2683.65	0.6949	3109	2910.42	16.9	129	3005.84
0.0249	86748	2693.68	0.7828	2760	2914.63	19.01	115	3008.71

0.029	74484	2704.12	0.8816	2451	2918.73	21.39	102	3011.5
0.0336	64287	2714.93	0.9928	2177	2922.76	24.06	90.8	3014.19
0.0388	55671	2726.04	1.118	1933	2926.71	27.07	80.8	3016.78
0.0447	48323	2737.36	1.259	1717	2930.6	30.46	71.9	3019.25
0.0512	42189	2748.81	1.417	1525	2934.43	34.26	64	3021.59
0.0587	36798	2760.29	1.595	1355	2938.21	38.55	57	3023.78
0.067	32240	2771.7	1.796	1204	2941.94	43.37	50.8	3025.8
0.0764	28273	2782.94	2.021	1070	2945.64	48.79	45.3	3027.65
0.0869	24857	2793.89	2.275	951	2949.29	54.79	40.4	3029.27
0.0988	21863	2804.48	2.56	845	2952.91	60.79	36.5	3030.55
0.1121	19270	2814.62	2.881	751	2956.49	66.79	33.3	3031.56
0.1271	16995	2824.25	3.242	667	2960.04	72	31	3032.27
0.144	15001	2833.33	3.648	593	2963.56			
0.163	13253	2841.84	4.105	527	2967.04			

Solución.

Graficamos el cambio de presión, definido como $\Delta p = p_{ws}(\Delta t) - p_{wf}$ y su derivada logarítmica contra el tiempo de cierre Δt, como se muestra en la figura:

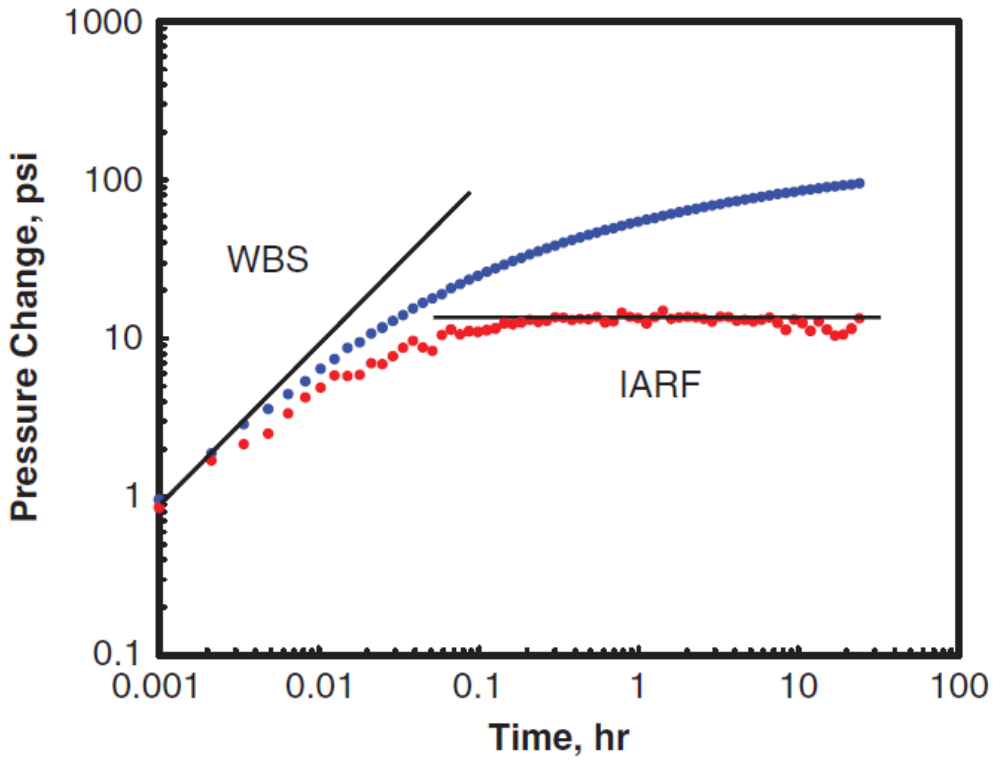

Ilustración 7 Gráfica de datos de campo usando el tiempo de cierre. (de Applied Well Test Interpretation, 2013)

Se identifica el periodo dominado por efectos de almacenamiento indicado por la porción que se encuentra dentro de la recta de pendiente 1, y el periodo de flujo radial (*IARF)* indicado por la parte horizontal de la derivada. Para este ejemplo se intuye que el pozo esta estimulado ya que casi no podemos apreciar efectos de almacenamiento.

Cabe resaltar que los datos de la derivada se desvían de la horizontal al final de la prueba. Es conveniente ignorar estos datos ya que se puede concluir que el tiempo de cierre tiene que ser mucho menor que el tiempo de producción en este método.

Se coloca la gráfica de datos de campo sobre la curva tipo. Y de manera similar a como hicimos durante el análisis de la prueba de decremento, se mantienen los ejes paralelos entre ambas gráficas. Se mueve la gráfica de datos de campo de manera vertical hasta que la parte horizontal del flujo radial se alinea con la horizontal de la curva tipo.

Manteniendo los ejes de ambas gráficas paralelos, ahora se desplaza la gráfica de datos de campo horizontalmente hasta que la pendiente de 1 se alinee con la curva tipo.

Sobre la gráfica de la curva tipo, encontrar la curva de presión y la derivada de la presión que se alinee de mejor manera con los datos observados en campo y obtenemos el dato de $C_D e^{2s}$.

Se selecciona un punto conveniente en el eje x para leer t_D/C_D en la curva tipo y Δt en la curva de datos de campo. Lo mismo para el eje y se lee p_D en la curva tipo y Δp en la gráfica de datos de campo:

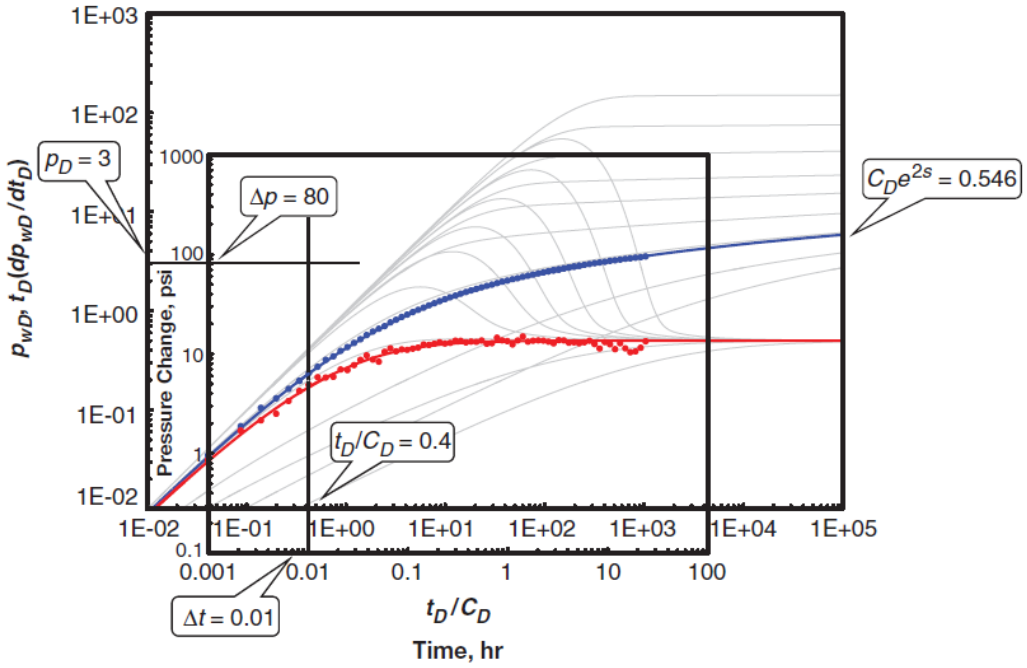

Ilustración 8 Análisis de prueba de incremento. (de Applied Well Test Interpretation, 2013)

Ahora se calcula la permeabilidad con los datos de presión y la ecuación 27:

$$k = \frac{qB\mu}{0.00708h}\left(\frac{p_D}{\Delta p}\right)_{MP} = \frac{50(1.078)(8.36)}{0.00708(65)}\left(\frac{3}{80}\right) = 36.7\ [md]$$

Después se calcula el coeficiente de almacenamiento adimensional con la ecuación 28 y los datos de tiempo:

$$C_D = \frac{0.0002637k}{\varphi\mu c_t r_w^2}\left(\frac{t}{t_D/C_D}\right)_{MP} = \frac{0.0002637(36.7)}{0.15(8.36)(7.64x10^{-6})(0.33)^2}\left(\frac{0.01}{0.4}\right) = 93$$

Se calcula el coeficiente de almacenamiento con ayuda de la ecuación 29:

$$C = \frac{\varphi c_t r_w^2}{0.894}C_D = \frac{0.15(7.64x10^{-6})(65)(0.33)^2}{0.894}(93) = 0.00084\ \left[\frac{bbl}{psi}\right]$$

Finalmente se obtiene el factor de daño con el dato de $C_D e^{2s}$ y con ayuda de la ecuación 30:

$$S = 0.5\ln\left(\frac{C_D e^{2s}}{C_D}\right) = 0.5\ln\left(\frac{0.546}{97}\right) = -2.6$$

Ahora usando el tiempo equivalente de Awargal es posible graficar el cambio de la presión y su derivada logarítmica contra el tiempo equivalente de Awargal definido en la ecuación 18:

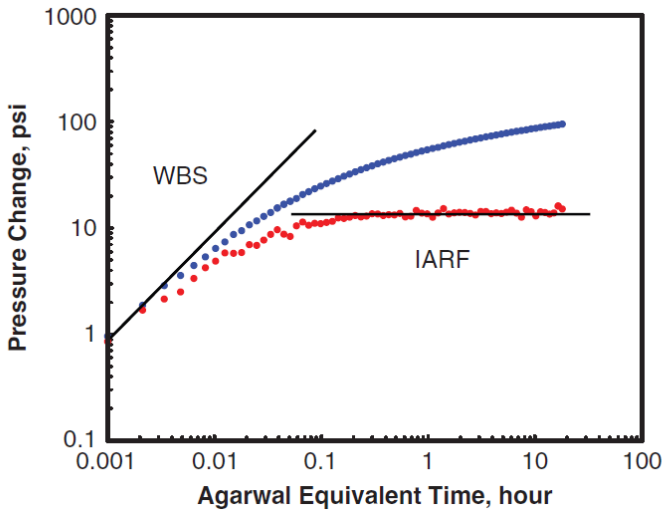

Ilustración 9 Gráfica de datos de campo usando el tiempo equivalente de Agarwal. (de Applied Well Test Interpretation, 2013)

Posteriormente se tiene que identificar el periodo dominado por efectos de almacenamiento, así como el periodo de flujo radial (*IARF)*. A continuación, los siguientes pasos son obtener los datos para hacer uso de las ecuaciones 28, 29, 30 son los mismos.

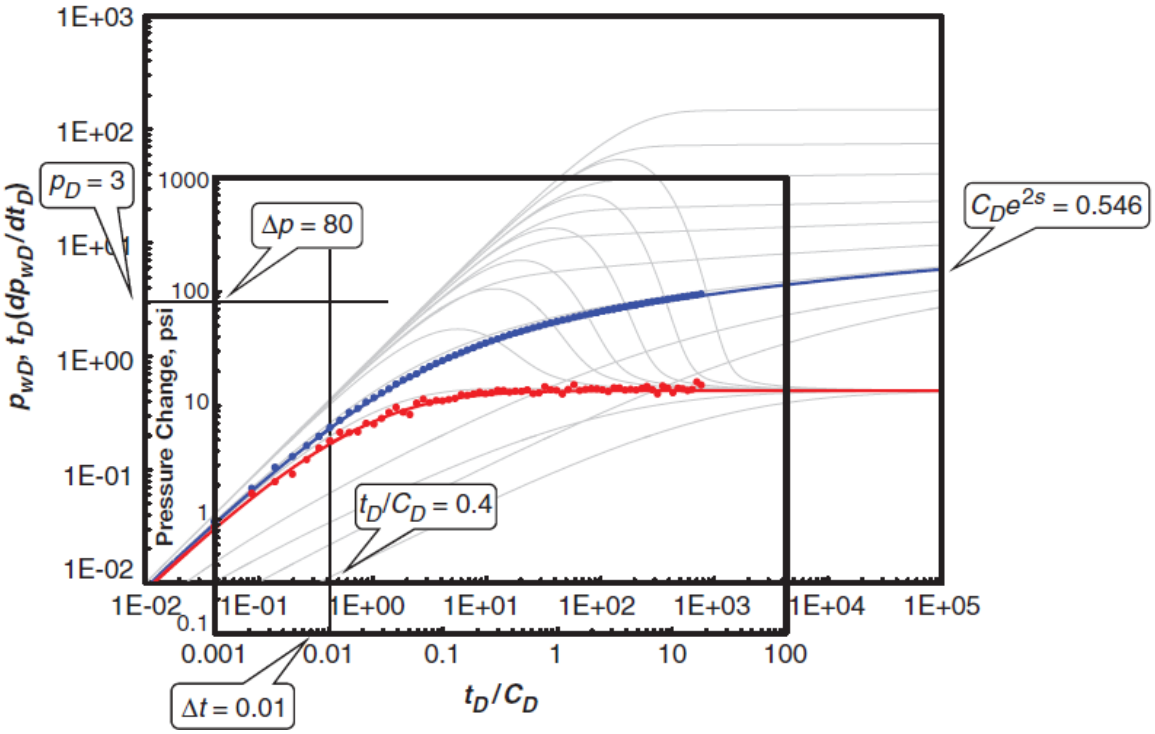

Ilustración 10 Superposición de gráfica de datos de campo sobre gráfica de curvas tipo. (de Applied Well Test Interpretation, 2013)

Nótese que a diferencia de cuando se utiliza el tiempo de cierre, los datos de la derivada no se desvían de la horizontal en el final de la prueba, permitiendo hacer uso de todos los datos disponibles.

