DESARROLLO DE UNA HERRAMIENTA SOFTWARE PARA LA DETERMINACIÓN DE UBICACIONES POTENCIALES DE POZOS PRODUCTORES VERTICALES UTILIZANDO LA SIMULACIÓN NÚMERICA DE YACIMIENTOS.

CARLOS ENRIQUE BARRETO GARAVITO
OSCAR ALBERTO CATAÑO HENAO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOQUÍMICAS
ESCUELA DE INGENIERÍA DE PETRÓLEOS
BUCARAMANGA
2017.

DESARROLLO DE UNA HERRAMIENTA SOFTWARE PARA LA DETERMINACIÓN DE UBICACIONES POTENCIALES DE POZOS PRODUCTORES VERTICALES UTILIZANDO LA SIMULACIÓN NÚMERICA DE YACIMIENTOS.

CARLOS ENRIQUE BARRETO GARAVITO
OSCAR ALBERTO CATAÑO HENAO

Trabajo de grado presentado como requisito para optar por el título
INGENIERO DE PETRÓLEOS

DIRECTOR:
Msc. JHON PINTO CARVAJAL

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOQUÍMICAS
ESCUELA DE INGENIERÍA DE PETRÓLEOS
BUCARAMANGA
2017.

DEDICATORIA

"Con calma y paciencia también se llega, al fin de la meta si uno es constante."

Primeramente, a mi Dios por permitirme terminar una etapa más en mi vida, por sus bendiciones durante el desarrollo de mis estudios, brindándome sabiduría para culminar mis metas, por ser esa luz que me guía en la oscuridad y protegerme en todo momento.

A mis padres, Delys Meza Garavito y Enrique Barreto Pérez, por ser el motor de mi vida y darme su amor incondicional, sus esfuerzos, su entrega y dedicación, por siempre estar a mi lado sin pedir nada a cambio, por ese amor de padres que siempre me han dado y su lucha para que sus hijos salgan adelante.

A mis hermanos, José y Keila, quienes desde la distancia me han brindado todo el apoyo, la confianza y la fuerza necesaria para lograr culminar esta etapa en mi vida.

A mi tía, Ruby, quien no alcanzó a verme graduado, pero a quien dedico este triunfo; que con su partida dejó un gran vacío en mi vida.

A mi tío, Miguel Meza, por ser un gran ejemplo de vida y enseñarme a ser perseverante y responsable.

A mi otra familia, la familia Meza Pulido y todos sus miembros, quienes me acogieron en sus hogares y me brindaron todo el apoyo y confianza para lograr desarrollar mis estudios universitarios.

A mis tíos y primas que siempre estuvieron pendiente de mí, y me brindaron sus consejos para nunca desfallecer en la jornada diaria.

A mi compañero de tesis Oscar Cataño, que confió en mí y fue como un hermano durante los últimos semestres de mi carrera y logramos sacar adelante nuestro proyecto.

Por último, a todos mis grandes amigos que compartieron un tiempo muy importante e inolvidable, que siempre fueron esas personas de mucha alegría, buenos tiempos y malos tiempos, pero siempre me brindaron su amistad, para ser una gran hermandad.

CARLOS ENRIQUE BARRETO GARAVITO

DEDICATORIA

Primeramente, a mi Dios por permitirme terminar una etapa más en mi vida, por sus bendiciones durante el desarrollo de mis estudios, brindándome sabiduría para culminar mis metas, por ser esa luz que me guía en la oscuridad y protegerme en todo momento.

A mi madre Luz Marina Henao por ser el amor de mi vida y darme su amor incondicional, su esfuerzo, su entrega y dedicación, por ser esa persona que siempre estar a mi lado sin pedir nada a cambio, por ese amor de madre que siempre me has dado y su lucha para que sus hijos salgan adelante.

A mis hijos Andrés y Sofia que son lo más importe de mi vida y ser esas personas que me impulsan a culminar mis metas.

A mi novia Yeisi Florez por ser mi apoyo incondicional en las buenas y en las malas, por estar ahí cuando más la necesitaba y ser paciente conmigo y tener ese espíritu de una mujer luchadora y colaboradora con los demás.

A mis tíos y tías que siempre estuvieron pendiente de mí, y me brindaron sus consejos para nunca desfallecer en la jornada diaria.

A mi compañero de tesis Carlos Barreto, que confió en mí y fue como un hermano durante los últimos semestres de mi carrera y logramos sacar adelante nuestro proyecto.

Por último, a todos mis grandes amigos que compartieron un tiempo muy importante e inolvidable, que siempre fueron esas personas de mucha alegría, buenos tiempos y malos tiempos, pero siempre me brindaron su amistad, para ser una gran hermandad.

OSCAR ALBERTO CATAÑO HENAO

AGRADECIMIENTOS

A la Universidad Industrial de Santander, nuestra alma mater, por recibirnos, formarnos profesionalmente y plantar en nosotros los valores e ideas que nos permitan aportar un grano de arena al desarrollo y crecimiento de la sociedad.

A la Escuela de Ingeniería de Petróleos y a cada uno de nuestros profesores, quienes, de principio a fin formaron parte fundamental de nuestro crecimiento académico y personal.

A nuestro director, el Ingeniero M.Sc. John Pinto Carvajal, quien depositó su confianza y nos guio con su experiencia en cada etapa de este proceso. A nuestro amigo Héctor Vargas, quien nos asesoró poniendo a nuestra disposición todo su conocimiento con humildad, paciencia y cariño.

CONTENIDO.

Pág.

LISTA DE TABLA.

Pág.

LISTA DE FIGURAS.

Pág.

LISTA DE ANEXOS

Pág.

RESUMEN.

TÍTULO: DESARROLLO DE UNA HERRAMIENTA SOFWARE PARA LA DETERMINACIÓN DE UBICACIONES POTENCIALES DE POZOS PRODUCTORES VERTICALES UTILIZANDO LA SIMULACIÓN NÚMERICA DE YACIMIENTOS.*
AUTORES: CARLOS BARRETO GARAVITO
OSCAR ALBERTO CATAÑO HENAO**

PALABRAS CLAVE: Matlab, Valor Presente Neto, Metodología, Algoritmos, modelo de simulación numérica, Ubicación de Pozos.

DESCRIPCIÓN:

El objetivo principal de este trabajo de investigación es desarrollar una metodología para la implementación de una estrategia de ubicación de pozos productores en un enmallado de simulación numérica representativo de un campo petrolero. En primera instancia se describe el soporte teórico de las diferentes herramientas y métodos desarrollados para el diseño de estrategias de ubicación de pozos en campos de petróleo.

Posteriormente, se describe el modelo conceptual de simulación utilizado en el presente trabajo, sus propiedades PVT, petrofísica, interacción roca fluido, etc. Una vez establecido lo anterior, se establecen a través de juicio ingenieril las diferentes variables predominantes en el desarrollo de la función objetivo-establecida. Posteriormente se aplica la metodología propuesta haciendo uso del simulador numérico CMG y la herramienta de cálculo Matlab. En primera instancia se definieron los cuatro principales parámetros de evaluación para la función objetivo: Porosidad, Permeabilidad Efectiva al Aceite, Saturación de Aceite y el Net To Gross.

Una vez aplicada la metodología establecida se realiza un análisis cualitativo de los resultados obtenidos, para así evaluar la eficiencia de operación de la metodología hibrida aplicada. Se logró observar que con la aplicación de esta metodología es posible evaluar rápida y correctamente la función objetivo-propuesta, Valor Presente Neto y finalmente permitir establecer el desarrollo de la estrategia de ubicación de 15 (Quince) pozos productores en el modelo base de simulación numérica.

* Trabajo de Grado.
** Facultad de Ingenierías Fisicoquímicas. Escuela de Ingeniería de Petróleos Director M.Sc. John Pinto Carvajal.

ABSTRACT.

TITLE: DEVELOPMENT OF A SOFTWARE TOOL FOR THE DETERMINATION OF POTENTIAL LOCATIONS OF VERTICAL PRODUCTION WELL USING THE NUMERICAL SIMULATION OF PLACES.*

AUTHORS: CARLOS BARRETO GARAVITO.
OSCAR ALBERTO CATAÑO HENAO.**

KEY WORDS: Matlab, Net Present Value, Methodology, Algorithms, Numerical Simulation Model, Well Location.

DESCRIPTION:

The main objective of this research work is to develop a methodology for the implementation of a strategy of locating producing wells in a numerical simulation mesh representative of an oil field. In the first instance, the theoretical support of the different tools and methods developed for the design of well location strategies in oil fields is described.

Afterwards, the simulation conceptual model used in the present work is described, its PVT properties, petrophysics, fluid rock iteration, etc. Once the above is established, the different variables predominant in the performance of the established objective function are established through engineering judgment. Subsequently, the proposed methodology is applied using the CMG numerical simulator and the Matlab calculation tool. In the first instance, the four main evaluation parameters for the objective function were defined: Porosity, Effective Permeability to Oil, Oil Saturation and Net to Gross.

Once the established methodology is applied, a qualitative analysis of the results is performed, to evaluate the efficiency of operation of the applied hybrid methodology. It was observed that with the application of this methodology it is possible to quickly and correctly evaluate the proposed objective function, Net Present Value, in order to establish the development of the location strategy of 10 (ten) producing wells in the simulation base model Numerical value.

* Undergrate Project.
** Faculty of Physic-chemical Engineering. School of Engineering of Petroleum Director M.Sc. John Pinto Carvajal.

INTRODUCCIÓN.

Uno de los principales focos de investigación de la industria petrolera actualmente es el óptimo desarrollo de sus reservas, obteniendo un deseable aumento en sus reservas de crudos ya existentes y desarrollados. La realización de determinada tarea no es de fácil manejo y desarrollo para atribuirla de manera singular al juicio empírico de un determinado equipo de trabajo. Por tal motivo la atención es centrada en desarrollar mecanismos y aplicaciones que conjugadas con herramientas tecnológicas generen un mejor y ágil desempeño en el momento de su aplicación en el campo.

Uno de los principales enfoques es la hibridación de programas que nos ayuden a realizar cálculos de unas formas más ágiles, correctas y eficientes. El desarrollo de estas técnicas conjugadas impera el discernimiento de cada uno de los programas involucrados en el desarrollo de determinada técnica. Uno de los factores más importantes en el desarrollo de estas técnicas es generar una alta compatibilidad entre los datos manejados por cada uno de los software o programas empleados, de esta forma lograr desarrollar la técnica conjugada en lenguajes similares. El desarrollo de estas técnicas, metodologías y procesos tienen como objetivo principal una disminución de la carga computacional manejada por simuladores comerciales utilizados convencionalmente dentro de la industria petrolera en el área de simulación numérica de yacimientos, a partir de la implementación de otros programas que realicen un manejo computacional más rápido y correcto de determinadas variables o parámetros involucrados en la evaluación de una función objetivo en particular.

En el presente trabajo se desarrolla una herramienta software que permite la aplicación de una estrategia de ubicación de pozos productores en un enmallado de simulación numérica a través de la conjugación de un simulador numérico comercial y cálculos aritméticos desarrollados en otra herramienta informática (Matlab).

En primera instancia se realiza una revisión bibliográfica donde se exponen las diferentes técnicas, métodos y herramientas utilizadas para una deseable ubicación de pozos en el desarrollo de un campo de petróleo. Posteriormente se diseña el modelo de simulación representativo de un campo de aplicación y a partir de este punto se desarrolla la metodología propuesta, haciendo uso de herramientas computacionales como: simulador numérico y Matlab. Finalmente, con los resultados arrojados por el desarrollo de la metodología se realiza el respectivo análisis de los resultados y la efectividad de la conjugación de las herramientas computacionales.

Lograr culminar la construcción de una metodología hibrida requiere trabajo, inicialmente se hace necesario establecer los parámetros o variables que se van a emplear para alcanzar tal fin. Cabe resaltar que no es necesario el ajuste de todas las variables, y es acá donde juegan un papel muy importante tanto los criterios ingenieriles para seleccionar las más adecuadas, como también los métodos de muestreo que permiten una selección óptima de las variables.

Finalmente, la conjugación de las herramientas tales como simuladores numéricos convencionales y lenguajes de programación basados en cálculos numéricos, nos ayudan a la culminación del desarrollo de un modelo hibrido que tiene como objetivo principal dar a conocer las ubicaciones con mayor potencial productivo para pozos productores dentro de un enmallado base de simulación, para posteriormente ser ubicados, evaluados y desarrollados dentro de un periodo de tiempo establecido.

1. REVISION BIBLIOGRAFICA DE METODOLOGIAS EMPLEADAS PARA EL DESARROLLO DE ESTRATEGIAS EN UBICACIÓN DE POZOS EN CAMPOS PETROLEROS.

La optimización de la ubicación de pozos tiene el potencial de convertirse en un procedimiento importante dentro de la planificación del desarrollo de campo. El objetivo principal de un esfuerzo de optimización de ubicación de pozos es proporcionar a los encargados de la toma de decisiones un asesoramiento de alta calidad sobre dónde colocar los pozos. Para lograr esta tarea, se desarrolla una metodología que busca ubicaciones de pozos que mejoran en gran medida el rendimiento económico relevante o maximizar la recuperación esperada del activo de hidrocarburos. Sin embargo, el problema de la ubicación de pozos es desafiante porque la relación entre la ubicación de pozos, la geología del yacimiento y los patrones de flujo de fluido resultantes es, pero en los casos más triviales, complejos y por lo tanto el desarrollo de la metodología necesita confiar en procesos que involucran simulaciones de yacimientos computacionalmente costosas para encontrar los volúmenes finales de producción asociados con una configuración dada de pozos. Además, la eficiencia del esfuerzo de optimización, la calidad de los consejos proporcionados se basa en las limitaciones realistas que se han introducido en el problema de optimización. En particular, es importante que las limitaciones de diseño con las que el equipo de desarrollo de campo funcione, explícita o implícitamente, se articulen y formalicen en la búsqueda de la ubicación óptima del pozo.

Con el objetivo de desarrollar metodologías que nos ayuden a obtener de manera más eficiente una adecuada ubicación del grupo de pozos que conlleve a una óptima solución de la función objetivo propuesta particularmente para determinados casos, es necesario utilizar técnicas de optimización heurística, igualmente también es posible el uso de métodos para construir funciones de modelos proxy para ayudar a la optimización y herramientas de evaluación de incertidumbre.

Dentro del grupo de las técnicas de optimización heurísticas se encuentran, por ejemplo, Simulación Análoga, Enjambre de Partículas, Algoritmos Evolutivos. En el caso de los modelos proxy se encuentran técnicas como: Regresión Polinómica, Modelos Kriging, Modelos Spline y Modelos de Neural Networks.

1.1 TÉCNICAS UTILIZADAS EN LA UBICACIÓN DE POZOS.

Con el fin de resolver problemas complicados con eficiencia, en ocasiones es necesario emplear algunos métodos de optimización y construir una metodología de control que no garantice encontrar la mejor respuesta pero que casi siempre encuentre una buena solución. De esta forma, surgen los diferentes métodos heurísticos descritos a continuación.