Ahora calculando la permeabilidad con los datos de presión y la ecuación 27:

$$k = \frac{qB\mu}{0.00708h}\left(\frac{p_D}{\Delta p}\right)_{MP} = \frac{50(1.078)(8.36)}{0.00708(65)}\left(\frac{3}{80}\right) = 36.7\ [md]$$

Después se calcula el coeficiente de almacenamiento adimensional con la ecuación 28 y los datos de tiempo:

$$C_D = \frac{0.0002637k}{\varphi \mu c_t r_w^2}\left(\frac{t}{t_D/C_D}\right)_{MP} = \frac{0.0002637(36.7)}{0.15(8.36)(7.64x10^{-6})(0.33)^2}\left(\frac{0.01}{0.4}\right) = 93$$

Se obtiene el coeficiente de almacenamiento con ayuda de la ecuación 29:

$$C = \frac{\varphi c_t r_w^2}{0.894} C_D = \frac{0.15(7.64x10^{-6})(65)(0.33)^2}{0.894}(93) = 0.00084\ \left[\frac{bbl}{psi}\right]$$

Finalmente se calcula el factor de daño con el dato de $C_D e^{2s}$ y con ayuda de la ecuación 30:

$$S = 0.5 \ln\left(\frac{C_D e^{2s}}{C_D}\right) = 0.5 \ln\left(\frac{0.546}{97}\right) = -2.6$$

Capítulo 3. Machine Learning y su Aplicación en la Industria Petrolera

3.1 INTRODUCCIÓN

Una serie de técnicas que permiten a los algoritmos trabajar de mejor manera, basándose en su experiencia es una de las maneras más simples en la que se puede definir al *machine learning*.

El machine learning puede realizar análisis predictivos de manera mucho más veloz que cualquier ser humano sería capaz, como resultado de esto, el machine learning puede ayudar a los profesionistas a trabajar de manera mucho más eficiente.

Actualmente el machine learning es esencialmente diferente de la estadística. Sí, el machine learning tiene bases estadísticas, pero hace diferentes suposiciones a la estadística tradicional porque sus objetivos son distintos.

3.2 BIG DATA

Ahora bien, ¿qué se necesita para la construcción de un algoritmo de machine learning? La respuesta corta es que se necesitan datos, grandes cantidades de datos. Así como el ser humano aprende mejor cuando existe un mayor aporte de conocimiento, así lo hacen estos algoritmos.

Pero no solo puedes alimentar al algoritmo con una serie de tablas y esperar que trabaje de manera correcta. Se debe de contar con una manera de organizar y analizar todos estos datos, con la finalidad de que sean de fácil acceso y que el algoritmo sea capaz de ver patrones dentro de ellos con mayor facilidad. Una vez que se tengan los datos ordenados de una manera útil y entendible, se puede empezar a alimentar al algoritmo para que este los manipule de una manera que produzca un resultado.

Los algoritmos de machine learning son solamente útiles cuando están entrenados, porque el entrenamiento le permite a la computadora usar análisis previos para trabajar con nuevos datos que nunca había visto anteriormente.

3.3 TIPOS DE APRENDIZAJE

Es posible dividir el machine learning en tres grupos principales basándose en su propósito:

Aprendizaje supervisado, que ocurre cuando un algoritmo aprende de datos ejemplo en conjunto con resultados asociados a estos mismos datos, con el objetivo de después poder predecir la respuesta correcta cuando se le presenta nueva información.

El **Aprendizaje no supervisado** se refiere a cuando un algoritmo aprende de ejemplos sin una respuesta asociada, dejando que el algoritmo determine los patrones en los datos por sí mismo. Este tipo de aprendizaje provee al usuario con nuevo conocimiento sobre el significado de los datos, y da como resultado nuevos datos de entrada para algoritmos de aprendizaje supervisado.

Y por último el **aprendizaje de reforzamiento**, que sucede cuando le presentas secuencialmente al algoritmo ejemplos que tienen etiquetas faltantes, como los resultados en aprendizaje no supervisado, sin embargo, se tiene que ofrecer una retroalimentación positiva o negativa para cada ejemplo acorde a la solución que el algoritmo propone, como si fuera prueba y error.

3.4 TERMINOLOGÍA

Un algoritmo de machine learning puede proveer una respuesta o predicción cuando está apoyado de la información requerida o datos de prueba, y una respuesta asociada que generalmente son ejemplos de predicciones que quieres que el algoritmo sea capaz de estimar.

Toda aquella información utilizada como un dato de entrada se conoce como *feature* o característica en español. Existen dos tipos distintos de característicos, cuantitativos y cualitativos. Los primeros son perfectos para el machine learning porque definen valores como números, a diferencia de los cualitativos que se refieren a etiquetas.

En el caso de los característicos cualitativos tienen que pasar por un preprocesamiento para que la máquina pueda hacer uso de esa información. Una de las técnicas más populares es el *one-hot encoding*, que consiste en asignar valores binarios a cada característico, por ejemplo, si un característico que voy a usar es el clima, puedo asignarle a “soleado” el valor 1, y a “lluvioso” y “nublado” el valor 0.

El algoritmo también puede sufrir problemas de memorización, en donde puede dar la ilusión de que todo está trabajando de manera correcta ya que parece haber encajado los datos de la muestra de entrenamiento muy bien, pero realmente muchos problemas se vuelven rápidamente evidentes cuando empiezas a usar dicho algoritmo con datos que están fueras de la muestra del entrenamiento, esto se llama *overfitting*, y ocurre cuando tu algoritmo ha aprendido tanto de tus datos que llego al punto de mapear formas de curvas y reglas que no existen realmente.

Si lo que se busca es visualizar el grado al cual el algoritmo de machine learning está sufriendo de problemas de parcialidad, se puede hacer uso de un gráfico conocido como *curva de aprendizaje*, donde gráficas el rendimiento de uno o varios algoritmos con respecto a las cantidades de datos utilizados para el entrenamiento.

Otra parte importante de los algoritmos de machine learning son los híper-parámetros, que son parámetros que tu configuras antes de que el proceso de aprendizaje siquiera empiece, y además estos no pueden ser aprendidos a partir de los datos.

3.5 ENTRENAMIENTO

En programación básica, el programador crea una función que acepta datos de entrada y que da como resultado un dato de salida, por ejemplo, crear la función de adición que acepta dos valores, como 1 y 2; el resultado invariablemente será 3.

En machine learning este proceso se revierte, ahora se saben los valores de entrada, como 1 y 2, y que además el resultado esperado debe de ser 3. Sin embargo, no se conoce qué función aplicar para crear el resultado deseado. El entrenamiento provee al algoritmo toda clase de ejemplos sobre los datos de entrada y resultados deseados, y es el mismo algoritmo que crea una función flexible conforme a los datos que se le dio.

El secreto del machine learning es generalizar esta función de manera que funcione con datos fuera del conjunto de entrenamiento. Para conseguir esta generalización partiendo de la gran cantidad de datos que se tienen el algoritmo se apoya de tres componentes muy importantes:

- **Representación:** El algoritmo debe de ser capaz de crear un modelo o función que produzca los resultados deseados a partir de los datos de entrada, parte de esto es descubrir qué elementos o características del conjunto de datos puede usar para aprender.
- **Evaluación:** El algoritmo por sí solo no puede distinguir la diferencia entre un modelo bueno y uno malo, requiere de una función de evaluación que determine cuál modelo es el que mejor funciona.
- **Optimización:** En algún punto, el proceso de entrenamiento produce un conjunto de modelos que obtienen los resultados deseados a partir del conjunto de datos de entrada, entonces tiene que determinar dentro de estos modelos cual es mejor, para así darlo como resultado del proceso de entrenamiento.

En un mundo perfecto después del entrenamiento se debería realizar una prueba del funcionamiento del algoritmo sobre datos de los cuales nunca antes ha aprendido, sin embargo, esperar por datos frescos no siempre es posible en términos de costos y tiempo. Esto da como resultado la común práctica de dividir los datos disponibles en un conjunto de entrenamiento y uno de prueba.

Además, cuando se requiere modificar y ajustar algunos parámetros del aprendizaje del algoritmo no es conveniente usar el conjunto de prueba, esto podría dar camino a un problema conocido como *snooping* (una forma de sesgo estadístico que manipula datos o análisis para obtener artificialmente resultados estadísticamente significativos.), por lo que se requiere de otro tercer conjunto que se denomina de validación.

Si se presenta el caso en el que la muestra de datos es pequeña, se puede introducir cierta parcialidad al algoritmo debido a la reducción del conjunto de entrenamiento, es decir, se puede excluir ejemplos realmente útiles para el aprendizaje fuera. Para este problema existe una técnica conocida como *cross-validation,* ya que se basa en un corte aleatorio de la muestra en un número k de *folds* (conjuntos variables de datos de entrenamiento y de prueba) del mismo tamaño. Después cada *fold* es utilizado como un conjunto de prueba y los restantes como conjunto de entrenamiento de manera turnada. Al final de cada uno de los entrenamientos se puede obtener un promedio de los resultados que indica una estimación probabilística del rendimiento predictivo.

3.6 FUNCIÓN DE COSTO

Detrás de la optimización de los algoritmos de machine learning están las respuestas de una función interna al algoritmo, conocida como función de costo, función de pérdida, o función objetivo. Y básicamente se refiere a una función de evaluación que mide que tan bien el algoritmo ha mapeado la función que describe el problema que trata de resolver.

La función de costo compara las predicciones del algoritmo con el resultado real, esto determina un nivel de error de manera matemática, y es este nivel de error el cual tiene que ser minimizado.

Es la misma función de costo la que transmite lo que es realmente importante y significativo de los resultados del algoritmo, es decir, se tiene que decidir y definir la función de costo en base a al entendimiento del problema que se quiere resolver y al nivel que se busca resolver. Por ejemplo, la función de costo para predecir el mercado de valores será distinta del algoritmo utilizado para dictar si alguna persona padece una enfermedad y, además, en este último ejemplo, es preferible estimar que ciertas personas tienen la enfermedad y que al final no la tengan, que dejar fuera pacientes que sí tenían la enfermedad.

3.7 EVALUACIÓN DE RENDIMIENTO

Además de la precisión de las predicciones del algoritmo existen varias métricas adicionales que se pueden utilizar para ganar conocimiento sobre el problema. Uno de los más comúnmente utilizados son las matrices de confusión.

Una matriz de confusión resume las instancias de prueba en una tabla que compara los valores predecidos con los verdaderos valores. De esta matriz es posible obtener valores como la precisión, que se refiere a la proporción de todos los casos verdaderos o correctos con respecto a la totalidad de los resultados. La precisión mide el porcentaje de predicciones positivas correctas con respecto a la totalidad de las predicciones positivas y, *recall* se refiere al porcentaje de los positivos correctos que fueron identificados con respecto a los positivos correctos y los falsos negativos. Finalmente, la puntuación $f1$ es un promedio armónico entre la precisión y el *recall*.

Para esta investigación se decidió hacer uso de dos métricas para medir el poder de predicción de cada modelo, el R^2 y el error cuadrático medio.

El primero es una medida estadística de qué tan cerca están los datos de la línea de regresión ajustada. Esto permite saber el porcentaje con el que varía el modelo con respecto al dato real, o en otras palabras, ayuda a visualizar si el modelo es capaz de seguir la tendencia de los datos, y entre mayor sea su valor, mejor es el modelo. El error medio cuadrático mide el promedio de los errores al cuadrado, es decir, la diferencia entre la predicción y el dato real, y permite concluir que tan alejados o cercanos estamos de realizar una predicción correcta, entre menor sea su valor, mejor es el modelo.

Hablar de grid serach

Ilustración 11 Matriz de Confusión.

3.8 HIPERPARÁMETROS

Los hiperparámetros son parámetros ajustables que permiten controlar el proceso de entrenamiento de un modelo. Por ejemplo, con redes neuronales, puede decidir el número de capas ocultas y el número de nodos de cada capa. El rendimiento de un modelo depende en gran medida de los hiperparámetros.

El ajuste de hiperparámetros, también denominado optimización de hiperparámetros es el proceso de encontrar la configuración de hiperparámetros que produzca el mejor rendimiento. Normalmente, el proceso es manual y costoso desde el punto de vista computacional.

3.9 REDES NEURONALES

Las redes neuronales son un tipo de algoritmo dentro del machine learning. Y el núcleo de este algoritmo está en la neurona, a veces también llamada unidad, y es el descendiente de un concepto mucho más simple conocido como **perceptrón**.

Un perceptrón sigue los siguientes pasos:

1. Toma valores de un entorno, como lo puede ser un conjunto de datos.
2. Pesa estos valores basándose en cuáles deberían afectar más la respuesta final.
3. Suma todos los valores que fueron pesados.
4. Se activa cuando la suma excede un valor determinado.

Desafortunadamente un perceptrón no puede aprender cuando las clases que trata de procesar no son linealmente separables. Aun así, los investigadores se percataron después de que, aunque un solo perceptrón no puede aprender la operación lógica XOR (una salida verdadera resulta si una, y solo una de las entradas a la puerta es verdadera), dos perceptrones trabajando juntos sí pueden.

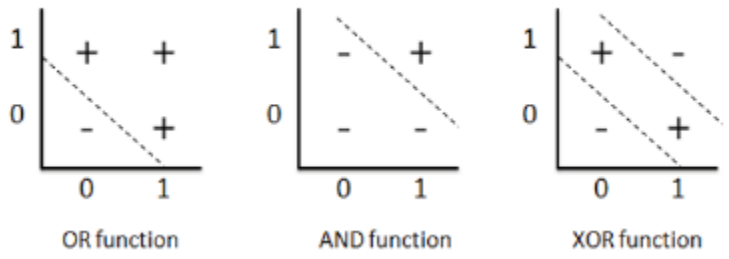

Ilustración 12 Ejemplificación de funciones de clasificación (John Mueller, 2021)

Entonces es fácil ver cómo las neuronas en una red neuronal son la evolución de los perceptrones; toman muchos valores con diferentes pesos como datos de entrada, los suman, y proveen como resultado esta misma suma, justo como lo haría un perceptrón. Sin embargo, en las neuronas existe una transformación más sofisticada de la suma, la función de activación, de la cuál hablaremos más adelante.

Una simple red neuronal se conforma de tres capas, una capa se refiere a un conjunto de nodos que simulan a las neuronas en un cerebro biológico. Estas capas son:

- **Capa de entrada:** representa a las características de entrada, y cada nodo es una característica predictiva.
- **Capa de salida:** representa las variables objetivo. En una clasificación binaria o en un problema de regresión la capa de salida solo contiene un nodo, mientras que en una clasificación multiclase cuenta con un número determinado de nodos.
- **Capa oculta:** puede considerarse compuesta de la información extraída de la capa anterior, y puede haber varias de ellas.

Estas capas están conectadas conceptualmente, transmiten señales de una neurona en una capa a otra que se encuentra en otra capa. Estas conexiones esta parametrizadas mediante pesos.