1.1.1 Simulated annealing: Es un algoritmo inicialmente desarrollado para la solución de problemas de optimización combinada. El tipo de problema típicamente considerado implica encontrar el orden óptimo de un sistema con un gran número de componentes. Una ordenación óptima es una que minimiza una cierta función global o una función objetivo. En el contexto del modelado del yacimiento estocástico, los componentes podrían ser valores de atributos del yacimiento como porosidad, permeabilidad y saturación definidas para bloques de determinado tamaño constante.

La función de evaluación u objetivo podría ser una medida de lo cerca que está el ordenamiento (disposición espacial de los valores de porosidad del bloque) que reproduce el patrón de correlación espacial (variograma) deducido de un estudio de afloramiento. Encontrar un orden óptimo equivale a generar un modelo numérico.

El reciente interés en el uso de la técnica de Simulated Annealing para la caracterización del yacimiento capitaliza dos nuevas ideas. En primer lugar, el problema de imagen se configura como un problema de optimización.

En segundo lugar, el problema de optimización se resuelve con Simulated Annealing. Este formalismo permite contabilizar diversos tipos de información mediante la construcción de funciones objetivos más complejas que la simple identificación de un modelo de variograma.

La selección del método de Simulated Annealing para resolver este problema de optimización se debe principalmente a la robustez del método y a la simplicidad para su implementación. El método de Simulated Annealing como un algoritmo de optimización fue descrito por primera vez por *Kirpatrick* et al. Se han encontrado muchas aplicaciones exitosas para problemas diferentes y complejos. Las aplicaciones a caracterizaciones de yacimientos y problemas de ingeniería han sido realizadas por *Zhou et al, Quenes et al. y Saad et al.*

B.L. Beckner et al. [1](1995) desarrollaron un programa de algoritmo de optimización alrededor del simulador Mobilis PEGASUS. El programa de optimización tiene tres componentes principales: una parte del controlador para analizar la entrada y salida de la simulación, un módulo de economía para calcular el Valor Actual Neto (VAN) y una parte del algoritmo para la iteración continua. Todo el proceso de iteración es automático sin ninguna entrada manual una vez que se inicia.

Utilizando el método Simulated Annealing como un motor de optimización inconcebible con un paquete de análisis económico y el simulador de depósito PEGASUS, ha producido una tecnología para optimizar el valor actual neto de un desarrollo de campo completo variando la ubicación y secuencia de los pozos de producción. El enfoque es automatizado y puede manejar una variedad de restricciones sobre la colocación de pozos y/o la programación de pozos.

[1] B.L. BECKNER *et al.* Field Development Planning Using Simulated Annealing - Optimal Economic Well Scheduling and Placement (1995)

Se investigaron dos casos diferentes de optimización. Estos casos iban desde la optimización de un reservorio uniforme, el desarrollo uniforme de costos de pozo hasta un caso que optimiza el desarrollo de un reservorio con permeabilidad variable, presiones iniciales variables y costos de pozos variables. En el primer ejemplo, se aplica a la variable permeabilidad, variable Presión Fondo Fluyendo (Pwf), el escenario de costos de pozos variables. El desarrollo razonable, sin aplicación del algoritmo, tuvo un proyecto en el Valor Actual Neto de 144.27 millones de dólares con una producción acumulada de 33.503 MMSTB mientras que en la aplicación método Simulated Annealing produjo un Valor Actual Neto del proyecto de 168.75 millones de dólares y una producción acumulada de petróleo de 36.818 MMSTB.

El segundo caso de optimización es comparar el Valor Actual Neto de la ubicación optimizada de pozos para la permeabilidad variable, la presión inicial variable y el caso uniforme de costo de pozo a una corrida utilizando la misma colocación y programación de pozos, pero haciendo que los costos de los pozos sean variables. El Valor Actual Neto del proyecto para el desarrollo razonable, no optimizado y heterogéneo del yacimiento es de 138,19 millones de dólares con una producción acumulada de petróleo de 33,024 MMSTB. Una vez más, el desarrollo optimizado produjo un Valor Actual Neto del proyecto de 168.75 millones de dólares y 36.818 MMSTB de aceite acumulado.

1.1.2 Algoritmo Particle Swarm: Es un Algoritmo Basado en Cúmulos de Partículas o Particle Swarm Optimization es una técnica metaheurística basada en poblaciones e inspirada en el comportamiento social del vuelo de las bandadas de aves o el movimiento de los bancos de peces. Particle Swarm Optimization (PSO) fue originalmente desarrollado por el psicólogo-sociólogo Jammes Kennedy y por el ingeniero electrónico Russell Eberhart en 1995. El desarrollo de Particle Swarm Optimization (PSO) tiene muchas similitudes con las técnicas de computación evolutiva, tales como los algoritmos genéticos (GA). El sistema se inicializa con una

población de soluciones y las búsquedas al azar para una solución óptima mediante la actualización de las generaciones. Sin embargo, a diferencia de un algoritmo genético (GA), Particle Swarm Optimization (PSO) no tiene operadores tales como la evolución crucen y mutación. En Particle Swarm Optimization (PSO), las soluciones posibles, llamados partículas, vuelan por el espacio del problema siguiendo las partículas óptimas actuales.

Cada partícula realiza un seguimiento de sus coordenadas en el espacio del problema que se asocian con la mejor solución (fitness) que ha logrado hasta el momento. (También se almacena el valor de aptitud.) Este valor se llama *pbest*. Otro "mejor" valor que se realiza un seguimiento por el optimizador de enjambre de partículas es el mejor valor, obtenido hasta el momento por cualquier partícula en los vecinos de la partícula. Este lugar se llama *lbest*. Cuando una partícula lleva toda la población como sus vecinos, el mejor valor es un mejor mundial y se llama *gbest*.

Mansoureh Jesmani and Mathias C. Bellout et al.[2] (2015). Aplicaron el desarrollo del algoritmo de enjambre de partículas a diferentes casos de estudios en la ubicación de pozos en campos de petróleo. El algoritmo PSO aplicado en este trabajo es un optimizador de derivadas libres basado en un procedimiento de búsqueda estocástico. El procedimiento consiste en un modelo del comportamiento de un enjambre. Incluye varios factores probabilísticos, e imita el movimiento colectivo de animales, por ejemplo, el vuelo de una bandada de aves. Incorporamos las diferentes restricciones de ubicación de pozos en el algoritmo PSO utilizando dos técnicas diferentes de manejo de restricciones: un procedimiento de decodificador y el método de penalización. El procedimiento del decodificador mapea el espacio de búsqueda factible en cubos n-dimensionales y tiene la ventaja de no requerir ajuste de parámetros. El método de penalización convierte el problema de

[2] MANSOUREH Jesmani And MATHIAS C. Bellout et al. *Particle Swarm Optimization Algorithm for Optimun well placement subject to Realistic Field Development constrains.* SPE 175590.2015

optimización restringido en uno no restringido introduciendo un término adicional, denominado función de penalización, a la función objetivo.

Estas técnicas de manejo de restricciones se aplican a dos casos diferentes de producción de yacimientos. El primer caso es un problema sencillo, donde el algoritmo PSO se implementa con una restricción de cota y una longitud pozo en particular para encontrar la ubicación óptima de un productor horizontal. El segundo caso trata un modelo de campo sintético con datos de porosidad y permeabilidad realista y geometría rectangular. Para este caso, utilizando la implementación del decodificador del algoritmo PSO, optimizaron la colocación de ocho pozos verticales que se colocaron dentro de regiones del yacimiento de forma irregular determinadas basándose en consideraciones geológicas, por ejemplo, fallas. En términos de rendimiento económico para este caso, el PSO con decodificador genera una mejora del 16% en comparación con la configuración inicial del pozo.

El segundo caso de aplicación es un modelo de un campo de aceite y gas, llamado Norme, ubicado en el mar noruego, tiene unas reservas de aceite y gas iguales a 73.9 $x\ 10^6\ \ STB$ y 13.6$x\ 10^9\ SCF$, respectivamente, está compuesto por 14 capas de 40 X 64 bloques en los que están activos 27.755 bloques. Se optimizan las ubicaciones de 8 pozos verticales, 5 productores y 3 inyectores. Los pozos de producción e inyección se perforan a través de todas las capas. Se optimizan dos variables por pozo para encontrar las coordenadas i y j del pozo. Por lo tanto, hay 16 variables de optimización. Los productores son controlados por BHP (Bottom Hole Pressure) 3626 Psi constante, con el límite superior de la tasa de aceite igual a 31450 STB/día. Las tasas de inyección se fijan en 62898 STB/día y el límite superior del BHP para los inyectores es igual a 4641 Psi. La proyección del tiempo de producción se establece en 3 años.

Cuatro ejecuciones de optimización se realizaron debido a la naturaleza estocástica del algoritmo PSO. La mejor optimización genera un incremento del Valor Presente Neto (VPN) del 16,3%. Para este caso, todas las ejecuciones de optimización convergen a una configuración final después de aproximadamente 750

simulaciones, aunque el algoritmo PSO continúa su búsqueda hasta alcanzar el máximo número de evaluaciones de la función objetivo (en este caso 2450). El decodificador mantiene todas las partículas en la región factible del espacio de búsqueda. Un trabajo adicional puede incluir una comprobación de convergencia para asegurar que se detenga una ejecución de optimización una vez que no se observa ninguna mejora adicional en la función de coste.

1.1.3 Algoritmos Evolutivos: Algoritmos Genéticos es una técnica de búsqueda estocástica y heurística basada en la teoría de la evolución natural y la selección. La idea básica gira en torno a la supervivencia de los más aptos y las soluciones se desarrollan a través del apareamiento (intercambio de información) de las mejores soluciones. Se permite una alternación ocasional de las soluciones de ajuste para explorar otras partes del espacio de búsqueda o para evitar el atrapamiento en optima solución local (Mitchell, 1996). El uso de algoritmos genéticos para el problema de optimización de ubicación de pozos se ha encontrado ideal debido a las siguientes razones:

- El algoritmo puede ser fácilmente paralelizado porque cada uno de los individuos puede ser evaluado por separado.
- La búsqueda del óptimo está orientada a encontrar el óptimo global en lugar de
- Optima local.
- Se desempeñan bien en problemas donde la función de acondicionamiento físico es compleja, discontinua, ruidosa, cambia con el tiempo, o tiene muchos optima locales (Holland, 1992).
- El algoritmo es capaz de manipular muchos parámetros de forma simultánea.
- No se requieren gradientes durante el proceso de optimización.
- Dado que la población inicial se compone de múltiples soluciones en lugar de una sola, tenemos la oportunidad de explorar más el espacio de búsqueda en cada generación.
- El algoritmo puede ser mejorado y combinado con otras técnicas.

Dos tipos de algoritmos genéticos se utilizan en problemas de optimización: Algoritmo Genético Binario (bGa) y el Algoritmo Genético Continuo (cGA). En Algoritmo Genético Binario (bGa), el proceso de optimización incorpora la codificación del valor de cada variable a su valor binario correspondiente, aplicando los operadores de Algoritmos Genéticos (GA) al cromosoma, obteniendo los descendientes resultantes y reubicándolos en el espacio real.

Por el contrario, los Algoritmos Genéticos Continuos (cGA) usan números de valor real directamente. Además de las ventajas de algoritmos genéticos mencionadas anteriormente, los Algoritmos Genéticos Continuos (cGA) en particular son más atractivos para usar en las ubicaciones óptimas de pozos en las estrategias de desarrollo de campos petroleros por las siguientes razones:

- El individuo puede asumir cualquier valor en el dominio de búsqueda proporcionando mayor resolución en comparación con el bGA discreto.
- Es más fácil reforzar la adhesión variable a los límites del problema en el cGA.
- El proceso de codificación/decodificación de variables en los algoritmos genéticos binarios introduce deficiencias de traducción que se pueden prevenir en los algoritmos genéticos continuos. Un problema común se encuentra cuando una transición deseada entre dos valores adyacentes da lugar a la alteración de muchos bits binarios en ciertos parámetros. En otros casos, la alteración de un bit puede causar un cambio dramático en el valor de otras propiedades (Deb y Agrawal, 1995).

En la literatura se han discutido diversas investigaciones relacionadas con la implementación y ejecución de los algoritmos genéticos junto con la simulación numérica de yacimientos como una alternativa de solución para lograr una aceptable optimización de la ubicación de pozos en el desarrollo de campos petroleros.

A continuación, hacemos referencia a algunas de esas investigaciones.

Morales A. et al[3] (2011). Presentaron un nuevo algoritmo genético modificado (GA) para una optimización de ubicación de pozos bajo incertidumbre geológica. Las entradas del algoritmo son posibles modelos geológicos, y el nivel de riesgo que el usuario puede aceptar. En este algoritmo, se modifica el clásico algoritmo genético para que: 1) en la evaluación del valor de aptitud (producción acumulada o el valor actual neto) del individuo cada uno (así localización) todos los posibles modelos geológicos facilitados por los usuarios son evaluados, 2) sobre la convergencia de las el algoritmo, la salida es una ubicación óptima así del más fuerte individual, teniendo en cuenta todos los modelos geológicos y 3) esta ubicación óptima así se selecciona basándose en el nivel definido por el usuario de entrada de riesgo.

La mayoría de los algoritmos actuales disponibles en la bibliografía no permiten al usuario introducir el factor de riesgo y los pesos individuales deseados para cada realización. El algoritmo de riesgos limitados es posible proporcionar diferentes ubicaciones óptimas para los pozos, dependiendo del nivel de riesgo que el usuario quiere tomar. Todas estas características hacen que el nuevo algoritmo mucho más eficientes que los actuales algoritmos aplicados y bien de colocación en la literatura para el manejo de la incertidumbre geológica. Presentaron la aplicación del algoritmo de riesgos limitados a una optimización de la ubicación del pozo horizontal en un yacimiento de gas condensado, Campo Norte de Qatar, con múltiples campos de permeabilidad posibles y con diferentes factores de riesgo definidos por el usuario.

Emeric et al. [4] (2009) implementaron una herramienta de optimización basada en Algoritmos Genéticos para optimizar el número, ubicación y trayectoria de un número de pozos productores e inyectores desviados. Propusieron un método para manejar soluciones inviables creando una población de referencia que consistía

[3] MORALES A. et al. *A New Modified Algorithm for Well Placement optimization under Geological uncertainties.* SPE 143617. 2011

[4] EMERIC et al. *Well Placement Optimization Using a Genetic Algorithm with Nonlinear Constraints.* SPE *118808*.2009

sólo en soluciones completamente viables. Cualquier solución inviable encontrada en la optimización fue reparada aplicando el cruce entre ella y un individuo de la población de referencia hasta que se obtuvo una nueva solución factible. Aplicaron esta técnica en tres modelos de simulación de campo completo basados en casos reales usando dos estrategias diferentes: la primera con toda la población inicial definida aleatoriamente; Y el segundo incluyendo la propuesta de un ingeniero en la población inicial. Se observaron mejores resultados en la segunda estrategia y las soluciones fueron más intuitivas para el caso probado. También sugirieron y probaron un enfoque de optimización alternativa optimizando el tipo de pozo y el número de la propuesta de un ingeniero. Aunque los resultados finales no fueron tan buenos como la optimización completa, concluyeron que este enfoque puede utilizarse cuando hay limitación de tiempo para realizar la optimización completa en casos complejos.

Farshi et al.[5] (2008) convirtió un marco de optimización de posicionamiento y diseño de pozos que fue desarrollado por Yeten et al. (2002) de Algoritmo Genético Binario (bGA) a un Algoritmo Genético Continuo (cGA) de valor real. Que el cGA proporciona mejores resultados en comparación con el rendimiento de bGA en los mismos modelos sintéticos. Además, implementó varias mejoras en el proceso de optimización como la imposición de una distancia mínima entre los pozos y el modelado de pozos curvos.

Yeten et al.[6] (2002) aplicaron un Algoritmo Genético Binario para optimizar el tipo de pozo, localización y trayectoria para pozos no convencionales. Junto con eso, desarrollaron una herramienta de optimización basada en un algoritmo de gradiente conjugado no lineal para optimizar los controles operacionales de los pozos. También se implementaron varias funciones auxiliares incluyendo el Hill Climber

[5] FARSHI, M. *Improving Genetic Algorithms for Optimum Well Placement*, Master's Report, Department of Energy Resources Engineering, Stanford University, California. 2008
[6] YETEN, B., DURLOFSKY, L. AND AZIZ, K. *Optimization of Nonconventional Well Type, Location and Trajectory, paper* SPE 7756.2002

(HC). Además, se aplicaron cerca del pozo, lo que explica aproximadamente los efectos de la heterogeneidad en el flujo que se produce en la región cercana al pozo mediante el cálculo de un factor de Skin para cada segmento de pozo. Los resultados de este estudio se presentaron en modelos sintéticos fluvial y en capas, así como un modelo de sección de un campo de Arabia Saudita. Se introdujo una metodología de diseño experimental para cuantificar los efectos de la incertidumbre durante la optimización. El estudio también llevó a cabo análisis de sensibilidad.

Montes et al. (2001) optimizaron la colocación de pozos verticales usando un algoritmo genético sin ninguna hibridación. Intentaron discernir los efectos de los parámetros internos del algoritmo genético, como la probabilidad de mutación, el tamaño de la población, la semilla inicial y el uso del elitismo. Sus pruebas se aplicaron en dos modelos rectangulares sintéticos (un modelo de homogéneo y un modelo altamente heterogéneo). Para los casos probados, encontraron que la tasa de mutación ideal debería ser variable con la generación. El uso de semillas aleatorias para su problema mostró poca sensibilidad mientras que el uso del elitismo mostró una mejoría significativa. El estudio de tamaño de población que realizaron sugirió que un tamaño apropiado era igual al número de variables en el problema. Cuando se utilizaron poblaciones muy grandes, la convergencia de la solución se disuadió como más cromosomas de mala calidad tuvo que ser evaluado. También llamaron la atención sobre cuestiones como la convergencia absoluta y la estabilidad del algoritmo de optimización.

1.1.4 Método de Gradiente Base: El método de Handels et al. (2007) fue el primer intento de utilizar un algoritmo de optimización basado en gradientes para la ubicación óptima de pozos, donde se conoce como ubicaciones óptimas de pozos aquellas que maximizan el Valor Actual Neto (VAN) durante un tiempo total de producción específico.

Aunque este método busca encontrar las direcciones discretas (i, j) de la ubicación óptima del pozo para problemas bidimensionales, los pozos se trasladan del gridblock del simulador al gridblock del simulador basado en un cálculo del gradiente.

Handels et al. [7](2007); *Zandvliet et al. (2008), Sarma y Chen (2008)*.[8]Ubicaron pseudo-pozos en cuadriculas alrededor de cada pozo real. En lugar de utilizar una tasa muy pequeña para los pseudo-pozos, Sarma y Chen (2008) distribuyeron la tasa de pozos entre el pozo real y los pseudo-pozos asociados de acuerdo con la distancia desde cada pseudo-pozo al pozo real ubicada en la posición X_w, Y_w; es importante señalar que, en este enfoque, la ubicación real del pozo (X_w, Y_w) no está necesariamente en el centro del gridblock. Durante la optimización, las ubicaciones reales de los pozos (X_w, Y_w) son tratadas como los parámetros a estimar y la tasa en cada pozo real y sus pseudo-pozos son funciones explícitas de las coordenadas aréales, X_w, Y_w, de su ubicación.

Esta metodología permitió calcular el gradiente del Valor Actual Neto con respecto a X_w, Y_w. A diferencia del método de Handels et al. (2007) y Zandvliet et al. (2008), los parámetros $X_w \; y \; Y_w$son continuos, por lo tanto, el método es un Algoritmo de optimización de posicionamiento de pozo basado en gradiente verdadero. Las dos ventajas potenciales de la metodología de Sarma y Chen (2008) Sobre el Handels et al. (2007). Son: (1) la dirección de búsqueda no está limitada a las ocho direcciones definidas por el ocho segmentos de línea que conectan el centro del gridblock de ubicación del pozo real con los centros de los gridblocks que contienen pseudo-pozos y (2) el tamaño del paso en la dirección del gradiente no está limitado arbitrariamente por las dimensiones de un bloque de rejilla de modo que es posible

[7] HANDELS et al. *Efficient Well Placement Optimization with Gradient-Based Algorithms and Adjoint Models.* SPE 112257.2007

[8] ZANDVLIET et al. SARMA Y CHEN. *Adjoint-Based Well-Placement Optimization under Production Constraints. SPE 105797.2008*

que el pozo pueda ser movido de su gridblock correcto a un gridblock lejos de su ubicación actual.

Wang et al. (2007) introdujo un procedimiento basado en gradiente diferente para optimizar las ubicaciones de los pozos de inyección de agua. La idea de afirmación fue inicializar el procedimiento de optimización ubicando un pozo de inyección en cada gridblock que no contenga un pozo.

Además, agregaron un término a la función Valor Presente Neto (VPN) convencional para tener en cuenta el costo de perforar cada pozo. Pues las tasas eran ajustadas para maximizar el Valor Actual Neto a lo largo de la vida útil del yacimiento. Si la tasa de uno de los pozos iniciales se convierte en cero, el pozo se eliminaba del sistema y el costo de perforar ese pozo se elimina de la función Valor Presente Neto.

Como el problema requiere que las tasas de algunos pozos vayan a cero, Wang et al. (2007) aplicaron el algoritmo de ascenso más pronunciado con esta limitación, de modo que al menos un pozo se elimina en cada iteración que es computacionalmente ineficiente cuando el número inicial de pozos era grande. Zhang et al. (2010) mejoraron significativamente la eficiencia computacional del procedimiento de optimización usando un método de gradiente de proyección. Con este método, a iteraciones tempranas, a menudo es posible eliminar varios pozos durante una iteración única. Las técnicas utilizadas en Handels et al. (2007) y Zandvliet et al. (2008) reducirá ocasionalmente el número de pozos moviendo dos o más pozos en el mismo gridblock, en cuyo caso todos los pozos se combinan y se tratan como una solo.
El método de *Emerick et al.* (2009) puede a veces hacer una pequeña disminución en el número de pozos. La mayoría de otros métodos en la literatura requieren que el número de pozos se especifique a priori. Una ventaja del procedimiento de Wang et al. (2007) y Zhang et al. (2010), así como el algoritmo básico presentado por

Fahim F. *et al* (2010) es que permiten intentar la optimización de: (a) el número de pozos, (b) la ubicación de pozos, y (c) las tasas de pozos, simultáneamente.

Fahim F. et al. [9](2010) realizó una mejora a la metodología básica introducida en Wang et al. (2007) y Zhang et al. (2010) para que pueda ser aplicada a problemas más realistas, es decir, tridimensionales de flujo trifásico, donde el objetivo es optimizar no simplemente la ubicación de los pozos de inyección de agua, sino también la ubicación de los productores y los inyectores. Además, a diferencia del trabajo de Wang et al. (2007) y Zhang et al. (2010), estableció restricciones vinculadas a las presiones del pozo porque en realidad una inyección de agua el pozo inyector siempre está limitado a la presión máxima en el fondo del pozo (BHP) y los pozos productores siempre tienen una restricción de presión mínima en el fondo del pozo (BHP).

Dado que las restricciones de presión superior e inferior del fondo de pozo representan restricciones no lineales cuando nuestros parámetros o bien controles son tasas de flujo, la adición de limitaciones vinculadas a las presiones de fondo de pozo añade una complicación significativa para el problema la optimización.

Como el simulador comercial utilizado (Eclipse 300) no proporciona el gradiente de estas limitaciones no lineales Fahim F. et al (2010). Desarrollaron un método novedoso para asegurar la satisfacción de estas restricciones de presión de fondo no lineal para el control de los pozos. La contribución final por parte de Fahim F. et al. Para el problema óptimo de ubicación de pozos es la introducción de una etapa inicial donde se determinan las tasas óptimas de inyección y producción totales para determinado tiempo de producción de yacimiento.

[9] FAHIM F. *et al. A Two-Stage Well Placement Optimization Method Based on Adjoint Gradient. SPE 135304.*2010

En la mayoría de los algoritmos de optimización de ubicación de pozos presentados en la literatura, los controles de pozos y la vida esperada del yacimiento ambos son especificados previamente y son dos factores fijos durante la optimización, lo que resulta en una solución sub-óptima. Para superar este problema, utilizaron un primer paso de inicialización para encontrar las tasas de inyección/producción totales apropiadas para la vida del yacimiento. Entonces en la segunda etapa, utilizaron estas mismas tasas y restricciones para aplicar el método de gradiente de proyección y de esta manera reducir los pozos a un número óptimo en ubicaciones óptimas, así como optimizar los controles de pozos de los pozos restantes.

Los resultados alcanzados por Fahim F. et al. Indican que el procedimiento en dos etapas proporciona una mayor optimización de la función objetivo que la optimización realizada en una sola etapa propuesta por Wang et al. (2007) y Zhang et al. (2010), es decir, la optimización sin la fase de inicialización a veces produce un Valor Presente Neto ligeramente menor a la optimización de dos etapas.

1.2. MODELOS PROXY.

Dado que determinada función de evaluación para el desarrollo de los yacimientos (por ejemplo, el simulador numérico); frecuentemente presenta una carga computacional muy pesada y extensa para una evaluación rápida y optima, de determinada función en estudio, por lo tanto, la idea de utilizar la aproximación más manejable y sencilla de esta función es un pensamiento muy atractivo. En la literatura, los modelos proxy se conocen como una función de representación o modelo sustituto. Esta función se puede ver la interpolación y la extrapolación de una función, o la regresión multivariable. Hay muchos candidatos para el desarrollo de modelos proxy. Algunos de los más utilizados y eficaces los presentaremos a continuación.

La regresión polinómica es el más común de los métodos de generación de modelos proxy. Su principal objetivo es calcular un polinomio de grado *n* de un conjunto de observaciones, se calcula utilizando la evaluación de la función completa. La idea se extiende fácilmente a múltiples dimensiones. Los problemas surgen cuando los datos son erráticos o las dimensiones son altas.

En tales casos, los polinomios de orden muy alto son necesarios para un ajuste razonable para los datos, lo que provoca una falta de generalización. Funciones Spline se han introducido para aproximar los datos con la ubicación, polinomios de orden inferior de funciones continuas (primera derivada existe). Esto funciona bien, las funciones Spline multidimensionales son muy complejas y este enfoque sigue siendo un área de investigación.

Kriging es un algoritmo de interpolación-extrapolación utilizado por Sacks te al. (1989) en las ciencias de la tierra para predecir las propiedades de la tierra. Con Kriging, tenemos control sobre las estadísticas de dos puntos de nuestras estimaciones a través del variograma. Por lo tanto, la configuración de los datos con respecto al punto que quiere ser estimado juega un papel importante, es decir, dependiendo la organización y configuración de los datos se obtendrá un punto de interpolación valido o no valido, que de igual forma está sujeto a la convergencia de los datos. Aunque no se utilizan comúnmente en las prácticas de ciencias de la tierra, el algoritmo se puede extender a múltiples dimensiones sin aumento significativo de la complejidad. Estimaciones Kriging también se utilizan en la ubicación exacta de conjunto de datos. Otra propiedad es favorable, dado un conjunto de datos, al utilizar todos los datos para la estimación (es decir, Kriging global), la matriz de covarianza tiene que ser calculada solamente una vez. Después, cada estimación sólo requiere la multiplicación de la matriz. También es posible obtener información de estimación de las variaciones en los puntos estimados que ayuda a extender la búsqueda a las zonas menos visitadas del espacio de búsqueda.

Redes Neuronales (NNs) (Anderson, 1995) se han utilizado también en la aproximación de funciones. Redes Neuronales han ganado la aceptación de las poderosas técnicas de interpolación en la comunidad de ingeniería. Redes Neuronales no son datos exactos, pero son capaces de representar funciones muy complejas, continuas o discretos. Redes Neuronales puede asignar espacios multidimensionales en el espacio dimensional inferior, Por lo tanto, se puede utilizar fácilmente para la regresión multivariable. A continuación, presentaremos una descripción más amplia de los métodos Kriging y Redes Neuronales (NNs)

1.2.1. Regresión polinomial: En particular, los modelos provenientes de regresiones polinómicas son los más empleados en la industria petrolera para el análisis y representación de procesos físicos. Esto debido, a su facilidad de entendimiento, flexibilidad y eficiencia computacional (Zubarev, 2009). En general, para formular una regresión de tipo cuadrática se emplea la ecuación 1.

$$y(x) = \beta_o \sum_{i=1}^{n_d} \beta_i x_i + \sum_{i=1}^{n_d} \sum_{j=1}^{n_d} \beta_{ij} x_i j_i + \sum_{i=1}^{n_d} \beta_{ii} xi^2 \qquad (1)$$

Donde x es el vector de variables de entrada de longitud n_d, x_i es un término lineal, $x_i x_j$es un término intermedio y x_i es el término cuadrático y β_0 , β_i , β_{ij}, β_j representan los coeficientes de regresión desconocidos. Los términos beta son determinados a partir de mínimos cuadrados. El uso de un modelo de regresión polinómica incluye la selección de los parámetros a incluir y la estimación de los coeficientes de regresión. Adicionalmente, se debe tener presente que este tipo de modelos no es eficiente para representar fenómenos altamente no lineales en tiempo y espacio.