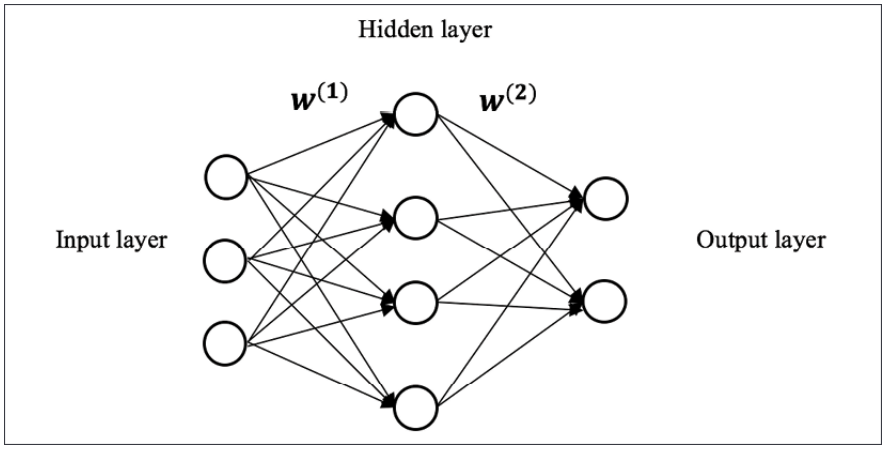

Ilustración 13 Ejemplo de una red neuronal simple. (Yuxi Liu, 2019)

Ahora bien, si se tiene una entrada x de dimensiones n y que la capa oculta del modelo está compuesta de H nodos. La matriz de pesos $W^{(1)}$ que conecta la entrada con la capa oculta es del tamaño $n \times H$, donde cada columna representa el coeficiente que asocia la entrada con el nodo oculto número h. La salida (también llamada "activación") de la capa oculta puede ser expresada matemáticamente como:

$$a^{(2)} = f\left(z^{(2)}\right) = f(W^{(1)}x) \tag{36}$$

Aquí, la función $f(z)$ es la función de activación. Como su nombre lo indica, la función de activación se cerciora de que tan activada esta cada neurona, simulando el cómo los cerebros funcionan. Una función de activación puede dar como resultado un valor cero hasta que la entrada alcance una cierta barrera, o puede decrecer o incrementar un valor mediante una re-escalación no linear.

Las funciones de activación más comunes incluyen la función sigmoidal, la tangencial hiperbólica y la función ReLU (rectified linear unit):

$$Sigmoidal\ (z) = \frac{1}{1 + e^{-z}} \tag{37}$$

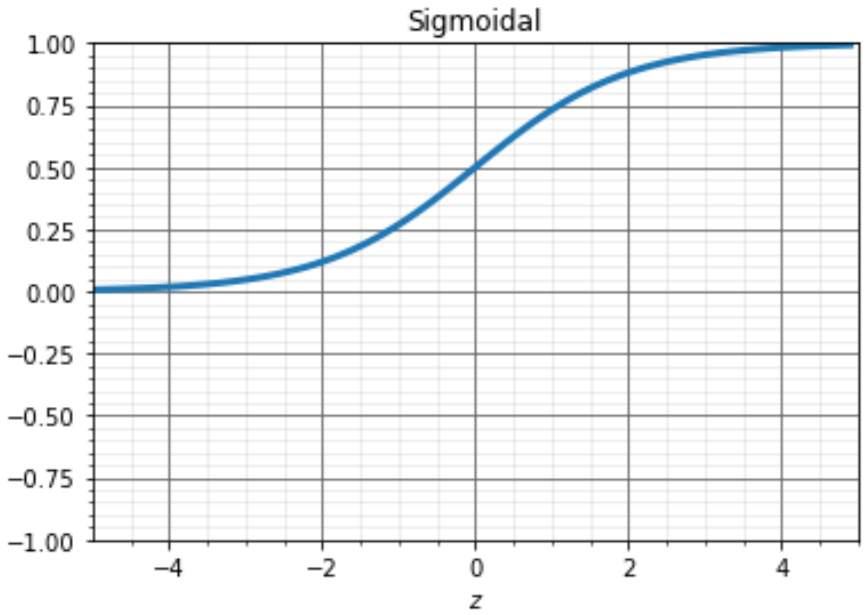

Ilustración 14 Representación gráfica de la función sigmoidal.

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} = \frac{2}{1 + e^{-2z}} - 1 \tag{38}$$

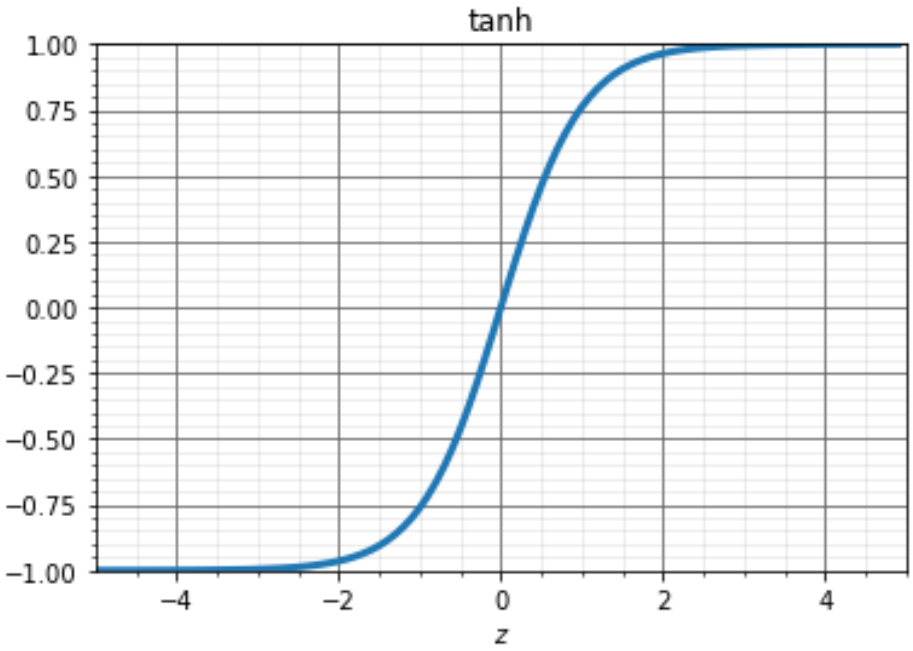

Ilustración 15 Ilustración 6 Representación gráfica de la función tanh.

$$ReLU\ (z) = z^{+} = \max(0, z) \tag{39}$$

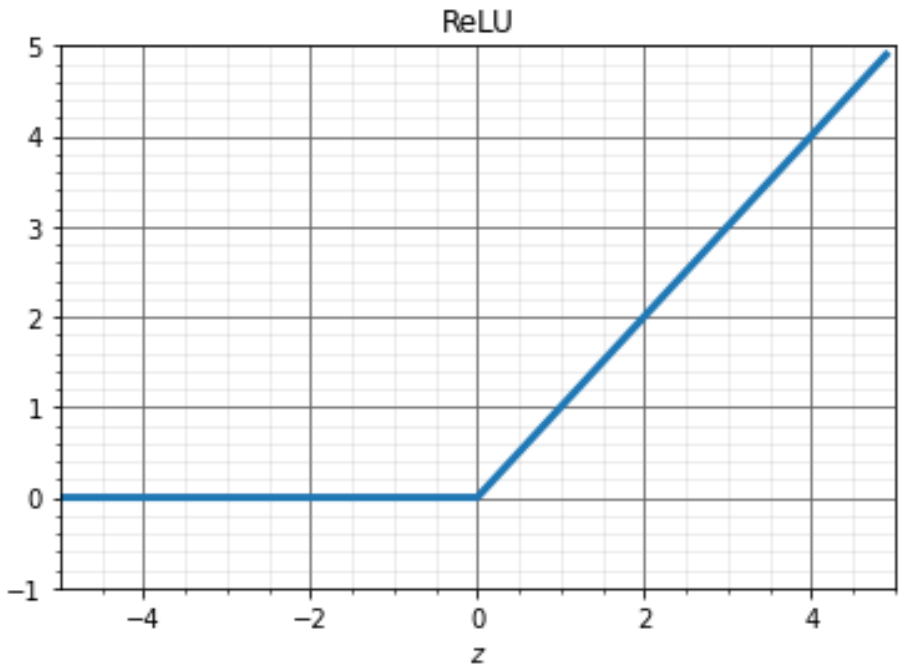

Ilustración 16 Ilustración 6 Representación gráfica de la función ReLU.

De la misma manera, si se asume que se tiene un único nodo en la capa de salida y entonces la matriz de peso $W^{(2)}$ que conecta la capa oculta con la capa de salida es de $H \times 1$, en un problema de regresión la salida puede expresarse de la siguiente manera:

$$a^{(3)} = f\left(z^{(3)}\right) = W^{(2)}a^{(2)} \tag{40}$$

Entonces, ¿cómo se puede obtener los pesos óptimos del modelo?, de manera muy similar a la regresión logística se puede utilizar el método del gradiente descendiente con el objetivo de minimizar una función de costo, como lo puede ser el error medio cuadrático.

Aquí la diferencia es que los gradientes son calculados mediante un algoritmo llamado *backpropagation*. Después de cada cálculo hacia delante en la red neuronal, un cálculo hacia atrás es realizado para ajustar los parámetros del modelo.

Esto quiere decir que el cómputo del gradiente es de manera inversa, se calcula el gradiente de la última capa primero y el de la primera capa es calculado al último. La parte de "propagación" se refiere a que cálculos parciales del gradiente de una capa son reutilizados para el cómputo del gradiente de la capa anterior.

Entonces el algoritmo de *backpropagation* se puede resumir en:

1. Se viaja a través de la red desde la entrada hasta la salida y calculamos los valores de salida tanto de la capa oculta como de la capa de salida, $a^{(2)}$ y $a^{(3)}$ respectivamente.
2. Para la última capa, calcular la derivada de la función de costo con respecto de la entrada a la capa de salida:

$$\delta^{(3)} = \frac{\partial}{\partial z^{(3)}} J(W) = -\left(y - a^{(3)}\right) \cdot f'\left(z^{(3)}\right) = a^{(3)} - y \tag{41}$$

3. Para la capa oculta calcular la derivada de la función de costo con respecto a la entrada a la capa oculta:

$$\delta^{(2)} = \frac{\partial}{\partial z^{(2)}} J(W) = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial}{\partial z^{(3)}} J(W) = \left(\left(W^{(2)}\right)\delta^{(3)}\right) \cdot f'\left(z^{(2)}\right) \tag{42}$$

4. Calcular los gradientes mediante la aplicación de la regla de la cadena:

$$\Delta W^{(2)} = \frac{\partial J(W)}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial W^{(2)}} = \delta^{(3)} a^{(2)} \tag{43}$$

$$\Delta W^{(1)} = \frac{\partial J(W)}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial W^{(1)}} = \delta^{(2)} x \tag{44}$$

5. Actualizar los pesos con los gradientes calculados y la tasa de aprendizaje:

$$W^{(1)} \coloneqq W^{(1)} - \frac{1}{m} \alpha \Delta W^{(1)} \tag{45}$$

$$W^{(2)} \coloneqq W^{(2)} - \frac{1}{m} \alpha \Delta W^{(2)} \tag{46}$$

Donde m es el número de muestras; entonces actualizas todos los pesos de manera repetida retomando estos mismos pasos con los últimos pesos calculados hasta que la función de costo convenir.

Una red neuronal es muy poderosa ya que puede encontrar cualquier característica de los datos con la estructura correcta, sin embargo, esto no la exenta de la debilidad que puede sufrir cualquier algoritmo de *machine learning*, y eso es el *overfitting* si no se tiene suficiente control sobre los procesos de aprendizaje.

Existen tres principales métodos para imponer restricciones en una red neuronal para evitar el *overfitting*: Regularización L1/L2, Método *dropout*, y *early stopping*.

El método de regularización L1/L2 se puede resumir en que consiste en agregar un término adicional a la función de costo. El método de *dropout* ignora cierto conjunto de nodos en la capa oculta durante la etapa

de aprendizaje, estos nodos son seleccionados de manera aleatoria. En *early stopping* el entrenamiento de la red finalizará si el rendimiento del modelo no mejora por un cierto número de iteraciones.

Capítulo 4. Metodología

4.1 ¿QUÉ DATOS SE NECESITAN?

Li Daolun et. al. en su artículo *Automatic well test interpretation based on convolutional neural network for a radial composite reservoir* primero recolectaron 200,000 gráficas log-log de presión que representaban yacimientos compuestos, con sus correspondientes combinaciones de parámetros de yacimientos como lo son la movilidad (donde se encuentra incluida la permeabilidad), el daño y el coeficiente de almacenamiento adimensional que, en otras palabras, son los parámetros objetivo de nuestra predicción. Cabe destacar que de estas pruebas 199,820 fueron generadas por métodos analíticos.

Estas gráficas que contenían la información para el entrenamiento fueron pre procesadas tomando únicamente 100 puntos de información, y después se construyó una matriz de 100 x 100 para alimentar el algoritmo de CNN.

Este enfoque permite generar los datos necesarios para poder entrenar una red neuronal, además de encaminar al pre procesamiento de los datos. Para efectos de esta investigación ya se mencionó a comienzos del capítulo 2 que se trabajará con el escenario más sencillo que puede presentar una prueba de presión, un único pozo produciendo a una sola fase, de una única capa en un yacimiento homogéneo. Y el modelo que representa este caso es también uno que dio al nacimiento de una de las curvas tipo más utilizadas en el análisis de pruebas de presión, el modelo presentado por Gringarten en su artículo *A comparison between different skin and wellbore storage type-curves for early-time transient analysis.* Dicho modelo está resumido en la siguiente ecuación que se encuentra en el espacio de Laplace:

$$L\{P_D\} = \frac{K_0\left(\sqrt{p}\right) + S\sqrt{p}\,K_1\left(\sqrt{p}\right)}{p\left\{\sqrt{p}\,K_2\left(\sqrt{p}\right) + C_D\,p\,\left[K_0\left(\sqrt{p}\right) + S\sqrt{p}\,K_1\left(\sqrt{p}\right)\right]\right\}} \tag{47}$$

Donde K_0 y K_1 representan funciones de Bessel modificadas del segundo tipo de orden cero y uno. De igual manera, en esta ecuación podemos apreciar cómo influyen el daño y el almacenamiento en la respuesta de la presión. La permeabilidad entra en juego cuando se pasa de la presión adimensional a la real con ayuda de la ecuación (24).