1.2.2 Método Kriging: En este modelo se parte del principio que puntos próximos en el espacio tienden a tener valores similares contrario a los puntos más distantes. Es por ello, que basan su análisis en variogramas, por medio de los cuales pueden asignar pesos a los parámetros considerados para la construcción del modelo proxy (Fedutenku, Yang, Card & Nghiem, (2013)). Existen modelos Kriging tipo simple, ordinario y universal, sin embargo, el más empleado para la construcción de Proxys es el modelo Kriging ordinario. La ecuación para estimar el valor en el punto de interés es:

$$y\,(x) = f^T(x)\beta + r^t(x)R^{-1}(Y - F\beta) \qquad (2)$$

Donde $f(x)$ es el vector que contiene los términos de la función de regresión, β es el vector de los coeficientes de regresión desconocidos, $r^t(x)$ es un vector de correlación y R es una matriz de correlación, *F* y *Y* son vectores de valores de la función de regresión para el conjunto de datos de entrada y el vector de respuestas de conjunto de datos de entrada. Para estimar los valores del vector beta se requiere emplear un algoritmo de optimización.

Los modelos proxy construidos a partir de este método reproducen completamente la información empleada para su construcción; sin embargo, podrían no ser muy acertados con combinaciones de parámetros diferentes a los empleados en su construcción. El tiempo empleado para la construcción de este tipo de modelos depende de: el número de variables de entrada, el número de experimentos iniciales (diseño de experimentos), tipo de función y la capacidad computacional.
Pan y Horne (1998). [10]Usaron métodos de interpolación multivariable como Mínimos Cuadrados y Kriging como modelos proxis a la simulación del yacimiento. El propósito del primer algoritmo es construir una función que tiene una forma conocida simple para aproximar alguna función objetivo.

[10] PAN Y HORNE. *Improved Methods for Multivariate Optimization of Field Development Scheduling and Well Placement Desing. SPE 49055.1998*

El comportamiento de esta función objetivo se observa primero a través de una serie de simulaciones. Entonces, una función se construye de tal manera que minimiza la suma del cuadrado residual entre los datos y los valores de la función. Para comenzar su estudio, seleccionaron varias ubicaciones de pozos para la simulación numérica como una muestra para entrenar el proxy. A continuación, se generaron mapas de superficie de Valor Presente Neto (NPV) utilizando los dos modelos proxis. Estos mapas se utilizaron posteriormente para estimar los valores objetivos de la función en nuevos puntos. Observaron que el método de Kriging proporciona medios más precisos para estimar la función objetivo que la interpolación de mínimos cuadrados en los ejemplos probados.

1.2.3 Modelos Thin Plate Spline (Capa Delgada): Los modelos Spline de capa delgada es una interpolación multidimensional que puede ser aplicada un espacio arbitrario de datos11. Estos modelos se definen:

$$y(x) = \sum_{i=1}^{n_d+1} \beta_i m_i + \sum_{k=1}^{n_s} \theta_k U(\|x - x^k\|) \qquad (3)$$

Donde m es un vector de monomios, β_i y θ_k son los coeficientes desconocidos de la interpolación, $U(r) = r^2 \log(r)$ es la superficie Spline, x^k es un punto de muestra del conjunto de datos conocidos y x es el punto de interés interpolado. Este tipo de modelo tiene la capacidad de reproducir de forma muy exacta los datos de entradas, pero requieren que el número de experimentos sea mayor que el número de incertidumbre de cada parámetro.

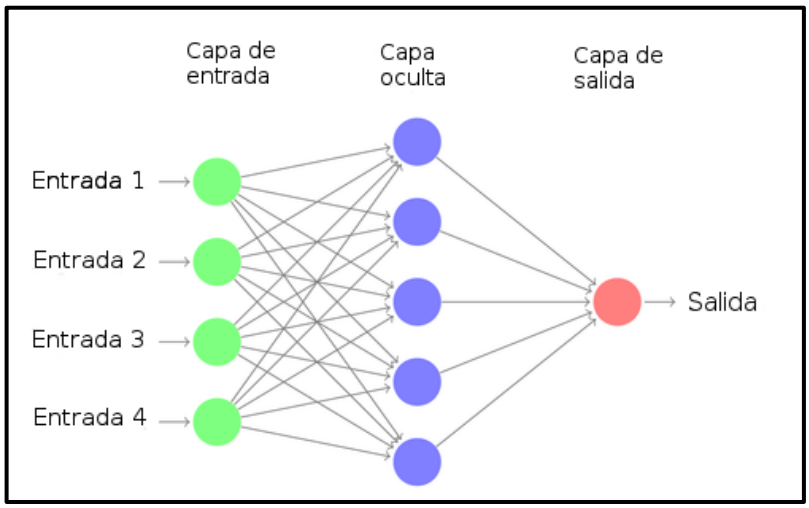
1.2.4. Método Neural Network: Las Redes Neuronales Artificiales, ANN (Artificial Neural Networks) están inspiradas en las redes neuronales bilógicas del cerebro humano.

[11] R. ARCHER., G. ZAKERI., U OF AUKLAND Y T. VAUDREY. Spline as an optimization Tool in Petroleum Engineering. 2005

Están constituidas por elementos que se comportan de forma similar a la neurona biológica en sus funciones más comunes. Estos elementos están organizados de forma parecida a la que presenta el cerebro humano.

Las ANN, además de parecerse al cerebro, presentan una serie de características propias del cerebro (i.e. aprenden de la experiencia, generalizan de ejemplos previos a ejemplos nuevos y abstraen las características principales de una serie de datos, Basogain, 2014). En la siguiente figura se presenta un esquema general de una red neuronal. En las ANN, la unidad análoga a la neurona bilógica es llamada elemento procesador, PE o nodo. Un elemento procesador tiene varias entradas y las combina, normalmente con una suma básica. La suma de las entradas es modificada por una función de transferencia y el valor de la salida de esta función de transferencia se pasa directamente a la salida del elemento procesador.

Figura 1. Esquema General de una Red Neuronal.



Fuente: Zangl, Graf & Al-kinani, 2006

La salida del PE se puede conectar a las entradas de otras neuronas artificiales (PE) mediante conexiones ponderadas correspondientes a la eficacia de la sinapsis de las conexiones neuronales.

Para construir una ANN, es necesario definir una topología: número de capas ocultas, nodos por capa y una función de activación. Esta última, es la encargada de calcular la información de salida para cada nodo. Por otra parte, el número de capas ocultas está limitado por el número de experimentos empleados para construir la ANN. Generalmente, para construir modelos proxy partiendo de redes neuronales se emplea la función de Sigmoid (ecuación 3).

$$y(x) = \frac{1}{1 + e^{x}} \quad (4)$$

Stoisits et al. (1999) El objetivo de este trabajo fue desarrollar un modelo de optimización que sería una mejora con respecto al modelo actual y también ser superior al software de optimización comercialmente disponible. Esto requirió el desarrollo de un modelo de producción de campo que incluyó modelos de desempeño de pozo, líneas superficiales y de instalaciones. Para cada iteración en el modelo de optimización se calcula la tasa de aceite del modelo de producción de campo.

Johnson y Rogers (2001) [12]usaron un proxy Neural Networks para el modelo numérico en un esfuerzo para optimizar las ubicaciones de pozos inyectores de agua para el campo de Pompano. Antes de la optimización, Johnson y Rogers preseleccionaron 25 ubicaciones candidatas para pozos de inyección basadas en criterios de inyectividad. El presente estudio encontró que una preselección de ubicaciones basadas en criterios distintos de la función objetivo puede ser limitante. Un apropiado grado de manejo y conocimiento de cada una de las técnicas presentadas anteriormente, permiten desarrollar herramientas auxiliares o sustitutas en procesos de simulación numérica de yacimientos, para la evaluación de determinada función a resolver. Es por eso por lo que en el desarrollo del trabajo

[12] JOHNSON Y ROGERS Applying Soft Computing Methods to Improve the Computational Tractability of a Subsurface Simulation Optimization Problem.2001

se hace necesario la consulta y discernimiento de estas técnicas para generar un ambiente de conocimiento y de guía respecto a la propuesta de generación de la herramienta propuesta por los autores, e igualmente generar el conocimiento de las formas, métodos, lenguajes, herramientas y procesos que estarán inmersas en el desarrollo de tal herramienta. De igual manera conocer los diferentes métodos y las diferentes herramientas generan una guía respecto al cómo y a la forma a utilizar para el desarrollar la herramienta.

2. MATLAB

2.1. DEFINICIÓN.

MATLAB es un programa de cálculo técnico y científico para el tratamiento de la información a través de matrices numéricas. Una potente herramienta de cálculo, útil para realizar desde elementales operaciones aritméticas hasta complejos algoritmos numéricos. Además, MATLAB posee un sencillo lenguaje de programación propio, válido para automatizar la resolución de un problema expresable matemáticamente. Como caso particular, puede también trabajar con números escalares, con cadenas de caracteres y con otras estructuras de información más complejas.

MATLAB facilita un código básico, al que se hará referencia en este manual, y un conjunto de librerías especializadas "$toolboxes$". Además, proporciona una relación interactiva con el usuario a través de tres tipos de ventanas:

- *Ventana de Herramientas:* Donde teclearemos las instrucciones para el funcionamiento del programa.
- *Ventana de Figuras:* Donde haremos las representaciones gráficas en dos y tres dimensiones.
- *Ventana de Edición:* Que situaremos en el editor propio que nos facilita el mismo.

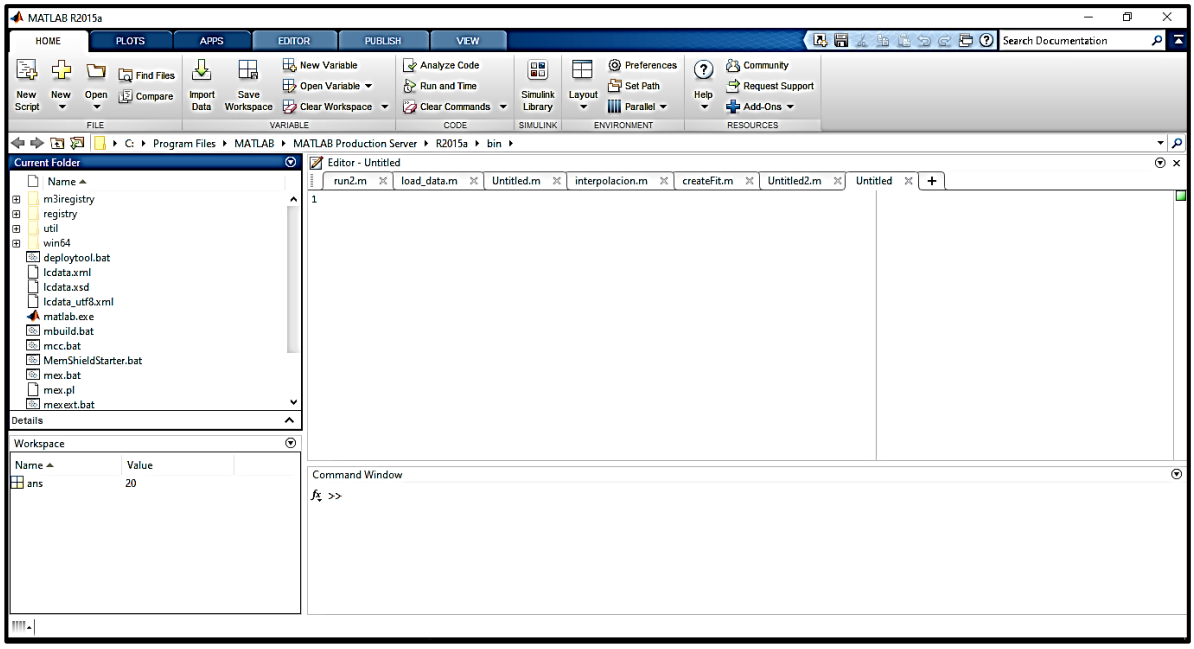
Así, en la ventana de comandos, integrada en la ventana de trabajo, ejecutaremos sencillas instrucciones que nos permitan manejar datos de entrada para generar resultados de salida. En la ventana de figuras visualizaremos gráficas correspondientes a estudios de interés. Finalmente, cuando la resolución de nuestro problema requiera la combinación sucesiva de diferentes instrucciones para crear estructuras lógicas más complejas, precisaremos diseñar algoritmos a través de M-archivos en la ventana de edición, que estará constituida por un procesador de textos integrado en MATLAB.

El entorno de trabajo MATLAB consta cinco partes

a) El entorno de desarrollo.
b) Las librerías de funciones.
c) El lenguaje de programación MATLAB.
d) El manejo de gráficos.
e) El Interfaz de Programas de Aplicación (API)

2.1.1 Ventana de trabajo: La ventana de trabajo o escritorio aparecerá inmediatamente al comenzar una sesión con la aplicación MATLAB.

Figura 2. Ventana de Trabajo Matlab.



En la ventana de trabajo (ver Figura 1), se encuentran los siguientes elementos invariantes:

- Barra de título**:** Con el nombre del programa.
- Barra de menú**:** Con todas las opciones del menú general.

- Barra de herramientas**:** Formada por varios iconos de acceso rápido a las opciones más utilizadas del submenú de ficheros y de edición.
- Ventana de comandos**:** (*Command Window*), donde se escriben las diferentes instrucciones con que daremos órdenes al sistema.
- Espacio de trabajo: (*Workspace*), donde se almacenarán las variables y resultados presentes en memoria y un listado de los componentes instalados (*Launch Pad*).

En la ventana de comandos cada instrucción debe ser escrita a continuación del inductor del entorno prompt (≫) que es el indicador de que MATLAB está preparado para recibir órdenes. Una vez escrita alguna instrucción siempre es necesario pulsar la tecla **enter** (↵) para que MATLAB la ejecute.

Tras realizar determinada operación numérica el sistema responderá creando, en el espacio de trabajo, una variable de sistema llamada **ans** en la que almacena el valor numérico obtenido, siempre que nosotros no hubiésemos creado alguna otra variable para almacenarlo.

Otras instrucciones de interés,

» clc

Limpia la información en la ventana de comandos, aunque, la información no esté almacenada en el espacio de trabajo.

» clf

Limpia la información gráfica visualizada en la ventana de figuras activa, pero no la cierra.

» close

Cierra la ventana de figuras activa, cuando exista.

- **Introducción de matrices en MATLAB**: Las estructuras básicas que utiliza MATLAB para manejar la información son las matrices Una matriz es una disposición rectangular de elementos ordenados, en filas y columnas, y encerrados entre corchetes, que podemos representar como sigue:

Figura 3.Estructura matriz básica.