Entonces mediante un código de Python (mostrado en el Apéndice A) es posible generar cuantas curvas se deseen a partir de distintas combinaciones de permeabilidad, daño y almacenamiento del pozo, como lo muestra la siguiente figura:

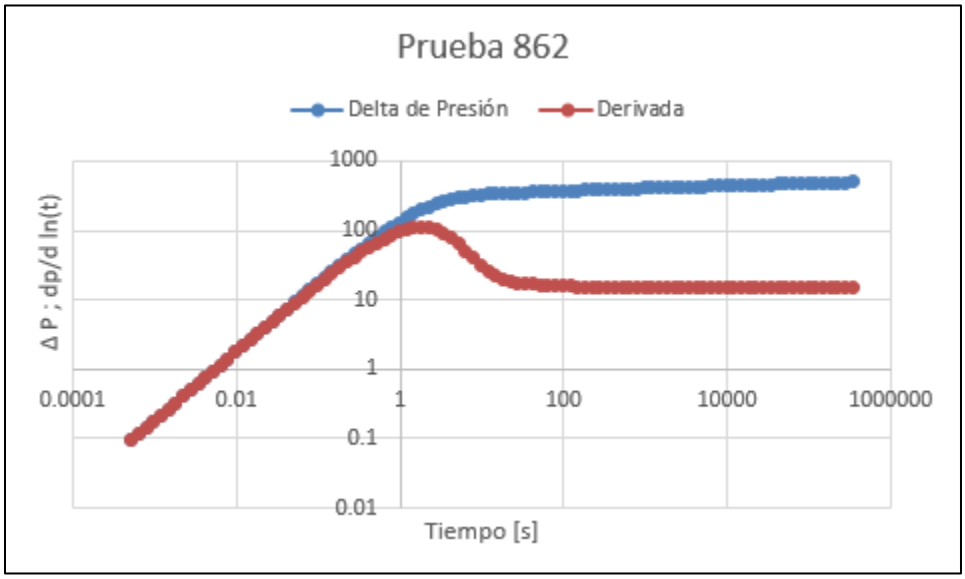

Ilustración 17 Curva generada por nosotros a través de la solución de la ecuación de Gringarten.

Para esta investigación se generaron 10,000 curvas con un rango de valores como se muestra en la siguiente tabla:

Tabla 4.1 Rango de valores para la generación de curvas de presión.

Permeabilidad (*k*)	10 – 120 [md]
Daño (*S*)	0 – 6.5
Almacenamiento del pozo adimensional (C_D)	100 - 5000
Gasto (*q*)	174 – 300 [BPD]
Porosidad (Φ)	0.1 – 0.5

4.2 PREPROCESAMIENTO DE LOS DATOS

Cada curva generada se guarda en un archivo de Excel donde podemos encontrar los datos de tiempo, delta de presión y derivada de la presión. Así como los datos con las que se generó dicha curva. Por lo que cuando se importen todos los datos para entrenar un modelo de machine learning, será posible guardarlos dentro de una misma matriz para su manipulación.

a)

Tiempo	Presión	Derivada
0.000515	3.568774	3.441071
0.000635	4.365384	4.182739
0.000783	5.332702	5.071627
0.000966	6.50417	6.131356
0.001191	7.918392	7.386799
0.001468	9.6193	8.862842
0.00181	11.65599	10.58242
0.002231	14.08208	12.56366
0.002751	16.95429	14.81584
0.003391	20.33015	17.33425

b)

permeabilidad	Daño	Almacenamiento
20	5	1500

Ilustración 18 a) Datos guardados de cada curva (donde el tiempo está en segundos y la presión en psi). b) Datos guardados con los que se generó cada curva (donde la permeabilidad está en mD).

Un modelo de machine learning se alimenta de una matriz 2D, para este punto se está trabajando con una matriz de datos 3D, donde una dimensión es el número de pruebas, la segunda son las columnas de tiempo, delta de presión y derivada de presión, y la tercera son los datos de cada una de esas columnas, tal y como se muestra en la ilustración 19.

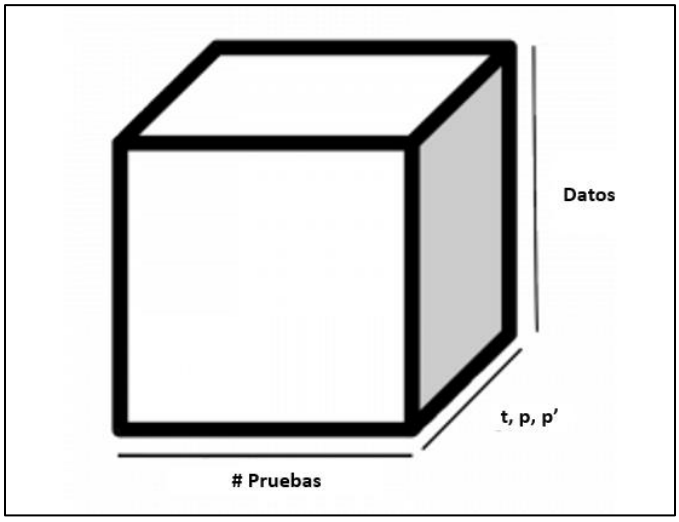

Ilustración 19 Matriz 3D que se obtiene de la importación de todos los datos generados.

Se tiene que transformar dicha matriz 3D a una 2D, esto se logra mediante un desdoblamiento en el que para cada prueba ponemos sus respectivos datos de tiempo, delta de presión, y derivada de presión dentro de una misma columna. O, dicho de otra manera, se pasa de una matriz con dimensiones

$\{\#Pruebas\ x\ t, p, p' x\ Datos\}$ a una con dimensiones $\{\#Pruebas\ x\ (t, p, p' \cdot Datos)\}$, como se ejemplifica en la ilustración 20.

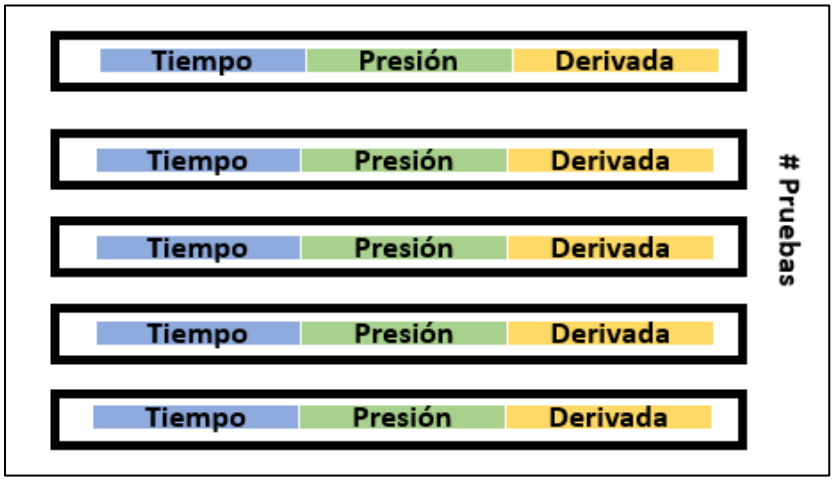

Ilustración 20 Matriz 2D que resulta del "desdoblamiento" de la matriz 3D de datos generados.

Para el último paso de pre procesamiento de los datos es relevante el trabajo de Rouhollah Ahmadi et. al. en *Automatic well-testing model diagnosis and parameter estimation using artificial neural networks and design of experiments*, donde se puede ver la combinación de distintos modelos de redes neuronales para tanto la clasificación del modelo del yacimiento como para la estimación de parámetros.

Aquí los autores después de la obtención de puntos de las curvas de derivadas aplican una serie de normalizaciones sobre los datos, ya que algunos algoritmos de machine learning dependen de que los datos no se extiendan sobre grandes rangos de valores. En el caso de este trabajo esto se logra con ayuda de una función dentro de la paquetería de *sklearn* llamada *StandarScaler().*

En este artículo también podemos encontrar con una reducción de dimensiones de los mismos datos, donde los autores solo escogen puntos relevantes dentro de las curvas para la construcción del set de datos. En esta investigación se seleccionan 30 puntos para representar a cada curva de presión.

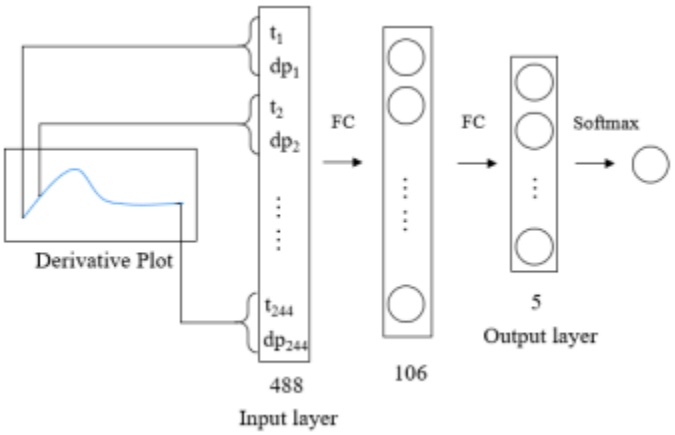

Ilustración 21 Diagrama que representa los datos de entrada en la red neuronal. (de Hongyang Chu et. al.)

Entonces de estos datos 8000 pruebas serán utilizadas para entrenar al modelo por lo que nuestra matriz de entrenamiento será una con las dimensiones de $8000\ x\ 90$ (8000 pruebas de entrenamiento, de las cuales se tomaron 30 puntos específicos de las tres columnas generadas para cada prueba, dichas columnas son tiempo, presión y derivada de presión), mientras que las 2000 restantes serán utilizadas como sets de validación y prueba, con lo que será posible calcular que tan bueno es nuestro modelo al predecir la interpretación de las distintas pruebas de presión.

4.3 CONSTRUCCIÓN DE LA RED NEURONAL

Para la construcción del modelo creado en este trabajo se utilizó la paquetería de *Tensorflow* que cuenta con una gran cantidad de algoritmos y herramientas de machine learning. Como el modelo a construir va a resolver un problema de regresión con 3 variables de salida tenemos que construir nuestro modelo capa por capa con ayuda del API funcional de *Tensorflow.* Esto significa que se comportara como un árbol que se extiende con sus ramas, todas las variables de salida se empezaran a calcular en un tronco principal, pero conforme avance el proceso cada una tomará una dirección diferente en ramas individuales.

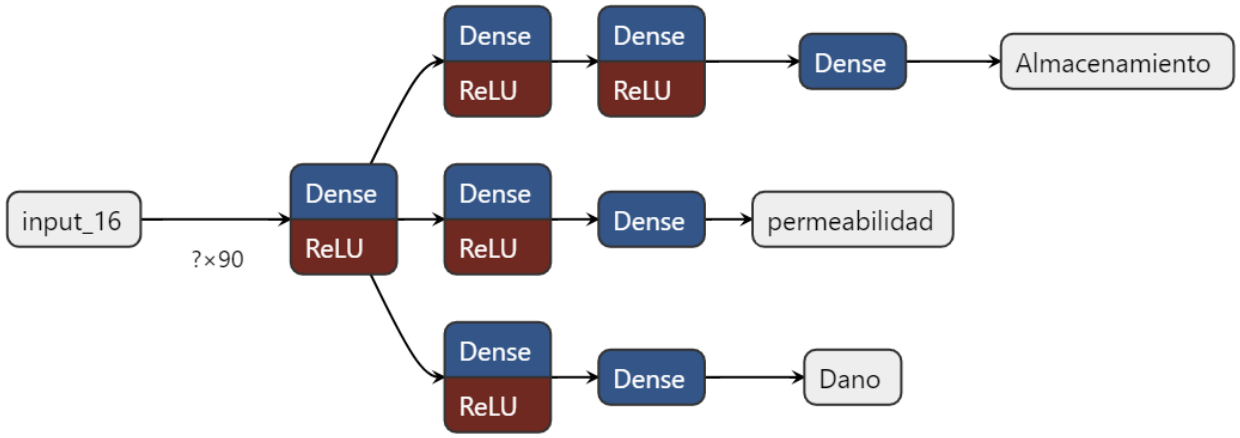

Ilustración 22 Esquema que representa la estructura de la red neuronal construida.

Construyendo y entrenando este modelo es posible ya obtener una predicción para la permeabilidad, el daño y el almacenamiento adimensional. Pero aún se desconoce si es el mejor modelo que se pueda obtener, para determinar el mejor modelo se hace uso de una técnica conocida como *Grid Search* para optimizar los hiperpárametros más importantes dentro de una red neuronal. Está técnica consiste en definir un espacio de búsqueda como una malla que contenga todos los distintos valores que los diferentes hiperparametros puedan tener, y de forma cíclica evaluar cada una de las posiciones dentro de esta malla, es decir, crear un modelo para cada una de las distintas combinaciones de hiperpárametros que puedan existir. Los hiperpárametros a optimizar dentro de estar red neuronal son los siguientes:

- No. de capas
- No. de nodos en cada capa
- Tasa de aprendizaje
- Epochs (número de veces que el conjunto de datos de entrenamiento es pasado a través de la red neuronal)

Y el rango de valores que puede tener cada hiperparámetro dentro de esta malla se muestra en la siguiente tabla:

Tabla 4.2 Rango de valores de los hiperparámetros.

Número de Capas	3 a 5
Número de Nodos en cada capa	8 a 120
Tasa de aprendizaje	0.01 a 0.4
Epochs	300 a 1000

Se crearon 1458 modelos distintos, posteriormente cada uno de ellos fue puesto a prueba sobre un conjunto de datos de validación conformado por 1000 pruebas para así ser comparados utilizando las métricas de R^2 y error cuadrático medio como base para determinar cuál de ellos es el mejor.

Trial ID	Show Metrics	hidden_size	hidden_size2	hidden_size3	hidden_size4	hidden_size5	epochs	learning_rate	r2k	r2D	r2A
9791170f22a4d...	☐	20.000	8.0000	60.000	20.000	20.000	300.00	0.010000	0.99396	0.96023	0.99791
46d54b58dadcc...	☐	20.000	8.0000	60.000	8.0000	60.000	1000.0	0.010000	0.97809	0.95882	0.97426
834fcb804c8a6f...	☐	20.000	20.000	60.000	8.0000	8.0000	300.00	0.010000	0.99403	0.95255	0.97399
3f3065e757a25...	☐	20.000	8.0000	8.0000	20.000	20.000	300.00	0.010000	0.99088	0.95096	0.97925
c9e82af286f4c6...	☐	20.000	20.000	8.0000	20.000	8.0000	300.00	0.010000	0.99329	0.94803	0.99056
f0aadb22d65aa...	☐	20.000	8.0000	60.000	8.0000	20.000	1000.0	0.010000	0.94113	0.94489	0.98224
96d52ee3b28b8...	☐	20.000	20.000	20.000	8.0000	60.000	1000.0	0.010000	0.98673	0.94447	0.99408
acc70a825468f...	☐	20.000	8.0000	8.0000	8.0000	8.0000	300.00	0.010000	0.99410	0.94308	0.98528
53f0c97b4926e...	☐	20.000	20.000	20.000	60.000	8.0000	1000.0	0.010000	0.97027	0.94020	0.97163
103b08c238b5f...	☐	20.000	8.0000	60.000	60.000	8.0000	1000.0	0.010000	0.98282	0.94011	0.55724
6622bf12cd5d0...	☐	20.000	20.000	8.0000	60.000	20.000	1000.0	0.010000	0.98710	0.93587	0.74275
248a7b616e46f...	☐	20.000	20.000	60.000	60.000	20.000	300.00	0.010000	0.99112	0.92932	0.99476
7693f41fe0c92...	☐	20.000	20.000	20.000	8.0000	8.0000	300.00	0.010000	0.99585	0.92307	0.99038
ba48615c284c7...	☐	20.000	20.000	20.000	60.000	60.000	1000.0	0.010000	0.49729	0.91672	0.97297
6fe395d824d5d...	☐	20.000	20.000	8.0000	60.000	8.0000	1000.0	0.010000	0.98140	0.90453	0.69257
68ddee5462ef7...	☐	20.000	20.000	20.000	20.000	20.000	300.00	0.010000	0.99551	0.90426	0.97529
5f18774d0f5c2...	☐	20.000	8.0000	60.000	20.000	8.0000	300.00	0.010000	0.99591	0.90374	0.95518
1627bd08abe28...	☐	20.000	20.000	20.000	20.000	8.0000	300.00	0.010000	0.98524	0.90369	0.99363
9e3ef92a12044...	☐	20.000	20.000	8.0000	60.000	60.000	1000.0	0.010000	0.99296	0.90306	0.95486
ff5a4eae32a92e...	☐	20.000	20.000	20.000	8.0000	60.000	300.00	0.010000	0.98395	0.90124	0.99149
8c5b60f81de74...	☐	20.000	20.000	20.000	8.0000	8.0000	1000.0	0.010000	0.99245	0.89983	0.97562
640af3fbe7bdbf...	☐	20.000	20.000	8.0000	8.0000	20.000	1000.0	0.010000	0.98756	0.89768	0.98241
6ae4d2c24e6b7...	☐	20.000	20.000	20.000	60.000	8.0000	300.00	0.010000	0.97387	0.89535	0.69731
12fea270c3efd5...	☐	20.000	20.000	60.000	20.000	60.000	300.00	0.010000	0.98776	0.89266	0.98985
6f011577c4bea...	☐	20.000	8.0000	8.0000	20.000	8.0000	300.00	0.010000	0.97688	0.89046	0.95109
60de4f4dcafd1...	☐	20.000	20.000	20.000	60.000	60.000	300.00	0.010000	0.94861	0.88919	0.94852
3eeba35d39e46...	☐	20.000	8.0000	8.0000	20.000	20.000	1000.0	0.010000	0.97880	0.88774	0.94449
40a607a866b05...	☐	20.000	8.0000	60.000	8.0000	60.000	300.00	0.010000	0.98926	0.88598	0.99775
8c888acc0bb00...	☐	20.000	20.000	8.0000	20.000	20.000	1000.0	0.010000	0.96892	0.88329	0.97699
52c3095ab31cd...	☐	20.000	20.000	8.0000	8.0000	8.0000	300.00	0.010000	0.99655	0.87755	0.99782
0c05a61221d9c...	☐	20.000	20.000	8.0000	8.0000	20.000	300.00	0.010000	0.98750	0.87659	0.98631

Ilustración 23 Resultados sobre que híper parámetros son los mejores.

En la ilustración 23 se muestra un listado de los distintos modelos creados ordenados de mejor rendimiento a menor, siendo la mejor red neuronal la que encabeza la lista y que se ve de la siguiente manera:

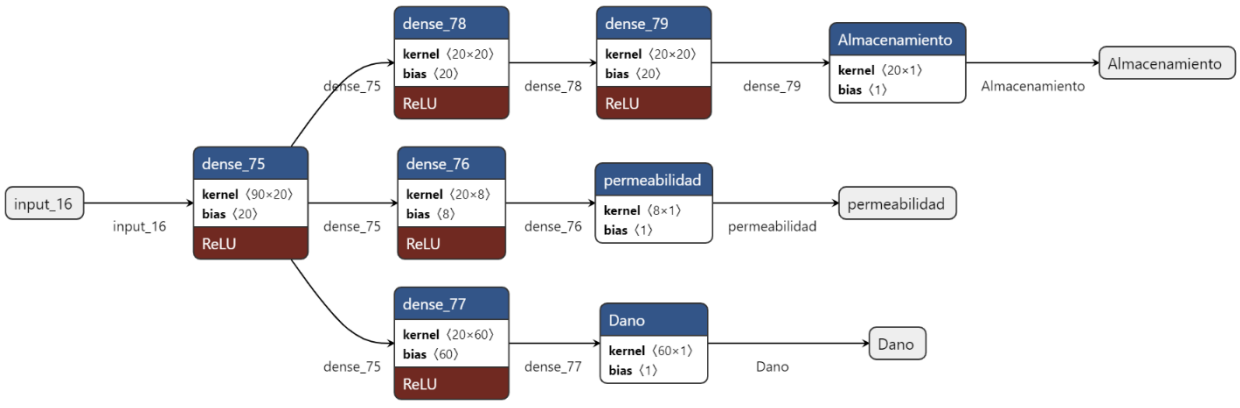

Ilustración 24 Mejor red neuronal del proceso de Grid Search.

En la ilustración 24 se puede apreciar que para esta red neuronal en la primera capa se cuenta con 20 nodos, en las siguientes se ramifica teniendo así para la permeabilidad una capa con 8 nodos, para el daño una capa de 60 nodos, mientras que para el caso del almacenamiento con base en esta investigación se encontró el mejor resultado con dos capas de 20 nodos cada una. De igual manera en esta imagen se observa la

función de activación *ReLU* que se utilizó a lo largo de toda la red y que se explicó en el capítulo 3 de este trabajo. Esta red cuenta con un poder predictivo de $R^2 = 0.9838$ para el conjunto de datos de validación.

Capítulo 5. Análisis de Resultados

5.1 EFICIENCIA DEL MODELO

Primero se puso a prueba la red neuronal contra las 1000 pruebas restantes que conforman el conjunto de datos de pruebas. La siguiente tabla resumen los resultados de esta prueba para R^2.

Tabla 5.1 Eficiencias de predicción de la red neuronal en el conjunto de datos de prueba.

Indicador	Valor
R^2 de permeabilidad	0.978
R^2 de Daño	0.932
R^2 de Almacenamiento	0.957
R^2 General	0.956

El valor de 1 en R^2 implicaría una perfecta correlación entre los valores predichos por la red y los reales, significaría que no existe variancia entre estos dos valores, es por ello que buscamos acercarnos a ese valor en la mayor medida posible. Sin embargo, es importante también mencionar que valores muy cercanos a 1 de R^2 pueden ser una señal de *overfiting,* fenómeno que, como se ha mencionado en capítulos anteriores, desmerece al modelo de todo poder predictivo que pueda poseer.

Tabla 5.2 Error cuadrático medio (ECM) de predicción de la red neuronal sobre el conjunto de datos de prueba.

Indicador	Valor
ECM de permeabilidad	22.1098
ECM de Daño	0.2379
ECM de Almacenamiento	2930.37

La tabla 5.2 resume los errores cuadráticos medios para cada uno de los valores que predice la red neuronal. Mayor el valor mayor es el error, aunque se vean números realmente grandes no hay porque pensar que el modelo es malo, de hecho, no hay valor correcto a obtener en cuanto al ECM, pero es de ayuda para identificar si ha habido algún problema de *overfiting.* También se concluye que en el caso del almacenamiento el error es mucho mayor debido a que los valores a estimar van desde el rango de las centenas hasta los millares, es decir, si el valor predicho falla en estimar el valor real por 50 unidades por ejemplo, esas 50 unidades representan poco en comparación a las magnitudes que maneja el

almacenamiento, pero aun así presentará un error mucho mayor a el daño o la permeabilidad que trabajan con valores mucho más pequeños en relación con el almacenamiento.

Estos resultados están dentro de nuestros estándares para concluir que la red es capaz de seguir la tendencia de los datos.

5.2 COMPARACIÓN CON OTROS ALGORITMOS DE MACHINE LEARNING

Se entrenaron otros algoritmos de machine learning para que sirvieran como fundamento de la justificación de haber seleccionado una red neuronal para este problema y no cualquiera de los otros algoritmos aquí presentados. En la tabla 5.3 podemos observar los resultados de esta comparación.

Tabla 5.3 Comparación de distintos algoritmos de machine learning.

Modelo	Eficiencia (R^2) promedio
Red Neuronal	0.956
K-nearest neighbors	0.818
Random Forest	0.674
Decision tree	- 0.148
Regresión lineal	- 360.5

Todos los modelos fueron entrenados con el mismo conjunto de datos de entrenamiento, de igual manera los valores de R^2 fueron obtenidos a partir de poner a prueba a cada modelo con el set de datos de prueba.

La red neuronal encabeza la lista con una mayor eficiencia de predicción, seguida de un algoritmo de similar complejidad como lo puede ser el *K-nearest neighbors.* Aunque estos dos encabezan la tabla existe una diferencia bastante grande entre ambos, y a partir de ahí la eficiencia disminuye drásticamente hasta llegar algoritmos mucho más sencillos como los son los *decisión tree* o una regresión lineal que no son capaces de seguir la tendencia de los datos, de ahí que arrojen valores de R^2 negativos.

5.3 EL MODELO CON DATOS AJENOS AL CONJUNTO DE DATOS SINTETICOS

El funcionamiento de la herramienta consiste en tomar una prueba de presión cualquiera que se quiera interpretar, en este trabajo se tomó como ejemplo un par de pruebas que se encuentran en el libro *Applied Well Test Interpretation* de John P. Spivey y W. John Lee. Una de estas pruebas es la que se resolvió como ejemplo de análisis durante el capítulo 3 de este mismo trabajo de tesis por lo que los datos de esta prueba están en la tabla 2.4 y se encuentran graficados en la ilustración 4.

Los resultados obtenidos de manera manual de una interpretación real contra los predichos por la red neuronal se comparan en la tabla 5.4.

Tabla 5.4 Resultados de un análisis manual contra los predichos por la red neuronal de la prueba de presión presentada en la tabla 2.4

Dato	Análisis Manual	Predicción	Error relativo
Permeabilidad	$25.4\ [md]$	$29.59\ [md]$	16.49 %
Daño	6.7	6.098	8.98 %
Almacenamiento adimensional	145	155.37	7.15 %

Observando los errores relativos de la tabla 5.4 se puede ver que la red neuronal es capaz de predecir con cierta fidelidad el resultado real. En esta ocasión el mayor error se presenta en la predicción de la permeabilidad, pero como se verá más adelante, este no siempre es el caso.

Otra prueba de presión con la que se puede comprobar la efectividad de la red neuronal es la que se presenta en la tabla 5.5 y que se encuentra graficada en la ilustración 25.

Tabla 5.5 Datos de tiempo y presión de segundo ejemplo de prueba de presión.

t hr	$\boldsymbol{p_{wf}}$ psia	t hr	$\boldsymbol{p_{wf}}$ psia	t hr	$\boldsymbol{p_{wf}}$ psia
0	3032.27	0.163	2960.04	4.105	2833.33
0.001	3031.56	0.1844	2956.49	4.619	2824.25
0.0021	3030.55	0.2085	2952.91	5.198	2814.62
0.0034	3029.27	0.2655	2949.29	5.848	2804.48
0.0048	3027.65	0.266	2945.64	6.581	2793.89
0.0064	3026.78	0.3002	2941.94	7.404	2782.94
0.0082	3025.8	0.3387	2938.21	8.331	2771.7
0.0102	3021.59	0.3821	2934.43	9.373	2760.29
0.0125	3019.25	0.4308	2930.6	10.55	2748.81
0.0151	3016.78	0.4857	2926.71	11.87	2737.36
0.018	3014.19	0.5474	2922.76	13.35	2726.04
0.0212	3011.5	0.6168	2918.73	15.02	2714.93
0.0249	3008.71	0.6949	2914.63	16.9	2704.12
0.029	3005.84	0.7828	2910.42	19.01	2693.68
0.0336	3002.89	0.8816	2906.11	21.39	2683.65
0.0388	2999.87	0.9928	2901.66	24.06	2674.09
0.0447	2996.76	1.118	2897.05	27.07	2665.02

0.0512	2993.66	1.259	2892.26	30.46	2656.46
0.0587	2990.48	1.417	2887.24	34.26	2648.43
0.067	2983.97	1.595	2881.96	38.55	2640.91
0.0764	2980.65	1.796	2876.36	43.37	2633.91
0.0869	2977.3	2.021	2870.4	48.79	2627.4
0.0988	2973.92	2.275	2864.01	54.79	2621.38
0.1121	2970.5	2.56	2857.15	60.79	2615.82
0.1271	2967.04	2.881	2849.77	66.79	2610.68
0.144	2963.56	3.648	2841.84	72	2605.93

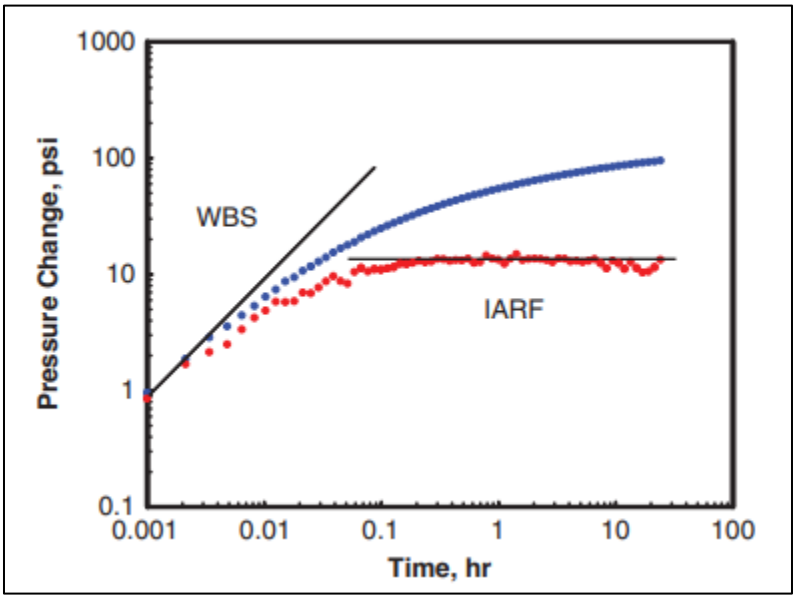

Ilustración 25 Prueba obtenida de un ejemplo de aplicación de Applied Well Test Interpretation, 2013).