Columna *j*-ésima ↴

$$A = \left(a_{ij}\right)_{\substack{i=1,2,\ldots,m \\ j=1,2,\ldots,n}} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2j} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3j} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{i1} & a_{i2} & a_{i3} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix}_{mxn} \leftarrow \text{Fila } i\text{-ésima}$$

Donde un elemento a_{ij} ocupa la fila *i*-ésima y la columna *j*-ésima dentro de la matriz. La identificación de los elementos por medio de los subíndices que los caracterizan tiene importancia capital para su manejo. Fijémonos que la matriz A posee *m* filas y *n* columnas, y nos permite almacenar *mxn* datos que pueden ser reales o complejos, de esta forma diremos que dicha matriz tiene orden o dimensión *mxn*.

A partir de la definición de matriz, los vectores y escalares se definen jugando con el número de filas y columnas. Así, un vector columna vendrá determinado por $m > 1$ filas y $n = 1$ columna, con lo cual:

$$A = \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ \vdots \\ a_{m1} \end{bmatrix}_{mx1}$$

Un vector fila vendrá determinado por $m = 1$ fila y $n > 1$ columnas, cuya expresión será:

$$A = [a_{11} \quad a_{12} \quad a_{13} \quad \cdots \quad a_{1n}]_{1xn} \qquad (5)$$

Como caso particular trabajaremos con escalares que son matrices de orden 1*x*1, es decir:

$$A = [a_{11}]_{1x1} \qquad (6)$$

MATLAB considera una matriz definida sin ningún elemento entre corchetes como una matriz de *dimensión cero*, es decir, vacía, aunque aparecerá declarada y almacenada en el espacio de trabajo. De esta forma nos aseguraremos su declaración vacía sin más que introducir.

$$\gg A = [\,]$$

Una matriz $A = \begin{bmatrix} 1 & 2 & 3 \\ -4 & 5 & -6 \\ 7 & -8 & 9 \end{bmatrix}$ se puede introducir en MATLAB en alguna de las siguientes formas:

$$\gg A = [1 \;\; 2 \;\; 3; -4 \;\; 5 - 6; \; 7 \;\; -8 \;\; 9]$$

$$\gg A = [1, 2, 3; \; -4, 5, -6; \; 7, -8, 9];$$

Observemos que con la introducción de punto y coma al final de una variable matricial, la matriz quedará guardada en la memoria de trabajo, pero no aparecerá explícitamente en la pantalla.

2.1.2 Operaciones aritméticas elementales con matrices: A continuación, mostramos las principales operaciones aritméticas elementales entre matrices considerando adecuadas las dimensiones de las mismas:

Tabla 1. Escritura en MATLAB.

OPERACION	**ESCRITURA EN MATLAB**
Suma de matrices.	$\gg A + B$
Producto de matrices.	$\gg A * B$
Potencia n-ésima de una matriz cuadrada.	$\gg A\hat{}n$
Suma de un escalar k a todos los elementos de una matriz.	$\gg A + k$
Multiplicación de un escalar k por los elementos de una matriz.	$\gg k * A$
Traspuesta de una matriz.	$\gg A'$
Determinante de una matriz cuadrada.	$\gg \det(A)$
Inversa de una matriz cuadrada.	$\gg inv(A)$
Rango de una matriz de orden *mxn.*	$\gg rank(A)$
División izquierda (similar a **inv(A)*B**).	$\gg A\backslash B$
División derecha (similar a **B*inv(A)**).	$\gg B/A$
Producto, elemento a elemento de dos matrices.	$\gg A.* B$
Cociente, elemento a elemento de dos matrices.	$\gg A./B$
Potencia n-ésima de los elementos de una matriz.	$\gg A.\hat{}n$
Potencia de los elementos de una matriz A cuyos exponentes son, elemento a elemento, los exponentes dados por B.	$\gg A.\hat{}B$
Calculo del orden de una matriz.	$\gg size(A)$

- **Operaciones aritméticas elementales con vectores**: Aunque los vectores constituyen un caso particular de las matrices, veamos algunas operaciones elementales donde los vectores están especialmente involucrados.

Si $x = (x_i)_{i=1,2,\dots,n}$ e $y = (y_i)_{i=1,2,\dots,n}$ son vectores del mismo orden, pueden definirse de forma específica, las siguientes operaciones aritméticas entre vectores:

1. Producto elemento a elemento: $x.* y = [x_1y_1 \quad x_2y_2 \ \dots \ x_ny_n]$ (7)

2. Producto escalar: $x * y' = [x_1y_1 + x_2y_2 + \dots x_ny_n]$ (8)

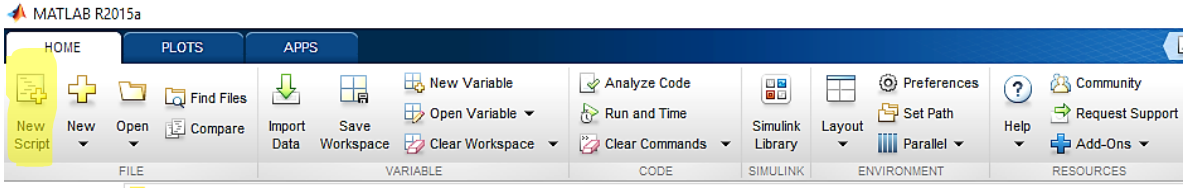
3. Producto de Kroneker: $x' * y = \begin{bmatrix} x_1 \\ \cdots \\ x_n \end{bmatrix} [y_1 \dots y_n] = \begin{bmatrix} x_1y_1 & \cdots & x_1y_n \\ \vdots & \cdots & \vdots \\ x_ny_1 & \cdots & x_ny_n \end{bmatrix}$ (9)

- **Creación de M-archivos.**

2.1.3 M-archivos script: Un script es un conjunto de instrucciones (de cualquier lenguaje) guardadas en un fichero (usualmente de texto) que son ejecutadas normalmente mediante un intérprete. Son útiles para automatizar pequeñas tareas. También puede hacer las veces de un "programa principal" para ejecutar una aplicación. En estos archivos se escriben series de comandos que desea ejecutar en conjunto. Los scripts no aceptan entradas y no devuelven ninguna salida. Operan en los datos del espacio de trabajo.

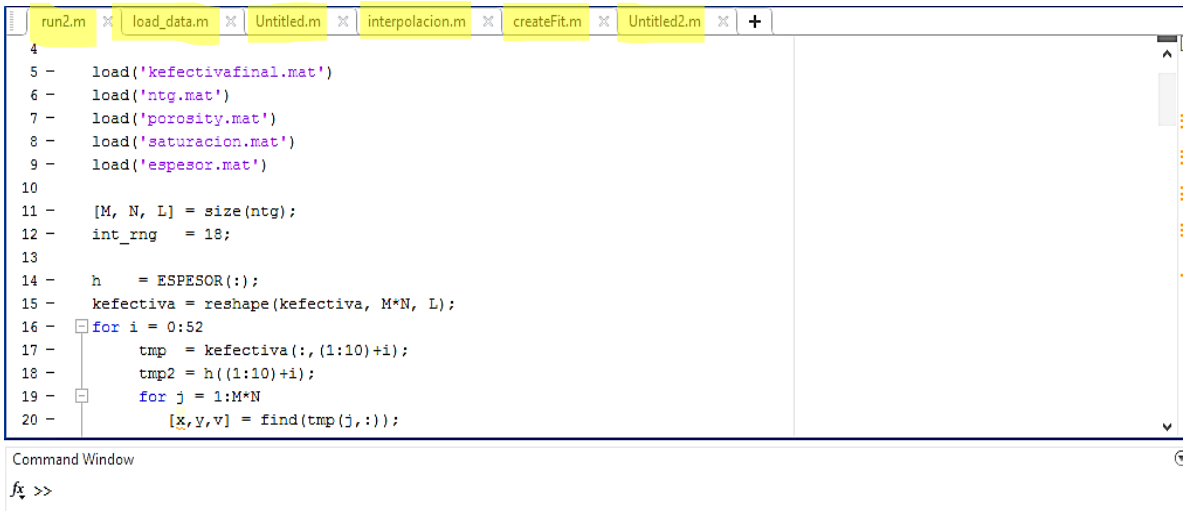
Así, para llevar a cabo una tarea, en vez de escribir las instrucciones una por una en la línea de comandos de MATLAB, se pueden escribir una detrás de otra en un fichero. Para ello se puede utilizar el Editor integrado: icono "$hoja\ en\ blanco$" del menú de herramientas, opción "$New\ M-file$" del Menú "$File$" o bien usando la orden >> $edit$.

Figura 4. Selección de un Nuevo Script.



Los scripts de MATLAB deben guardarse en un fichero con sufijo .m para ser reconocidos. Para ejecutar un script que esté en el directorio de trabajo, basta escribir su nombre (sin el sufijo) en la línea de comandos.

Figura 5. Ficheros con Script.

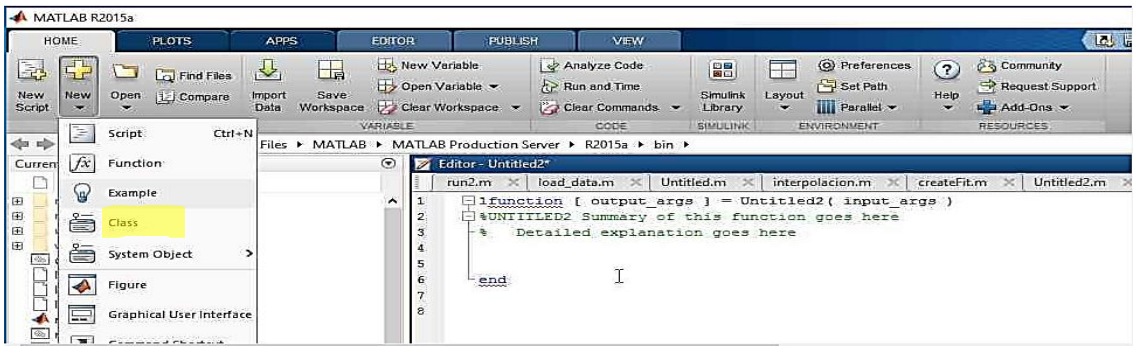


2.1.4. M-archivos función: Una función (habitualmente denominadas M-funciones en MATLAB), es un programa con una "interfase" de comunicación con el exterior mediante argumentos de entrada y de salida. En una función, a diferencia de un comando, se deben de pasar los argumentos. Las variables definidas y manipuladas dentro de la función son locales a esta y no operan globalmente en el espacio de trabajo. Los archivos de funciones se utilizan para extender a MATLAB, y crear nuevas funciones utilizando el lenguaje propio de MATLAB.

Su estructura general es:

$$function\ [variable\ de\ salida] = nombrefunction\ [variable\ de\ entrada]$$

Figura 6. Ventana de Comandos para la creación de una Función.



Las funciones deben guardarse en un fichero con el mismo nombre que la función y sufijo $.m$, se pueden incluir en el mismo fichero otras funciones, denominadas subfunciones, siempre y cuando la función que estemos llamando este en el mismo directorio donde está ubicado la otra función para que se puedan ejecutar.

Las órdenes evaluadas por la función, así como las variables intermedias creadas por estas órdenes, están escondidas, sólo son visibles las variables de entrada y salida. Esto hace que las funciones sean muy adecuadas para encapsular funciones matemáticas útiles o secuencias de órdenes que aparezcan a menudo.

MATLAB permite crear funciones propias en forma de archivos $.m$. Un archivo $.m$ de función es similar a un archivo script, al igual que ellos son archivos de texto creados en un editor de texto. La diferencia entre ambos es que la función sólo se comunica con el espacio de trabajo a través de las variables de entrada y salida, las variables intermedias dentro de la función no aparecen ni interactúan con el espacio de trabajo de MATLAB.

Para el desarrollo de la herramienta propuesta en este trabajo es necesario el conocimiento del ambiente de trabajo permitido por Matlab. De esta manera es posible establecer y seleccionar los diferentes métodos para el desarrollo de esta herramienta, es decir, un apropiado conocimiento de Matlab permite desarrollar las operaciones de lectura de documentos, carga de datos, procesamiento de datos, cálculos aritméticos y escrituras de códigos las cuales finalmente constituyen el

desarrollo de la herramienta propuesta en este trabajo de grado, en conclusión es importante el conocimiento y el manejo de la herramienta Matlab por lo cual se desarrolló una descripción en este capítulo del trabajo.

3. DESCRIPCIÓN DEL MODELO DE SIMULACION.

Un modelo de simulación numérica es una analogía representativa referente un modelo de campo entero, estableciendo similitudes durante su construcción principalmente sobre factores como: en el tamaño del modelo y configuración estructural del mismo, conservando las principales características geológicas, petrofísicas y de fluidos del modelo original.

El objetivo principal de implementar un modelo de simulación numérica es poder desarrollar y determinar los diferentes escenarios que se pueden presentar en la aplicación de determinado tipo de proyecto, que posteriormente pretende ser implementado en el modelo real de campo. Para este caso, se estudia el desarrollo de una metodología para lograr una adecuada ubicación de pozos productores dentro de un modelo sintético de simulación numérica. Con el objetivo final de proponer respectivas posiciones de ubicaciones de los pozos productores para el posterior desarrollo de la estrategia de producción.

El modelo de simulación base utilizado en el desarrollo de este proyecto fue construido y constituido a partir de información facilitada por el director de proyecto (Ing. John Pinto Carvajal), el cual trabaja en el área de simulación de yacimiento. La información facilitada hace parte de un módulo practico de simulación de la empresa de servicios de Schlumberguer. El modelo de simulación base hace referencia al campo Norte de Alwyn, el cual, situado en la parte sureste de la Cuenca Oriental de Shetland, produce petróleo y gas de los yacimientos del Grupo Brent y gas y condensado de la Formación Statfjord. El estilo estructural es de bloques de falla inclinados y erosionados que se sumergen hacia el oeste y se alinean de norte a sur conforme a la tendencia de falla normal principal.

Los elementos transversales NE-SW separan adicionalmente las acumulaciones de hidrocarburos.

Las columnas de hidrocarburos están restringidas al Tarbert ya la parte superior de la Formación Ness del Grupo Brent, en sedimentos asociados al retroceso del delta de Brent. La formación de Statfjord fue depositada en un ajuste aluvial.

3.1. DIMENSIONES DEL GRID DE SIMULACIÓN.

El primer paso en la creación de un modelo de simulación es establecer el tamaño del mismo, de igual manera es necesario establecer el tipo del enmallado a utilizar; para este caso se utiliza un enmallado cartesiano y el tamaño total del modelo de simulación fue de 10 acres.

Posterior a ello, debe establecerse el grado de discretización del modelo d simulación, es decir, el número de celdas en el cual será divido el modelo en cada una de las direcciones. La Tabla 1 presenta las dimensiones del enmallado del modelo base.

Tabla 2. Dimensiones del Grid de Simulación.