De primera instancia es posible ver a simple vista que esta prueba se ve muy distinta a la del ejemplo anterior. Los resultados obtenidos de manera manual de una interpretación real contra los predichos por la red neuronal se comparan en la tabla 5.6.

Tabla 5.6 Resultados de un análisis manual contra los predichos por la red neuronal de la prueba de presión presentada en la tabla 5.5

Dato	Análisis Real	Predicción	Error relativo
Permeabilidad	36.7 $[md]$	39.51 $[md]$	7.66 %
Daño	-2.6	-1.061	59.19 %

Almacenamiento adimensional	93	112.67	21 %

En esta ocasión la red neuronal predice de mejor manera el valor de la permeabilidad en contraste con el ejemplo anterior, en esta ocasión donde existe una mayor discrepancia es en el almacenamiento adimensional pero los valores siguen dentro de un rango razonable. El caso del daño en este ejemplo merece una mención totalmente aparte.

Recordando la tabla 4.1 donde se mencionan los rangos de valores de permeabilidad, daño y almacenamiento adimensional que se utilizaron para generar el conjunto de datos sintéticos utilizados para el entrenamiento, validación y prueba de la red neuronal es posible notar que el rango de daño que se utilizó iba desde 0 hasta 6.5, en ningún momento se le mostró a la red una prueba con daño negativo, sin embargo fue capaz de reconocer que la prueba no se parecía a ninguna que había visto con anterioridad y decidió dar una predicción negativa.

5.4 VALIDACIÓN DEL MODELO CON DATOS DE PRUEBAS DE PRESIÓN DE CAMPOS MEXICANOS

A continuación, se analizaron pruebas de presión reales de manera manual, después estos resultados fueron corroborados con ayuda del software de interpretación de pruebas de presión Harmony WellTest 2019.1 propiedad de IHS Markit, para al final compararlos con los resultados arrojados por la interpretación de la red neuronal.

- **Prueba 1:**

La primera prueba que se utilizó proviene del campo San Andrés en Poza Rica Veracruz, del pozo SA109 - 83, las especificaciones de esta prueba pueden encontrarse en las tablas 5.7 y 5.8.

Tabla 5.7 Datos del pozo SA109 - 83

Porosidad (Φ)	0.12
Espesor (h)	164 [ft]
Factor de volumen del aceite (B_o)	1.01
Viscosidad (μ)	0.7 [cp]
Compresibilidad total (C_t)	$2.5 * 10^{-5}$ [psi^{-1}]
Gasto de aceite (Q_o)	655 [BPD]
Radio del pozo (r_w)	0.31 [ft]

Tabla 5.8 Datos de la prueba de presión del pozo SA109 -83

Tiempo [h]	Presión [psi]
0	7274.95215
1	7192.47607
2	7152.66016

5	7124.22022
11	7070.18408
17	7053.12012
23	7030.36816
29	7020.41406
35	7011.88184
41	7003.3501
47	6997.66211
53	6993.396
59	6989.12988
67	6984.86377
69	6980.59815

Calculando la diferencia de presión y graficándola los datos de presión de dicha prueba se obtiene la siguiente curva:

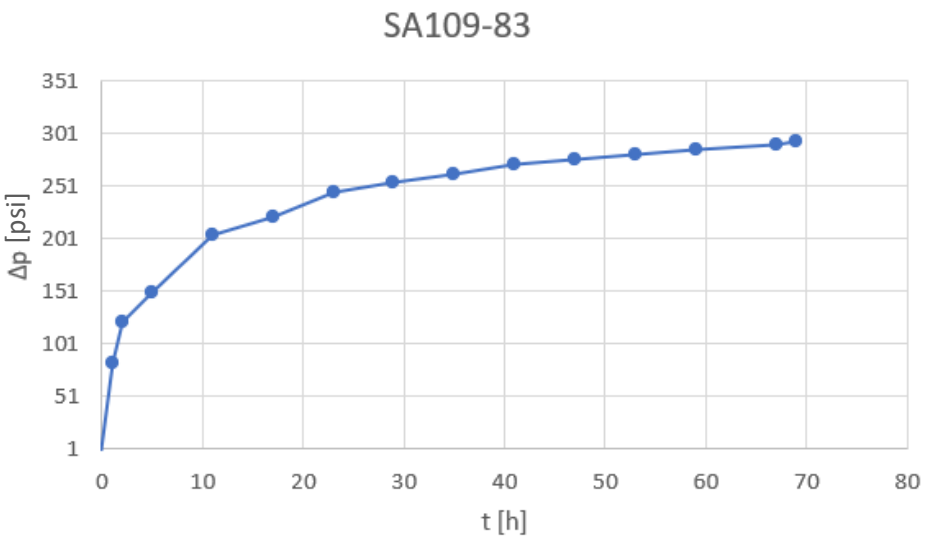

Ilustración 26 Diferencia de presión contra tiempo del pozo SA109 - 83.

Calculando la derivada y graficándola en una escala logarítmica junto con la diferencia de presión se obtiene la gráfica que permite la interpretación de la prueba:

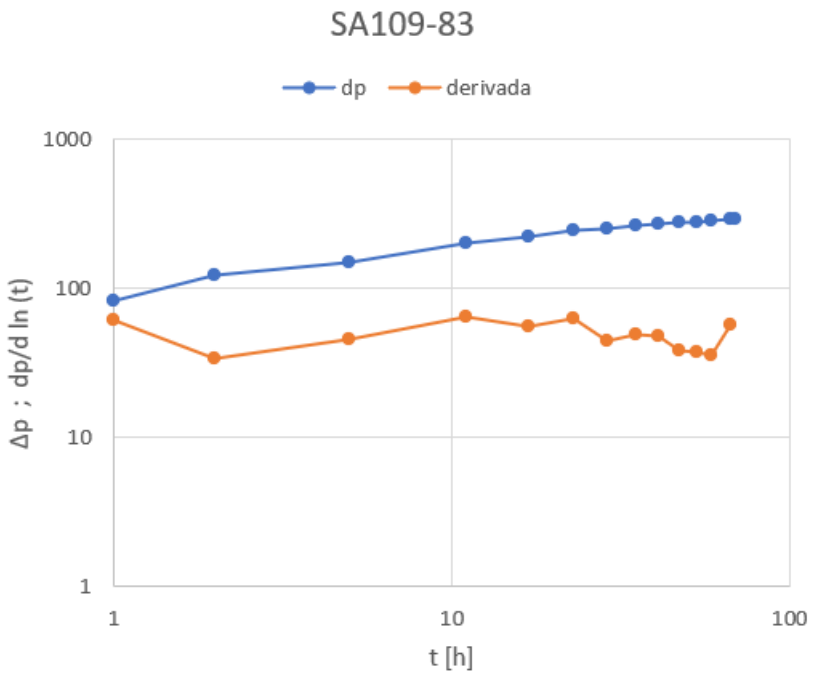

Ilustración 27 Gráfica log-log del pozo SA109 - 83.

Con ayuda de la curva tipo de Gringarten – Bourdet se pueden obtener los valores de interés para este trabajo.

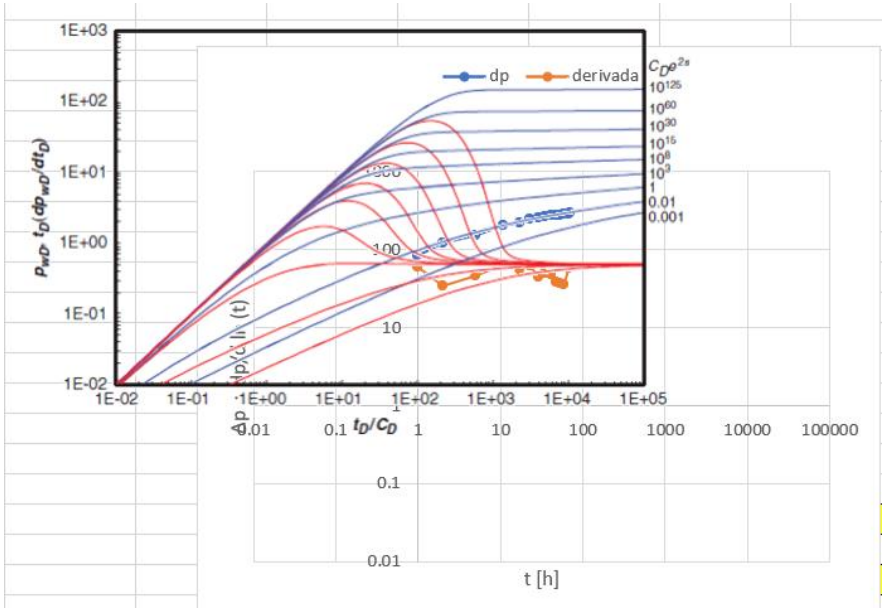

Ilustración 28 Análisis de la prueba del pozo SA109 - 83 con curva tipo Gringarten - Bourdet.

Tomando como *match* los puntos (10,1000) para $(p_D/\Delta p)_{MP}$ y (1,100) para $(t_D/t)_{MP}$ y utilizando las ecuaciones 27, 28 y 30 se obtiene que:

$$k = \frac{655(1.01)(0.7)}{0.00708(164)}\left(\frac{10}{1000}\right) = 3.988$$

$$C_D = \frac{0.0002637(3.988)}{(0.12)(0.7)(0.31^2)}\left(\frac{1}{100}\right) = 55.645$$

$$S = 0.5\ln\left(\frac{0.1}{55.645}\right) = -4.312$$

Estos datos se ven corroborados gracias al software Harmony WellTest:

Oil Well Test - Drawdown
Radial Flow Analysis

Analysis Results

Flow Capacity (kh)	246.7608	mD.m	Total Skin (s')	-3.708	
Effective Permeability (k)	4.9365	mD	Skin Due to Damage (s_d)	-3.708	
Effective Gas Permeability (k_g)		mD	Skin Due To Inclination (s_{tho})		
Effective Oil Permeability (k_o)	4.8365	mD	Skin Due To Partial Penetration (s_{PP})		
Effective Water Permeability (k_w)		mD	Pressure Drop Due to Total Skin (Δp_{Skin})		kPa(a)
Total Fluid Rate (in situ) ($q\beta_t$)	104.1450	rm³/d	Dimensionless Wellbore Storage (C_D)	67.439	
Total Mobility ($(k/\mu)_t$)	7.0521	mD/mPa.s	Wellbore Storage (C)	9.431e-04	m³/kPa
Total Transmissivity ($(kh/\mu)_t$)	352.5154	mDm/mPa.s			
Slope (m)	29.49	kPa/cycle			

Reservoir Parameters

Net Pay (h)	49.987	m
Total Porosity (ϕt)	0.12	%
Gas Saturation (S_g)	0.00	%
Oil Saturation (S_o)	100.00	%
Water Saturation (S_w)	0.00	%
Formation Compressibility (c_f)	3.625e-09	1/kPa
Total Compressibility (c_t)	3.625e-09	1/kPa
Wellbore Radius (r_w)	0.094	m

Fluid Properties

Reservoir Temperature (T_{resv})	100.0	°C
Reservoir Pressure (p_{resv})	50159	kPa(a)
Oil Gravity (γ_o)	30	°API
Oil Viscosity (μ_o)	0.7	mPa.s
Oil Compressibility (c_o)	1.75e-06	1/kPa
Oil Formation Volume Factor (B_o)	1.01	m³/m³
Solution Gas Ratio (R_s)	152.37	m³/m³
Oil Correlation	Vasquez and Beggs	
Oil Viscosity Correlation	Beggs & Robinson	

Production and Times

Corrected Time (t_o)	69.0	h
Total Cumulative Production Oil (Cum_{oil})	299	m³
Final Oil Rate ($q_{o\ final}$)	104.1450	m³/d

Ilustración 29 Reporte de Harmony WellTest 2019.1.

El resumen de la comparación entre el análisis manual, con ayuda de un software profesional y la interpretación de la red neuronal se presenta en la tabla 5.9.

Tabla 5.9 Comparación de resultados de distintas interpretaciones para el pozo SA109 - 83

	Análisis Manual	Harmony WellTest 2019.1	Red Neuronal
Permeabilidad (*k*)	3.988 [mD]	4.936 [mD]	9.222 [mD]
Daño (S)	-4.312	-3.708	-1.9476
Almacenamiento Adimensional (C_D)	55.645	67.439	98.057

Se observa que no existe gran diferencia entre los análisis manuales y el obtenido por el software, sin embargo la red neuronal se aleja más de los resultados verdaderos, estas diferencias se atribuyen al conjunto de datos de entrenamiento que dieron origen a la red neuronal, ya que en primera instancia estas pruebas entran en valores que no abarcan los rangos utilizados para la generación sintética de pruebas de presión, además, hay que considerar que dentro del conjunto de datos de entrenamiento no se encuentra ninguna prueba real.

- **Prueba 2:**

Esta segunda prueba proviene igualmente del campo San Andrés en Poza Rica Veracruz, del pozo SA175 - 69, las especificaciones de esta prueba pueden encontrarse en las tablas 5.10 y 5.11.

Tabla 5.10 Datos del pozo SA175 - 69

Porosidad (Φ)	0.15
Espesor (h)	164 [ft]
Factor de volumen del aceite (B_o)	1.01
Viscosidad (μ)	0.7 [cp]
Compresibilidad total (C_t)	$2.5 * 10^{-5}$ [psi^{-1}]
Gasto de aceite (Q_o)	1153 [BPD]
Radio del pozo (r_w)	0.36 [ft]

Tabla 5.11 Datos de la prueba de presión del pozo SA175 -69

Tiempo [h]	Presión [psi]
0	6098.95801
1	5857.21777
2	5715.01807
3	5624.00977
4	5548.64404
5	5494.60791
6	5454.79199
7	5420.66406
8	5389.37988
9	5366.62793
10	5345.29785
11	5325.39014

12	5308.32617
13	5294.10596
14	5278.46387
15	5267.08789
16	5257.13428
17	5247.18018
18	5237.22607
19	5230.11572
20	5223.00586
21	5213.05225
22	5205.94238
23	5200.2544
24	5190.30029

Realizando el mismo flujo de trabajo que en la prueba 1 obtenemos la tabla de comparación 5.12:

5.12 Comparación de resultados de distintas interpretaciones para el pozo SA175 - 69

	Análisis Manual	**Harmony WellTest 2019.1**	**Red Neuronal**
Permeabilidad (*k*)	4.212 [mD]	5.498 [mD]	11.379 [mD]
Daño (S)	-0.591	1.293	1.324
Almacenamiento Adimensional (C_D)	3265.116	3388.942	2950.849

Se puede apreciar como la red neuronal vuelve a discrepar con los otros dos análisis, sin embargo, sigue siendo capaz de encontrar estimaciones cercanas y en este caso en particular, logra acercarse de mejor manera, muy probablemente debido a que, en esta ocasión, varios de los valores a estimar se encuentran dentro de los rangos de las pruebas sintéticas que fueron utilizadas en el entrenamiento.