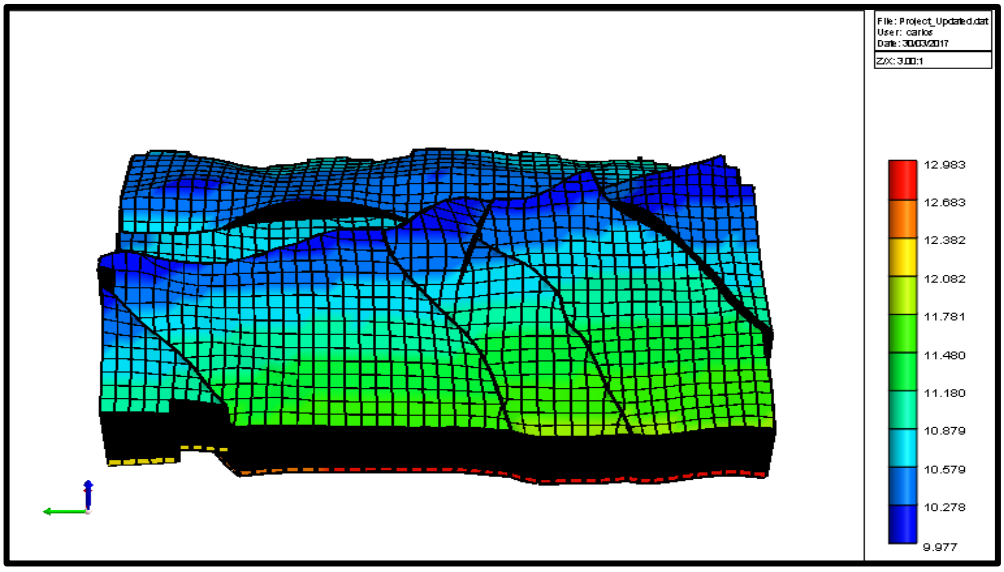
PROPIEDAD	VALOR
Divisiones en X	32
Divisiones en Y	47
Divisiones en Z	62
Número de Celdas	93248

Una mayor discretización del modelo permite obtener mayor detalle en los fenómenos analizados, sin embargo, implica un mayor tiempo de cómputo por parte del simulador. El número de las divisiones en cada una de las direcciones que componen el modelo se establecieron a juicio ingenieril a partir del criterio del ingeniero director de este trabajo, el criterio es basado bajo el conocimiento estructural y geológico del campo que se quiere representar en el modelo de

simulación, por lo tanto estas divisiones se seleccionaron a partir del juicio ingenieril con el principal objetivo de obtener un modelo de simulación representativo del campo objeto del desarrollo de este trabajo.

En la siguiente figura se observa el modelo base de simulación, el cual incluye en su construcción la presencia de siete fallas y de la misma forma se estableció la división del grid de simulación en cuatro unidades, para un mayor grado de similitud con el campo en cuestión.

Figura 7. Grid de Simulación



3.2. PROPIEDADES DEL MEDIO POROSO.

Las propiedades petrofísicas de la roca determinarán factores importantes en el desempeño del yacimiento, tales como: capacidad de almacenamiento de fluidos, la capacidad del medio rocoso para permitir el flujo de los fluidos almacenados, pérdidas de energía a nivel de yacimiento, etc.

Para el modelo simulación base en el desarrollo de este proyecto, el medio poroso está constituido por cuatro unidades geológicas, donde entre si varían las principales propiedades constituyentes del modelo, igualmente esta estructuralmente en presencia de siete fallas, para lograr el mayor grado de similitud con respecto al campo que se quiere representar. En la tabla 3 observamos la definición de cada una de las unidades geológicas y en la tabla 4 se observa la descripción de las todas las fallas existentes.

Tabla 3. Unidades Geológicas del enmallado de Simulación.

UNIDAD	TOPE (Layer)	FONDO (Layer)	NOMBRE
1	1	18	Tarbet
2	19	29	Ness 2
3	30	30	Ness 1
4	31	62	GeoUnit_4

Tabla 4. Fallas Geológicas del enmallado de Simulación.

FALLA	TOPE (Layer)	FONDO (Layer)	NOMBRE
1	1	62	FS1
2	1	62	FS2
3	1	62	FS3
4	1	62	FS4
5	1	62	FS5a
6	1	62	FS_Main_1
7	1	62	FS_Main_2

La utilización de diferentes unidades geológicas con su respectiva distribución de propiedades en el modelo de simulación ajusta el desempeño productivo del proceso a condiciones reales, debido a que de esta manera se tienen en cuenta las diversas unidades y estructuras geológicas a nivel de yacimiento fue necesario conocer la distribución de la porosidad en cada una de las unidades geológicas presentes en el enmallado de simulación. Estas propiedades pueden observarse en la Tabla 4.

Tabla 5.Distribución de Porosidad en el enmallado de Simulación.

POROSIDAD			
UNIDAD	MÍNIMO (%)	MÁXIMO (%)	DELTA (%)
Tarbet	0.1088	0.2796	0.1708
Ness 2	0.0010	0.2172	0.2162
Ness 1	0.000	0.2824	0.2884
GeoUnit_4	0.000	0.1057	0.1057

Tabla 6.Distribución de Permeabilidad Efectiva en el enmallado de Simulación.

PERMEABILIDAD (mD)			
UNIDAD	MÍNIMO (mD)	MÁXIMO (mD)	DELTA (mD)
Tarbet	159,4490	2556.1126	2396,6636
Ness 2	0.000	1071.4203	1071.4203
Ness 1	0.000	1196.2238	1196.2238
GeoUnit_4	0.000	226.5833	226.5833

Tabla 7.Distribución de Net To Gross en el enmallado de Simulación.

PERMEABILIDAD (mD)			
UNIDAD	MÍNIMO	MÁXIMO	DELTA (mD)
Tarbet	0.95	0.95	0.00
Ness 2	0.76	0.8075	0.0475
Ness 1	0.4067	0.5725	0.1658
GeoUnit_4	0.4067	0.5725	0.1658

La importancia de conocer la distribución de cada una de estas propiedades para el desarrollo de la metodología propuesta en este trabajo radica en la influencia directa que tienen estas propiedades en el almacenamiento y comportamiento de los fluidos (aceite y agua) presente en el yacimiento.

3.3. MODELO DE FLUIDOS.

El modelo de fluidos es uno de los componentes de mayor relevancia en un modelo de simulación numérica, dado que este representa el comportamiento de los fluidos hidrocarburos en yacimiento ante los cambios de presión, temperatura y volumen. Para representar lo anterior es necesario contar con diversas pruebas PVT, en caso de que eso sea imposible se requiere la elaboración de un PVT sintético, lo cual se consigue mediante la implementación de correlaciones numéricas para el cálculo de las diferentes propiedades PVT del fluido a diferentes presiones y temperaturas. En este modelo de simulación se utiliza un PVT sintético construido en la plataforma del Simulador IMEX, de la compañía CMG. En la Tabla 5 se observan los valores proporcionados al simulador para la elaboración del PVT artificial para un aceite negro.

Tabla 8. Propiedades de los Fluidos.

PROPIEDAD	VALOR
Temperatura de Yacimiento	190 °F
Presión de Burbuja	2715 psi
Densidad del Crudo	35 API
Densidad de Agua de Formación	60.78 lb/ft^3
Gravedad Especifica del Gas	0.66

En la siguiente, Figura 12, se observa el comportamiento del factor volumétrico de formación, Bo, y el comportamiento del gas en solución, Rs, contra la presión. Lo cuales son típicos para un aceite liviano como el que se representa en el presente modelo de simulación. El comportamiento del factor volumétrico de formación en función de la presión, dicho comportamiento es típico de un aceite liviano, en este caso, un crudo de 35° API.

El comportamiento de la viscosidad del crudo a diferentes condiciones puede ser observado en la siguiente Figura.

Figura 8.Factor volumétrico de formación (Bo) y Gas en Solución (Rs) vs Presión.

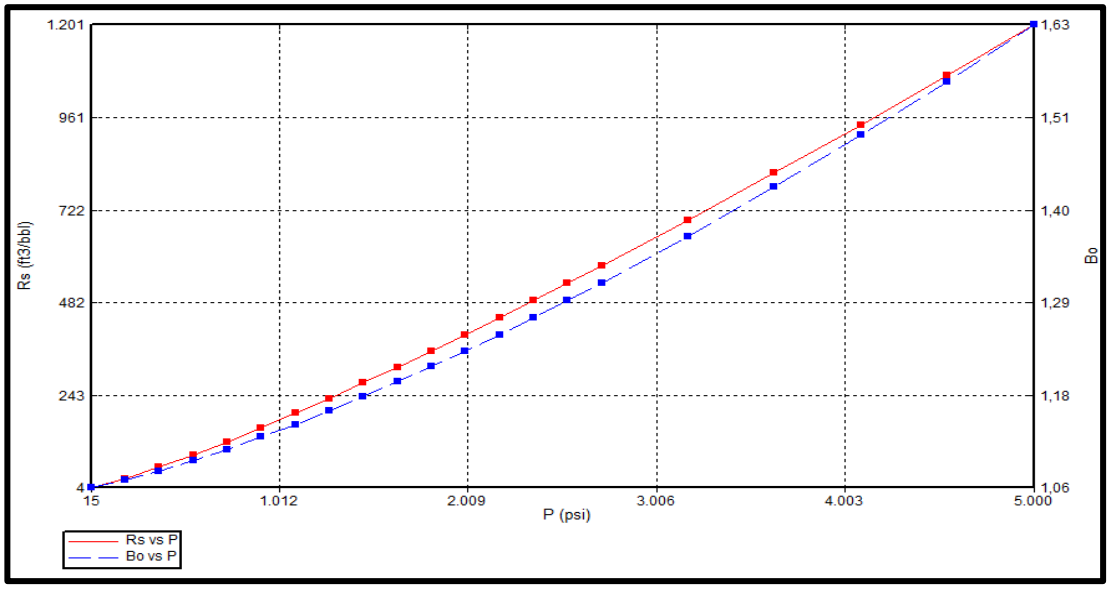
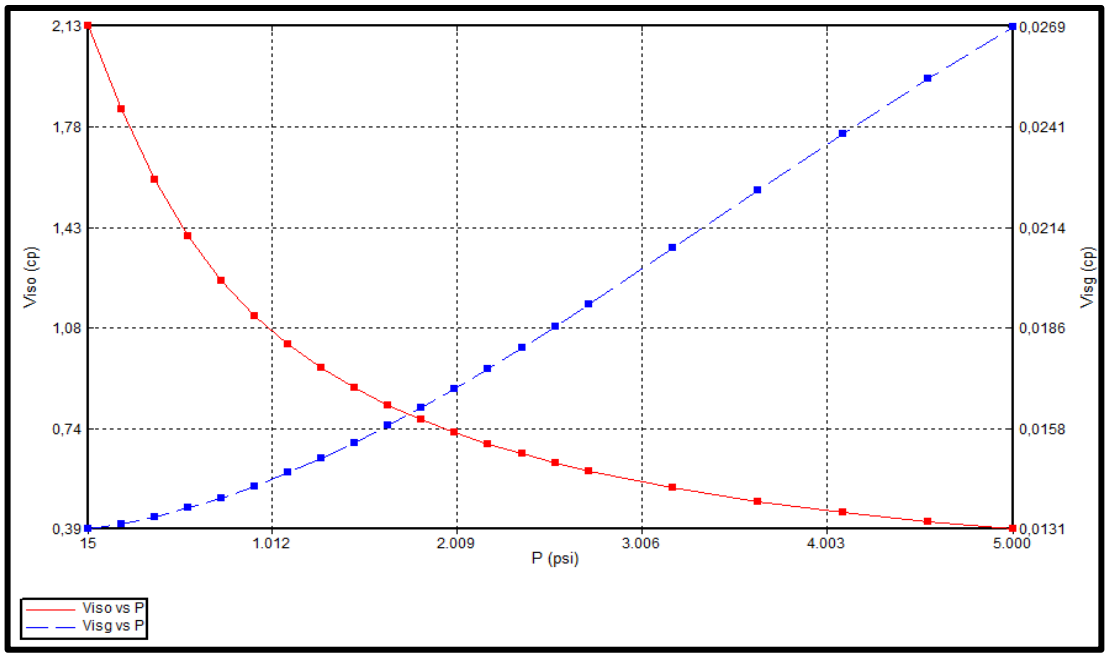


Figura 9.Viscosidad vs Presión.



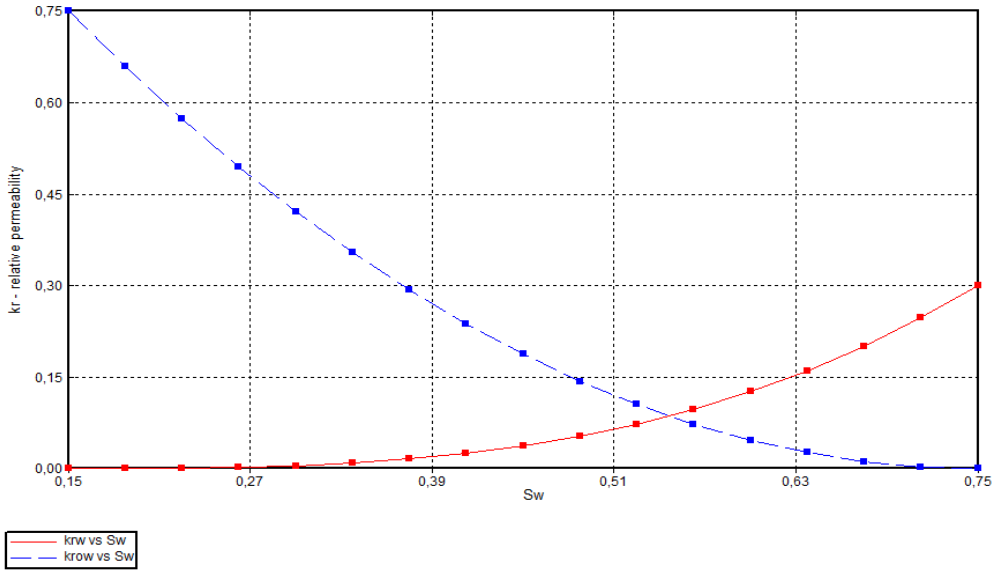
3.4. CURVAS DE PERMEABILIDADES RELATIVAS.

Debido a que no se contó con estudios petrofísicos puntuales, se procede a elaborar las curvas de permeabilidad relativa basados en los puntos finales (End- Points) de las mismas, para ello se utilizaron los valores descritos en la Tabla 7. Para ello se emplean las ecuaciones de Hirasaky para la construcción de las mismas.

Tabla 9. Puntos finales curvas de permeabilidad relativa.

END POINT	VALOR	END POINT	VALOR
Saturación de Agua Connata. (SWCON)	**0.15**	Saturación de aceite residual al agua (SORG)*	**0.15**
Saturación de agua Crítica (SWCRIT)	**0.15**	Saturación de gas Connata (SGCON)	**0.2**
Saturación de aceite irreducible al agua (SOIRW)	**0.25**	Saturación de gas crítica (SGCRIT)	**0.02**
Saturación de aceite residual al agua (SORW)*	**0.25**	Permeabilidad Relativa del agua @SORW (KRWIRO)	**0.3**
Saturación de aceite irreducible al gas (SOIRG)	**0.15**	Permeabilidad relativa del aceite @SWCON (KROCW)	**0.75**

Figura 10.Curvas de Permeabilidades Relativas.



3.5. CONDICIONES INICIALES DEL ENMALLADO DE SIMULACIÓN.

Las condiciones iniciales constituyen una parte muy esencial dentro de la construcción de un enmallado de simulación, en la construcción de nuestro modelo base se configuraron condiciones iniciales tales como: Sistema de equilibrio de Gravity-Capillary, Presión de referencia y Punto de contacto del sistema Agua-Aceite. La Tabla 8 presenta los valores de cada una de las condiciones iniciales establecidas para nuestro modelo base.

Tabla 10.Condiciones Iniciales del enmallado de Simulación.

CONDICIÓN INICIAL	VALOR
Presión de Yacimiento (REFPRES)	4525 psi
Profundidad de Referencia (REFDEPTH)	10402.1 Ft
Contacto Agua-Aceite (DWOC)	10827.3 Ft
Punto de Burbuja Constante (PB)	2715 psi

En el desarrollo de la configuración de los pozos, no se estableció ningún patrón de pozos predeterminado, ya que el principal objetivo del desarrollo de la metodología es determinar dichas ubicaciones. Se establecieron las condiciones operacionales de los pozos productores que serán ubicados, las dos condiciones establecidas son: restricción de producción de líquidos de 8000 Bbl/D , 10000 Bbl/D y 12000 Bbl/D una presión de fondo fluyendo de 2815 Psi.

4. METODOLOGIA APLICADA PARA LA DETERMINACIÓN DE LA UBICACIÓNES POTENCIALES DE LOS POZOS PRODUCTORES.

A continuación, se presenta la metodología desarrollada para la definición de la estrategia de ubicación de quince (15) pozos productores verticales en un enmallado de simulación Numérica. El desarrollo de esta metodología se realizó a través de la conjugación y aplicación de dos herramientas tales como: Software de Simulación Numérica de Yacimientos CMG (Computer Modelling Group) y la herramienta MATLAB.

La metodología construida a través de procesos alternativos entre estas dos herramientas consta principalmente de los pasos presentados a continuación:

- Exportación y almacenamiento de las variables del modelo de simulación (CMG) predominantes en la evaluación de las ubicaciones potenciales de los pozos en el enmallado de simulación.

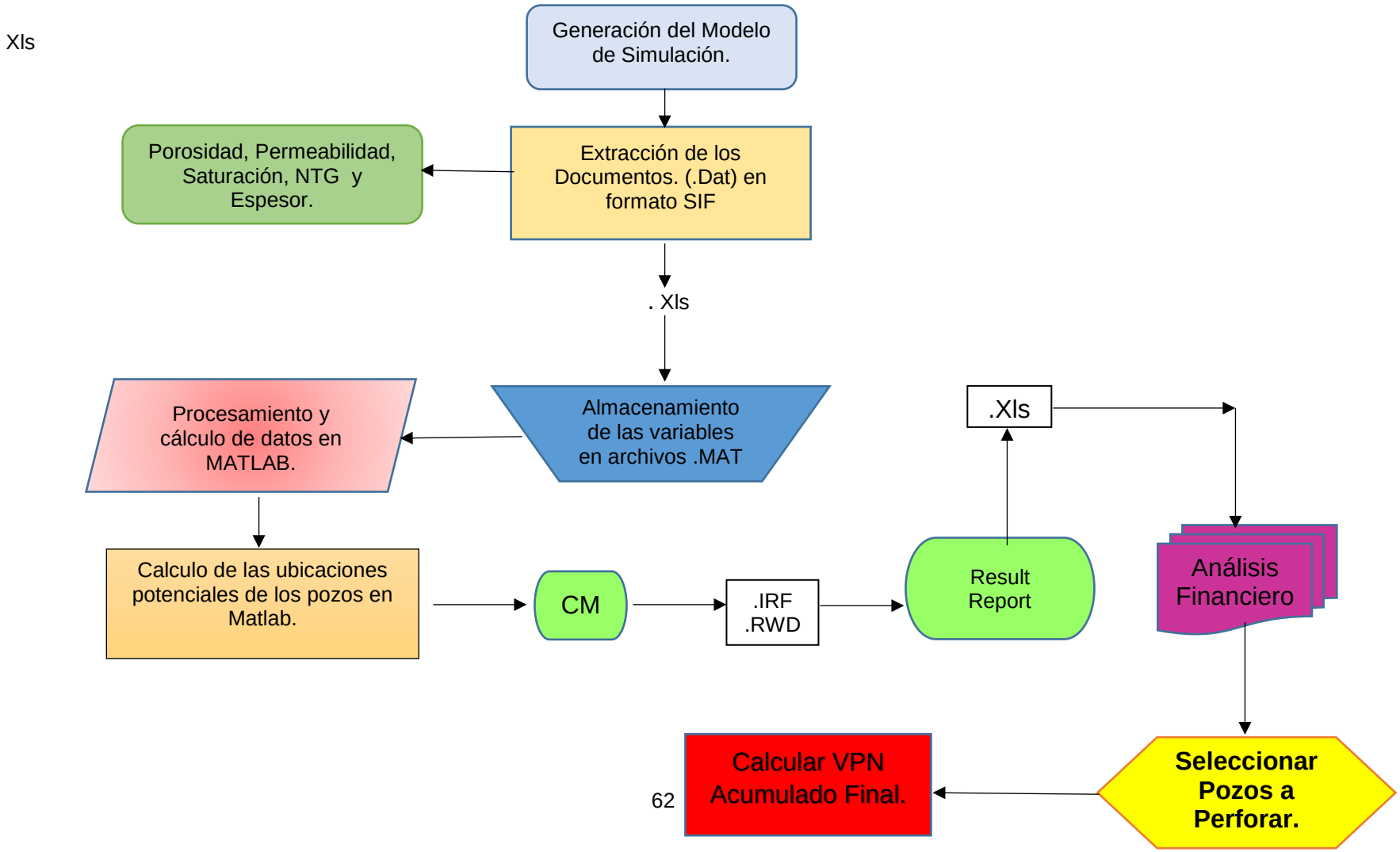
- Procesamiento y ejecución de operaciones aritméticas entre las diferentes variables previamente almacenadas.

- Selección de las ubicaciones correspondiente a los mayores potenciales y evaluación de la función objetivo correspondiente a cada una de las ubicaciones potenciales.

- Selección de los pozos a ubicar en el enmallado de simulación según el comportamiento de la función objetivo evaluada para cada ubicación. Definición y desarrollo de la estrategia de ubicación de los pozos productores.

- La siguiente figura muestra un esquema general de la metodología propuesta para la ubicación de los pozos verticales productores.

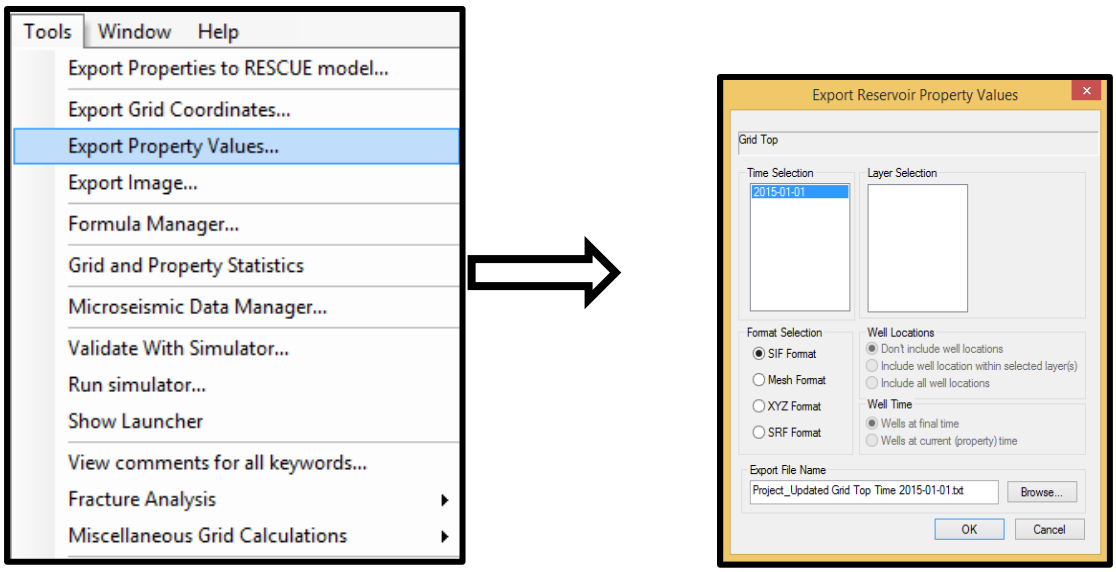
Figura 11. Metodología Aplicada para la Determinación de la Ubicaciones Potenciales de los Pozos Productores.



4.1. EXPORTACION Y ALMACENAMIENTO DE VARIABLES.

Luego de culminar la construcción del grid de simulación, es necesario conocer la distribución espacial (celda-celda) de cada una de las propiedades que se evaluaran para determinar las posibles ubicaciones potenciales para cada uno de los pozos a ubicar, las variables seleccionadas fueron principalmente cinco, las cuales son: Porosidad, Saturación, Net To Gross, Permeabilidad Efectiva al Aceite y el espesor de cada una de las capas que componen el modelo, que en su totalidad son 62 capas. La selección de dichas variables fue basada en la influencia directa que representan estas variables en el comportamiento del valor presente neto el cual es la función de evaluación en este trabajo, el comportamiento del valor presente neto está sometido al potencial productivo de la roca, calidad de la roca y otros parámetros operativos, por lo tanto con la selección de las cinco variables anteriormente mencionadas es posible evaluar la capacidad, calidad y potencial de respectivas ubicaciones donde se quiere evaluar el comportamiento del valor presente neto. De igual forma la selección de las variables se basó en un juicio ingenieril y en revisión bibliográfica de trabajos en los cuales desarrollaron una evaluación de una función objetivo igual a la evaluada en este trabajo. Para procedimiento DE exportación es posible utilizar la herramienta de exportación de variables del grid que nos brinda CMG. Esta herramienta nos permite obtener diferentes formatos de exportación como, por ejemplo; Formato SIF, Formato Mesh, Formato XYZ y Formato SRF. La dirección para ejecutar la exportación de las propiedades es la siguiente: *Barra de Herramientas>>Tools>>Export Property Values*

Figura 12. Exportación de los valores de cada propiedad en formato .SIF



El formato elegido para este trabajo es el formato .SIF, el cual podemos conocer la variación posicional en la dirección equis (x) de la propiedad la cual se exporto el documento; este documento contiene consecutivamente todas las variaciones de la propiedad en todas las direcciones (X,J,K) que componen el grid de simulación. Para realizar la lectura de las propiedades en este formato se debió tener en cuenta que para las celdas inactivas (Block Null) en el grid de simulación, estas se le asignaban un valor de cero en el documento de exportación. Es necesario conocer la distribución y posición de las celdas inactivas para cada una de las propiedades para realizar de manera correcta cálculos posteriores.

Posterior a haber realizado la exportación de todas las propiedades, estas son almacenadas en libros de Excel (. Xls). El almacenamiento de las variables exportadas se realizó previamente en Excel con el objetivo de mantener la correspondencia espacial (celda-celda) establecida en el simulador numérico, para luego ser almacenas en archivos Mat-File. No se realizó el almacenamiento directo de los archivos .SIF a Matlab debido a que no se presentaba una correspondencia espacial correcta en la exportación de las variables de esta forma directa.

De esta forma fue necesario generar un libro de Excel compuesto por 62 hojas las cuales cada una de ella representa un determinado layer o capa que componen el grid de simulación para posteriormente ser importadas a la herramienta de Matlab y desarrollar su almacenamiento. Consecutivamente al haber almacenado cada una de las variables en archivos de Excel es posible realizar el almacenamiento de cada una de las variables en Matlab empleando la herramienta de importación de Matlab ubicada en la barra de herramientas de Matlab. Inicialmente almacenamos la información de cada variable Layer por Layer, por ejemplo, para la variable porosidad se tenían 62 archivos. Mat uno por cada layer que compone al grid de simulación. El proceso de exportación de Excel a Matlab se realiza directamente entre las dos herramientas sin ninguna manipulación de los datos por parte de los autores. Con el objetivo de un manejo más rápido y eficiente dentro de la herramienta, se almacenan todos los archivos de una propiedad dentro de un mismo archivo (MAT-file) y esto se logra por la implementación de un Script para cada una de las propiedades. Como, por ejemplo:

Tabla 11. Script almacenamiento de las propiedades en archivos MAT-file.

Propiedad: Saturación.	Propiedad: Porosidad.

`clear all`
`close all`
`clc`
 	
`load('SATURACION/EXPORT/matlab.mat')`
 	
`saturacion = zeros(47, 32, 62);`
`for i=1:62`
`    eval(['saturacion(:,:,i)       =`
`',['SATURACION', num2str(i)],';']);`
`end`
`save saturacion.mat saturacion`
	`clear all`
	`close all`
	`clc`
	
	`load('POROSIDAD/Mat/matlab.mat')`
	
	`porosity = zeros(47, 32, 62);`
	`for i=1:62`
	`    eval(['porosity(:,:,i)       =`
	`',['POROSITY', num2str(i)],';']);`
	`end`
	`save porosity.mat porosity`

Este fragmento del código emplea un ciclo (for) para lograr el almacenamiento desde las capas 1 hasta la capa 62, almacenando así la totalidad de capas que conforman el grid de simulación, se almacena en una matriz de igual dimensiones que el Grid de simulación que estamos utilizando. Esta matriz contiene los valores celda-celda de la propiedad que haya sido almacenada. Cuando se ha logrado el almacenamiento de todas las variables en archivos MAT-file es posible realizar el procesamiento de estos datos en Matlab.

En principal propósito de realizar un buen almacenamiento de las variables es lograr realizar cálculos y operaciones numéricas entre sí de manera correcta para finalmente obtener el valor de un potencial de almacenamiento y flujo de fluidos líquidos presentes en yacimiento representado por el modelo de simulación.

Le asignamos el nombre de ‘Potencial’ al valor numérico que indica las zonas donde posiblemente se haga una ubicación inicial de los pozos para su posterior análisis económico.

Figura 13.Comparación de la distribución espacial de la porosidad en el Layer 11,25 para ambos programas utilizados. (Matlab y CMG).