Capítulo 6. Conclusiones.

Se logró desarrollar una herramienta computacional aplicando Machine Learning (ML) que fuera capaz de seguir la tendencia que muestran las pruebas de presión, para así obtener una interpretación automática que arroja valores de permeabilidad, daño y almacenamiento, para el caso de un pozo que produce en un yacimiento homogéneo. Esta interpretación está dentro de márgenes de error aceptables que pueden servir al ingeniero de yacimientos como un análisis sencillo, rápido y confiable, y que a su vez lo pueden guiar a través de una ambigüedad de identificación de modelos al momento de hacer un análisis más detallado.

Para lograr el objetivo anterior se investigaron los conceptos básicos que intervienen en la interpretación adecuada de una prueba de presión, así como distintas metodologías para llevarla a cabo. Una investigación de igual magnitud se realizó relacionada con algoritmos de *machine learning* y su funcionamiento especifico, aprendiendo lo referente a su construcción y aplicación.

Durante el desarrollo de este trabajo se generó de manera sintética un conjunto de pruebas de presión que fungirían el papel de datos de entrenamiento para el modelo de *machine learning*, y que se puso a prueba posteriormente con respecto a pruebas obtenidas de la literatura totalmente ajenas a los datos generados. Todo esto partió de una búsqueda amplia y entendimientos de las herramientas que ofrecen librerías de Python, como los son *tensorflow* y *SKLearn*.

Se probó la respuesta de la red neuronal ante pruebas de presión reales de campos mexicanos, y aunque los errores relativos de sus predicciones son altos, la red neuronal es capaz de seguir una tendencia que no difiere en gran medida con los resultados reales, mostrando así que de contar con el conjunto de entrenamiento adecuado, se pueden conseguir predicciones mucho más fiables de modelos más sofisticados.

Se concluye que el *machine learning* es una herramienta que también puede tener un rol bastante significativo en la industria petrolera en el área de caracterización de yacimientos, donde los problemas son del carácter inverso y de alta complejidad. Se pueden acondicionar los datos obtenidos de pruebas de presión, para que un algoritmo como las redes naturales logren entenderlos y aprender de él, para posteriormente generar sus propias predicciones.

6.1 ALCANCES Y LMITACIONES

Los modelos presentados en este trabajo no dejan de ser redes neuronales muy nobles, donde la cantidad de datos fue una limitante. Esto se puede apreciar en la mediana simpleza de la estructura de la red neuronal en sí misma, donde para problemas de alta complejidad se suele tener muchísimas más capaz de las que se mostraron aquí. Es ahí donde nace un área de oportunidad para la recolección y digitalización de recursos informativos sobre los diversos campos del país, con el objetivo de contar con una base de datos robusta que represente áreas de interés que puedan utilizarse en proyectos de esta índole.

También es importante recordar que en la presente tesis trabajamos con el modelo más sencillo de todos, si quisiéramos trabajar con modelos más complejos, como lo pueden ser los que describen a los yacimientos

naturalmente fracturados, serían necesario realizar todo el flujo de trabajo llevado a cabo en esta tesis nuevamente, esta vez con las particularidades que distinguen a un yacimiento naturalmente fracturado del que no lo es.

Referencias Bibliográficas

Ahmadi, R., Shahrabi, J. & Aminshahidy, B. (2017). *Automatic well-testing model diagnosis and parameter estimation using artificial neural networks and design of experiments.* Journal of Petroleum Exploration and Production Technology, Vol. 7, Número 3. 759 – 783. Doi: https://doi.org/10.1007/s13202-016-0293-z

Chang, O., Pan, Y., Dastan, A., Teague, D. & Frank Descant. (2019). *Application of Machine Learning in Transient Surveillance in a Deep-Water Oil Field.* Artículo presentado en SPE Western Regional Meeting, San Jose, California, USA. Doi: https://doi.org/10.2118/195278-MS

Chu, Hongyang., Liao, Xinwei., Dong, Peng., Chen, Zhiming., Zhao, Xiaoliang & Zou, Jiandong (2019). *An Automatic Classification Method of Well Testing Plot Based on Convolutional Neural Network (CNN).* Energies Vol. 12, Número 15. Doi: https;//doi.org/10.3390/en12152846

Daolun LI, Xuliang LIU, Wenshu ZHA, Jinghai YANG & Detang LU. (2020). *Automatic well test interpretation based on convolutional neural network for a radial composite reservoir*. Petroleum Exploration and Development. Volumen 47, Número 3. 623-631. Doi: https://doi.org/10.1016/S1876-3804(20)60079-9.

Gringarten, A. C., Bourdet, D. P., Landel, P. A. & Kniazeff V. J. (1979). *A Comparison Between Different Skin And Wellbore Storage Type-Curves For Early-Time Transient Analysis.* Artículo presentado en SPE Annual Technical Conference and Exhibition, Las Vegas, Nevada. Doi: https://doi-org.pbidi.unam.mx:2443/10.2118/8205-MS

Mueller, J.P. & Massaron, L. (2021). *Machine Learning for dummies.* Segunda edición. John Wiley & Sons, Inc.

Spivey, J.P. & Lee, W. J. (2013). *Applied Well Test Interpretation.* SPE Textbook series vol.13.

Tian, C. & Roland N. H. (2015) *Machine Learning Applied to Multiwell Test Analysis and Flow Rate Reconstruction.* Artículo presentado en SPE Annual Technical Conference and Exhibition, Houston, Texas, USA. Doi: https://doi.org/10.2118/175059-MS

Tian, C. & Roland N. H. (2019). *Applying Machine-Learning Techniques To Interpret Flow-Rate, Pressure, and Temperature Data From Permanent Downhole Gauges.* SPE Res Eval & Eng *22*. 386–401. Doi: https://doi.org/10.2118/174034-PA

Yonggui ,G., Ibrahim, M., Ali ,Z., Yashesh, P., Omar, A.S. & Ahmed, A.S. (2021). *Automated pressure transient analysis: A cloud-based approach.* Journal of Petroleum Science and Engineering, Vol. 196. Doi: https://doi.org/10.1016/j.petrol.2020.107627

Yuxi, H.L. (2020). *Python Machine Learning by Example.* Tercera edición. Packt.

Apéndice A

A continuación, se muestra y explica el código en python que genera las pruebas de presión a partir del trabajo de Gringarten en A comparison between different skin and wellbore storage type-curves for early-time transient analysis.

Primero se importan las librerías de las cuales se hará uso:

```
"""
######### LIBRERIAS #########
"""
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.special as sc
import math as m
from mpmath import *

```

Posteriormente se ingresan los datos que intervienen en la generación sintética de las pruebas presión, y en algunos casos, como el gasto *q* o la porosidad *poro* pueden tener un rango de valores, en esta demostración tienen un rango de 174 a 300 [BPD] y de 0.1 a 0.5 respectivamente. Todos estos valores pueden ser ajustados al problema que se quiera representar.

En la sección de *datos objetivos* se encuentran los rangos de valores para la permeabilidad, daño, y almacenamiento adimensional con el que se entrenó a la red. En *randomizadores* se pueden encontrar el número de puntos que se guardan de cada curva generada, así como la inicialización de los tiempos adimensionales y presión adimensionales para almacenar los datos generados.

```
"""
######### DATOS SECUNDARIOS #########
"""
q = [174,300]                       #[BPD]
B = 1.06                            #[1]
mu = 2.5                            #[cp]
h = 107                             #[ft]
ct = 4.2*10**-6                     #[psi^-1]
rw = 0.29                           #[ft]
r = 0.29                            #[ft]
poro = np.linspace(0.10,0.5,5)      #[1]

SheetCounter = 0

"""
######### DATOS OBJETIVO #########
"""
k = np.linspace(10,100,10)
S = [0,1,2,3,4,4.5,5,6,6.5,7]
Cd = [100,300,500,1000,1500,2000,2500,3000,4000,5000]   #EQ 9

"""
######### RANDOMIZADORES #########
"""
```

```
25.
26. #  .- ADIMENCIONALES
27. NoDeDatos = 100
28.
29. td = np.logspace(1,10,NoDeDatos) #EQ 8
30. p_d = np.zeros(len(td))
31.
```

Una vez que el código cuenta con estos valores puede proseguir con el proceso el que se resulve la ecuación en el espacio de Laplace presentada por Gringarten, pasa los datos al domino real y los recolecta en archivos de Excel.

```
1.  """
2.  ######### SOLUCION LAPLACE INVERSION #########
3.  - Basada en EQ 10 de Gringarten
4.  """
5.  for a in range(len(Cd)):
6.      for s in range(len(S)):
7.          fp = lambda z:(1/z)*((besselk(0,sqrt(z))+S[s]*sqrt(z)*besselk(1,sqrt(z)))
8.                              /(sqrt(z)*besselk(1,sqrt(z))+Cd[a]*z*(besselk(0,sqrt(z))
9.                              +S[s]*sqrt(z)*besselk(1,sqrt(z)))))
10.         for i in range(len(td)):
11.             p_d[i] = invertlaplace(fp,td[i],method='stehfest')
12.         # plt.figure('type curve')
13.         # plt.plot(td/(Cd[a]),p_d)
14.
15.         """
16.         ######### VALORES A DOMINIO DIMENSIONAL #########
17.         """
18.         for b in range(len(k)):
19.             for c in range ((len(poro))):
20.                 for d in range ((len(q))):
21.                     dp_calc = p_d*((141.2*q[d]*B*mu)
22.                                    /(k[b]*h))               #EQ 1 Despejada para DP
23.                     time = td*((poro[c]*mu*ct*rw**2)/(0.000264*k[b]))    #EQ 2
24.                     ruido = np.random.normal(0, .5, len(dp_calc))
25.                     dp_calc = dp_calc+ruido                              #RUIDO
26.
27.                     """
28.                     ######### CALCULO DE LA DERIVADA #########
29.                     """
30.                     ### Diferencias Finitas Centradas ###
31.                     derivada = np.zeros(len(dp_calc)-2)
32.                     for i in range(1,len(dp_calc)-1):
33.                         derivada[i-1] = time[i]*((dp_calc[i+1]-dp_calc[i-1])
34.                                                  /((time[i+1]-time[i-1])))
35.
36.                     N_time = np.delete(time,0)
37.                     N_time = np.delete(N_time,-1)
38.
39.                     ### Diferencias Finitas Hacia Adelante
40.                     # derivada2 = np.zeros(len(dp_calc)-1)
41.                     # for i in range(1,len(dp_calc)-1):
42.                     #     derivada2[i-1] = time[i]*((dp_calc[i+1]-dp_calc[i])
43.                                                   /((time[i+1]-time[i])))
44.
45.                     # N_time2 = np.delete(time,-1)
```

```
                    """
                    ######### RECOLECCIÓN DE DATOS #########
                    """
                    SheetCounter = SheetCounter + 1
                    time=np.delete(time,0)
                    time=np.delete(time,-1)
                    dp_calc=np.delete(dp_calc,0)
                    dp_calc=np.delete(dp_calc,-1)

                    datos=np.column_stack((time,dp_calc,derivada))
                    df = pd.DataFrame(data=datos,
                                      columns=["Tiempo","Presión",
                                               "Derivada"])
                    writer = pd.ExcelWriter(r'C:\Users\toled\OneDrive\\
                                            Escritorio\GuillermoToledo_Tesis\\
                                            Pruebas_10k_3.0\Prueba '
                                            +str(SheetCounter)+'.xlsx',
                                            engine='xlsxwriter')

                    df.to_excel(writer, sheet_name='Prueba ' + str(SheetCounter),
                                index=False)
                    writer.save()

                    targets = np.column_stack((k[b],S[s],Cd[a]))
                    df2 = pd.DataFrame(data=targets,
                                       columns=["permeabilidad";
                                                "Daño",
                                                "Almacenamiento"])
                    writer = pd.ExcelWriter(r'C:\Users\toled\OneDrive\\
                                            Escritorio\GuillermoToledo_Tesis\\
                                            Pruebas_10k_3.0\Targets '
                                            +str(SheetCounter)+'.xlsx',
                                            engine='xlsxwriter')

                    df2.to_excel(writer, sheet_name='Target '
                                 + str(SheetCounter),
                                 index=False)
                    writer.save()
                    print('Prueba: ', SheetCounter)
```

Apéndice B

El siguiente código muestra el procedimiento utilizado para construir, entrenar y guardar la red neuronal utilizada en este trabajo. Primero se muestran las librerías requeridas para este código.

```
import os
import pandas as pd
import tensorflow as tf
from sklearn import preprocessing
from sklearn.metrics import r2_score
import tensorboard
from tensorboard.plugins.hparams import api as hp
import numpy as np
from tensorflow import keras
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Input
import pickle
import math
```

Después se cargan todos los datos de las pruebas generadas y se guardan en una matriz de 2 dimensiones como se describe en el subcapítulo 4.2.

```
"""
############  CARGA DE DATOS  ############
"""

tf.random.set_seed(42)

path = r'C:\Users\toled\OneDrive\Escritorio\GuillermoToledo_Tesis\Pruebas_10k_ruido'
files = os.listdir(path)

Datos = pd.DataFrame()
Holder = pd.DataFrame()
dpH = pd.DataFrame()
tiempoH = pd.DataFrame()
derivadaH = pd.DataFrame()

for file in files:
     if file.startswith('Prueba'):
         Holder = pd.read_excel(r'C:\Users\toled\OneDrive\Escritorio\\
                         GuillermoToledo_Tesis\Pruebas_10k_ruido\\' + file)
         dpH = dpH.append(Holder.Presión, ignore_index = True)
         tiempoH = tiempoH.append(Holder.Tiempo, ignore_index = True)
         derivadaH = derivadaH.append(Holder.Derivada, ignore_index = True)

Targets = pd.DataFrame()
for file in files:
      if file.startswith('Target'):
          Targets = Targets.append(pd.read_excel(r'C:\Users\toled\OneDrive\Escritorio\\
                           GuillermoToledo_Tesis\Pruebas_10k_ruido\\' + file),
                           ignore_index=True)

separacion = math.floor(len(dpH.T)/30)
dp = np.zeros((10000,30))
tiempo = np.zeros((10000,30))
derivada = np.zeros((10000,30))
```

```
35.
36. for i in range(30):
37.     tiempo.T[i] =  tiempoH[i*separacion]
38.     dp.T[i] = dpH[i*separacion]
39.     derivada.T[i] = derivadaH[i*separacion]
```

Una vez que se cuenta con todos los datos, hay que pre procesarlos, se normalizan (además de guardar los valores con los que fueron normalizados para hacer uso de ellos cuando se busque hacer predicciones sobre pruebas totalmente ajenas al conjunto de datos de entrenamiento) y se les divide en un conjunto de datos para el entrenamiento, y otro para la validación y prueba.

```
1.  """
2.  ############  PROCESAMIENTO DE DATOS  ############
3.  """
4.
5.  scalerT = preprocessing.StandardScaler()
6.  scalerDP = preprocessing.StandardScaler()
7.  scalerDERI = preprocessing.StandardScaler()
8.
9.  dp = scalerDP.fit_transform(dp)
10. tiempo = scalerT.fit_transform(tiempo)
11. derivada = scalerDERI.fit_transform(derivada)
12.
13. with open('scalerT3.pickle', 'wb') as f:
14.     pickle.dump(scalerT, f)
15.
16. with open('scalerDP3.pickle', 'wb') as f:
17.     pickle.dump(scalerDP, f)
18.
19. with open('scalerDERI3.pickle', 'wb') as f:
20.     pickle.dump(scalerDERI, f)
21.
22. Datos = np.concatenate((tiempo,dp,derivada),axis=1)
23.
24. num_test_target = 2000
25. num_test_datos = num_test_target
26.
27. X_train = Datos[:-num_test_datos][:]
28.
29. y1 = np.array(Targets.permeabilidad[:-num_test_target])
30. y2 = np.array(Targets.Daño[:-num_test_target])
31. y3 = np.array(Targets.Almacenamiento[:-num_test_target])
32. y_train = (y1, y2 ,y3)
33.
34. X_test = Datos[-num_test_datos:][:]
35.
36. y1 = np.array(Targets.permeabilidad[-num_test_target:])
37. y2 = np.array(Targets.Daño[-num_test_target:])
38. y3 = np.array(Targets.Almacenamiento[-num_test_target:])
39. y_test = (y1,y2,y3)
```

Con esto es posible construir la red neuronal capa por capa con el API de *tensorflow,* la cual una vez disponible se entrena con el conjunto de datos de entrenamiento, y para realizar predicciones con respecto al conjunto de datos de prueba, para finalmente imprimir estas predicciones en conjunto con los valores calculados de R^2.

```
"""
############  CONSTRUCCIÓN DE MODELO  ############
"""
input_layer = Input(shape = (len(X_train.T),))
first_dense = Dense(units=20, activation='relu')(input_layer)
second_dense = Dense(units=8, activation='relu')(first_dense)
third_dense = Dense(units=60, activation= 'relu')(first_dense)
fourth_dense = Dense(units=20, activation= 'relu')(first_dense)
fifth_dense = Dense(units=20, activation= 'relu')(fourth_dense)


y1_output = Dense(units=1, name = 'permeabilidad')(second_dense)
y2_output = Dense(units=1, name = 'Dano')(third_dense)
y3_output = Dense(units=1, name = 'Almacenamiento')(fifth_dense)

model = Model(inputs=input_layer, outputs = [y1_output, y2_output, y3_output])

model.compile(loss={'permeabilidad': 'mean_squared_error', 'Dano': 'mean_squared_error',
              'Almacenamiento':                                    'mean_squared_error'},
optimizer=tf.keras.optimizers.Adam(0.01))

logdir="logs/fit_10k/"
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logdir)

model.fit(X_train, y_train, epochs=300,callbacks=[tensorboard_callback])
predictions = model.predict(X_test)

perm_predict = predictions[0]
Daño_predict = predictions[1]
Alm_predict = predictions[2]

print("Permeabilidad: ", perm_predict)
print("Daño: ", Daño_predict)
print("Almacenamiento: ", Alm_predict)
print('R2 individual de permeabilidad: ',r2_score(y_test[0], perm_predict))
print('R2 individual de Daño: ',r2_score(y_test[1], Daño_predict))
print('R2 individual de Almacenamiento: ',r2_score(y_test[2], Alm_predict))

y_comp = np.vstack((y_test[0],y_test[1],y_test[2]))
y_comp = y_comp.T
predi_comp = np.hstack((predictions[0],predictions[1],predictions[2]))
print("General R2: ",r2_score(y_comp, predi_comp))

model.save(
r'C:\Users\toled\OneDrive\Escritorio\GuillermoToledo_Tesis\FFNN_3.0/')  #MODEL SAVE
```

Adicionalmente si se desea realizar una optimización de hiperparámetros de la red se cuenta con el código para la realización de un *grid search.*

```
"""
############  COMPARACIÓN DE MODELOS  ############
"""
HP_HIDDEN = hp.HParam('hidden_size', hp.Discrete([20, 60, 120]))
HP_HIDDEN2 = hp.HParam('hidden_size2', hp.Discrete([8, 20, 60]))
HP_HIDDEN3 = hp.HParam('hidden_size3', hp.Discrete([8, 20, 60]))
HP_HIDDEN4 = hp.HParam('hidden_size4', hp.Discrete([8, 20, 60]))
HP_HIDDEN5 = hp.HParam('hidden_size5', hp.Discrete([8, 20, 60]))
```

```
HP_EPOCHS = hp.HParam('epochs', hp.Discrete([300,1000]))
HP_LEARNING_RATE = hp.HParam('learning_rate', hp.RealInterval(0.01, 0.4))

def train_test_model(hparams, logdir):
    input_layer = Input(shape = (len(X_train.T),))
    first_dense = Dense(units=hparams[HP_HIDDEN], activation='relu')(input_layer)
    second_dense = Dense(units=hparams[HP_HIDDEN2], activation='relu')(first_dense)
    third_dense = Dense(units=hparams[HP_HIDDEN3], activation='relu')(first_dense)
    fourth_dense = Dense(units=hparams[HP_HIDDEN4], activation= 'relu')(first_dense)
    fifth_dense = Dense(units=hparams[HP_HIDDEN5], activation= 'relu')(fourth_dense)

    y1_output = Dense(units=1, name = 'permeabilidad')(second_dense)
    y2_output = Dense(units=1, name = 'Dano')(third_dense)
    y3_output = Dense(units=1, name = 'Almacenamiento')(fifth_dense)

    model = Model(inputs=input_layer, outputs = [y1_output, y2_output, y3_output])

    model.compile(loss={'permeabilidad': 'mean_squared_error',
                        'Dano': 'mean_squared_error',
                        'Almacenamiento': 'mean_squared_error'},
                  optimizer=tf.keras.optimizers.Adam(
                                hparams[HP_LEARNING_RATE]),
                  metrics={'permeabilidad': 'mean_squared_error',
                           'Dano': 'mean_squared_error',
                           'Almacenamiento': 'mean_squared_error'})

    model.fit(X_train, y_train,
                  validation_data=(X_test, y_test),
                  epochs=hparams[HP_EPOCHS], verbose=False,
                callbacks=[
                    tf.keras.callbacks.TensorBoard(logdir),
                    hp.KerasCallback(logdir, hparams),
                    tf.keras.callbacks.EarlyStopping(
                        monitor='val_loss', min_delta=0,
                          patience=200, verbose=0, mode='auto',
                    )
                ],
                )
    loss, Y1_loss, Y2_loss, Y3_loss, Y1_rmse, Y2_rmse,
    Y3_rmse  = model.evaluate(X_test, y_test)
    pred = model.predict(X_test)
    r2k = r2_score(y_test[0], pred[0])
    r2D = r2_score(y_test[1], pred[1])
    r2A = r2_score(y_test[2], pred[2])
    return Y1_rmse, Y2_rmse, Y3_rmse, r2k, r2D, r2A

def run(hparams, logdir):
      with tf.summary.create_file_writer(logdir).as_default():
          hp.hparams_config(
              hparams=[HP_HIDDEN, HP_HIDDEN2, HP_HIDDEN3, HP_HIDDEN4, HP_HIDDEN5,
                       HP_EPOCHS, HP_LEARNING_RATE],
              metrics=[hp.Metric('mean_squared_error_k', display_name='msek'),
                        hp.Metric('mean_squared_error_D', display_name='mseD'),
                        hp.Metric('mean_squared_error_A', display_name='mseA'),
                       hp.Metric('r2k', display_name='r2k'),
                       hp.Metric('r2D', display_name='r2D'),
                       hp.Metric('r2A', display_name='r2A')],
          )
          Y1_rmse, Y2_rmse, Y3_rmse, r2k, r2D, r2A = train_test_model(hparams, logdir)
```

```
            tf.summary.scalar('msek', Y1_rmse, step=1)
            tf.summary.scalar('mseD', Y2_rmse, step=1)
            tf.summary.scalar('mseA', Y3_rmse, step=1)
            tf.summary.scalar('r2k', r2k, step=1)
            tf.summary.scalar('r2D', r2D, step=1)
            tf.summary.scalar('r2A', r2A, step=1)

session_num = 0
for hidden in HP_HIDDEN.domain.values:
    for hidden2 in HP_HIDDEN2.domain.values:
        for hidden3 in HP_HIDDEN3.domain.values:
            for hidden4 in HP_HIDDEN4.domain.values:
                for hidden5 in HP_HIDDEN5.domain.values:
                    for epochs in HP_EPOCHS.domain.values:
                        for learning_rate in tf.linspace(
                                    HP_LEARNING_RATE.domain.min_value,
                                    HP_LEARNING_RATE.domain.max_value, 3):
                                hparams = {
                                    HP_HIDDEN: hidden,
                                    HP_HIDDEN2: hidden2,
                                    HP_HIDDEN3: hidden3,
                                    HP_HIDDEN4: hidden4,
                                    HP_HIDDEN5: hidden5,
                                    HP_EPOCHS: epochs,
                                    HP_LEARNING_RATE:
                                            float("%.2f"%float(learning_rate)),
                                }
                                run_name = "run-%d" % session_num
                                print('--- Starting trial: %s' % run_name)
                                print({h.name: hparams[h] for h in hparams})
                                run(hparams, 'logs/hparam_tuning_10k_3.0/' + run_name)
                                session_num += 1
```

Este código de un *grid search* genera una carpeta que gurada los valores de R^2 y error cuadratico medio de cada modelo sobre el conjunto de datos de validación para después poder visualizar una comparación de todos ellos con ayuda de *tensorboard,* empleando el comando siguiente directamente en la consola.

```
python -m tensorboard.main --logdir=logs/hparam_tuning/ --port 6006
```

Apéndice C

El código que permite que cualquier prueba ajena al conjunto de datos de entrenamiento o de prueba puede interpretarse por medio de la red neuronal es el que se presenta a continuación. Las librerías que se necesitan son:

```
import pandas as pd
import numpy as np
from tensorflow import keras
from scipy.signal import savgol_filter
import pickle
import math
```

Después se cargan los datos de la prueba a interpretar, se busca dentro de estos mismos datos donde es que comienza la caída (o aumento) de presión dentro de estos mismos datos a la vez que se calcula la diferencia de presión Δp, para finalmente seleccionar los 30 datos más representativos de este periodo de prueba y poder calcular la derivada de la presión con respecto al logaritmo natural de del tiempo.

```
"""
######### PREPROCESAMIENTO DE DATOS ###########
"""

####### LECTURA DE DATOS #########
file = 'Test3.xlsx'
presion_name = 'P(psi)'
tiempo_name = 't(h)'

datos = pd.read_excel(r'C:\Users\toled\OneDrive\Escritorio\GuillermoToledo_Tesis\\' + file)
presion = datos.pop(presion_name)
tiempo = datos.pop (tiempo_name)

col = 0
dp = np.zeros(len(presion)-1)

##### BUSCAR DONDE EMPIEZA LA PRUEBA #######
for i in range (1,len(presion)):
    if (presion[0] - presion[i] < 1):
        col = col + +1
    else:
        dp[i-1] = presion[0] - presion[i]

tiempo = np.array(tiempo[col:])
tiempo = np.delete(tiempo,0)
presion = np.array(presion[col:])
dp = np.array(dp[col:])
derivada = np.zeros(len(presion)-2)

###### NORMALIZACIÓN DE ARREGLOS ######
tiempo_norm = np.zeros(30)
dp_norm = np.zeros (30)
derivada_norm = np.zeros(30)
derivada = np.zeros(len(dp))

separacion = math.floor((len(dp)/30))
```

```
37.
38. for i in range(30):
39.     tiempo_norm[i] =  tiempo[i*separacion]
40.     dp_norm[i] = dp[i*separacion]
41.     derivada_norm[i] = derivada[i*separacion]
42. ###### CALCULO DE LA DERIVADA #######
43. for i in range (1,len(dp_norm)-1):
44.         derivada_norm[i-1] = tiempo_norm[i]*((dp_norm[i+1]-dp_norm[i-1])
45.                              /((tiempo_norm[i+1]-tiempo_norm[i-1])))
```

Por último, se normalizan los datos con los mismos parámetros guardados del código presentado en el apéndice B; con esto se construye una matriz de dos dimensiones que será la que alimentará a la red neuronal para poder realizar una predicción. Se carga la red neuronal guardada en el apéndice B y se imprimen las predicciones.

```
1.  ###### PREPARAR MATRIZ DE ALIMENTACIÓN #########
2.
3.  with open('scalerT.pickle', 'rb') as f:
4.      scalerT = pickle.load(f)
5.
6.  with open('scalerDP.pickle', 'rb') as f:
7.      scalerDP = pickle.load(f)
8.
9.  with open('scalerDERI.pickle', 'rb') as f:
10.     scalerDERI = pickle.load(f)
11.
12. tiempo_norm = scalerT.transform(tiempo_norm.reshape(1,-1))
13.
14. dp_norm = scalerDP.transform(dp_norm.reshape(1,-1))
15.
16. derivada_norm = scalerDERI.transform(derivada_norm.reshape(1,-1))
17.
18. matriz = np.hstack((tiempo_norm,dp_norm,derivada_norm))
19.
20. """
21. ######### Preidccion con modelo ############
22. """
23. #LOAD MODEL
24. loaded_model = keras.models.load_model(r'C:\Users\toled\OneDrive\\
25.                                         Escritorio\GuillermoToledo_Tesis\\
26.                                         FFNN_2.0/')
27.
28. predicciones = loaded_model.predict(matriz)
29.
30. print ('Permeabilidad: ', predicciones[0][0][0])
31. print ('Daño: ', predicciones[1][0][0])
32. print ('CD: ', predicciones[2][0][0])
```