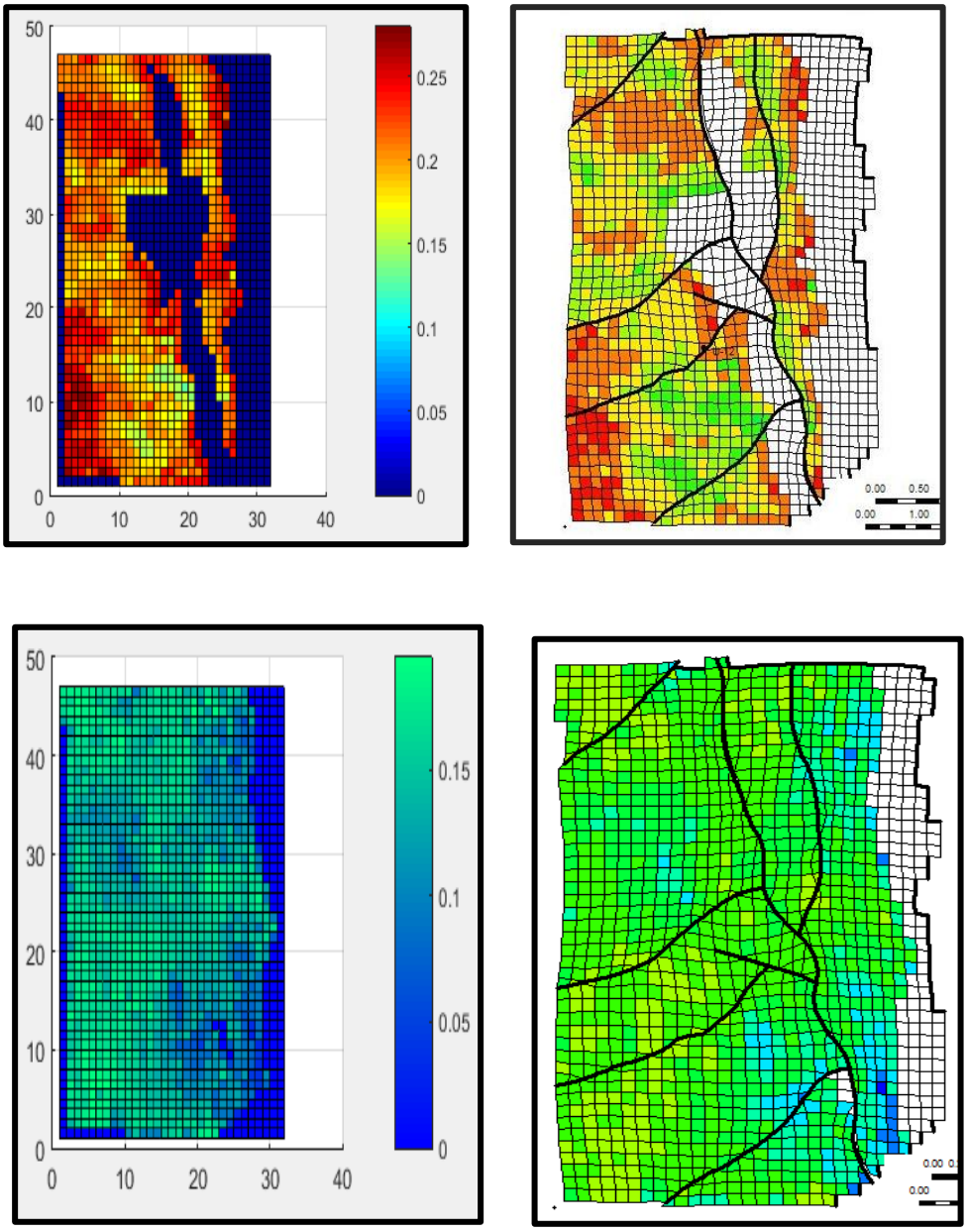


Figura 14. Comparación de la distribución espacial de la saturación en el Layer 18,41para ambos programas utilizados. (Matlab Y CMG)

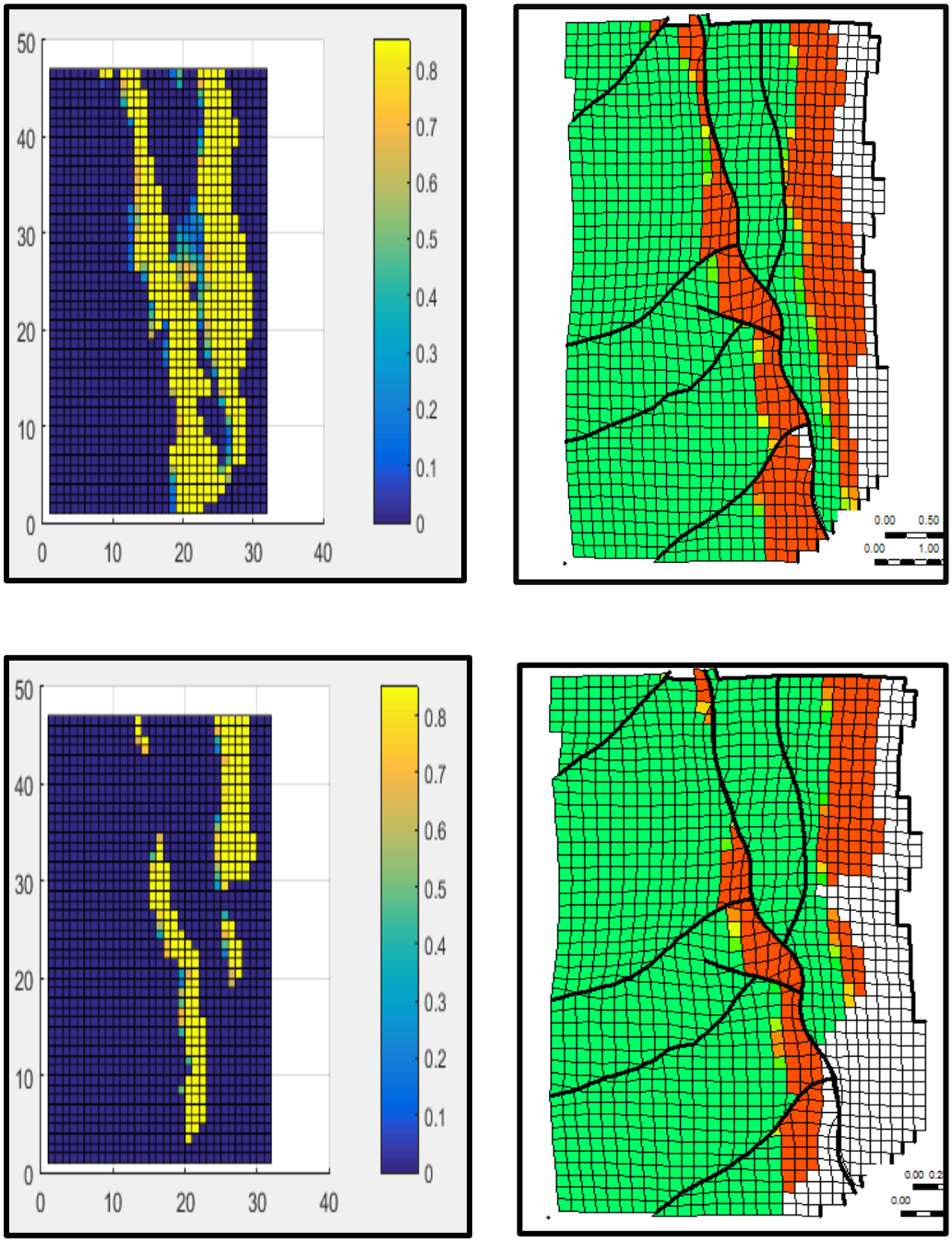


Figura 15. Comparación de la distribución espacial del Net To Gross de las capas 11,41para ambos programas utilizados. (Matlab Y CMG).

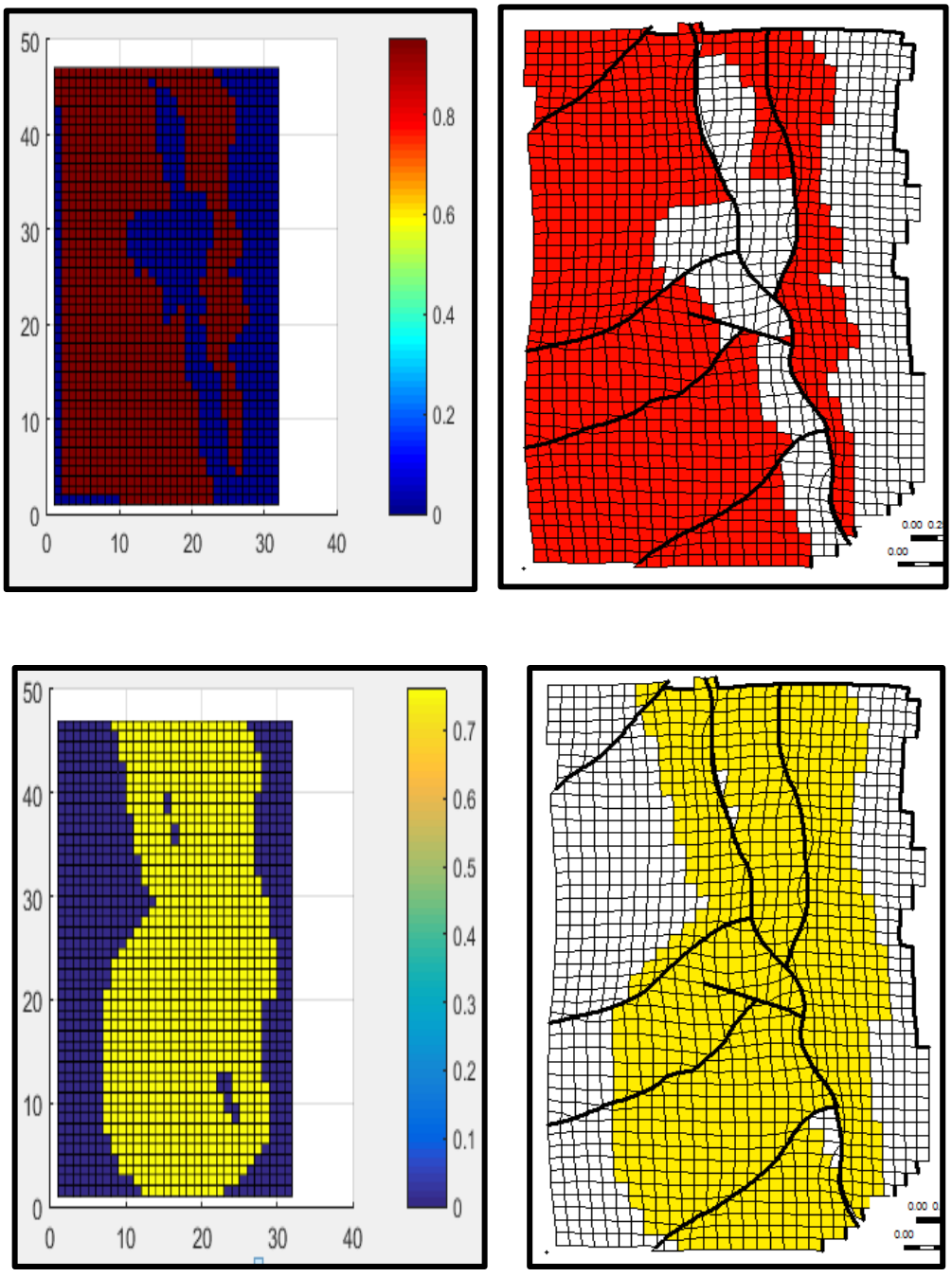
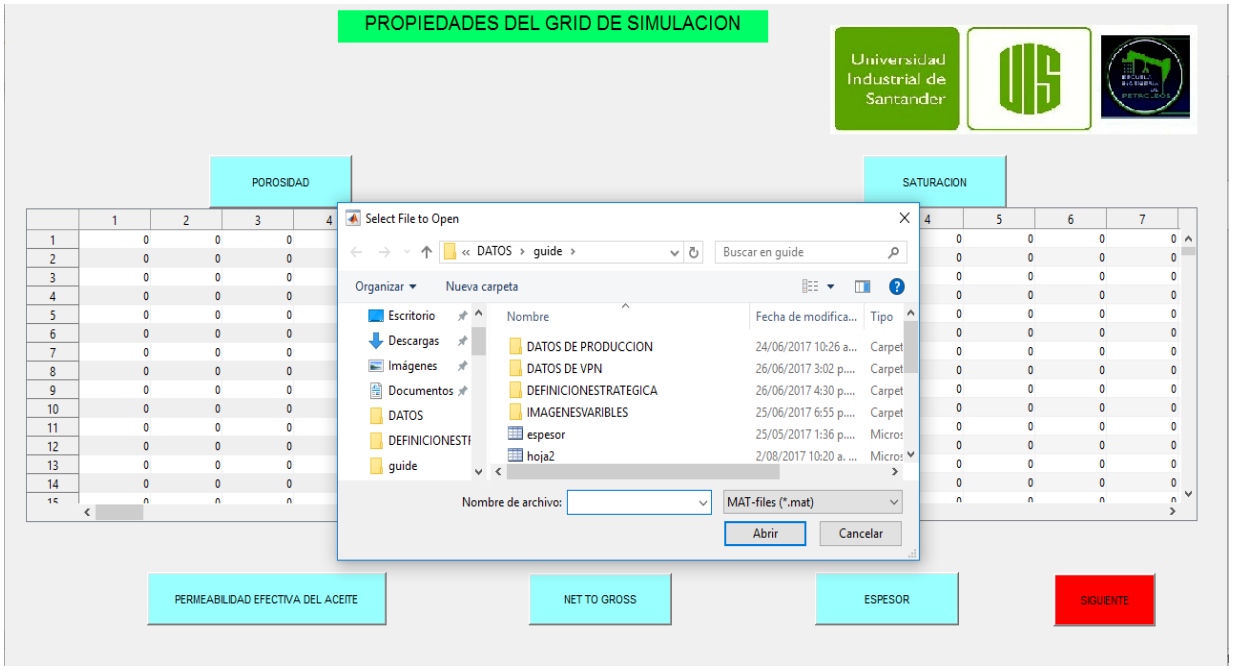


Figura 16. Ingreso de las propiedades del Grid de simulación a la Herramienta.



Las propiedades del grid de simulación son ingresadas a la interfaz por acción de cada uno de los botones que representa a cada variable, por ejemplo, el botón de POROSIDAD permite leer el archivo de almacenamiento de la variable porosidad y así seleccionarlo y cargarlo en la interfaz para su posterior uso. De igual manera se realiza el ingreso o lectura de las demás variables.

Para realizar la lectura y almacenamiento para la propiedad de la Permeabilidad efectiva al aceite se realizó una regresión polinómica de orden 2 ($R^2 = 1$), para obtener una relación analítica entre la permeabilidad relativa al aceite y la saturación de agua, debido a que esta propiedad no puede ser exportada de forma directa del grid de simulación.

$$K_{raceite} = 2.0833(Sw^2) - 3.125(Sw) + 1.1719 \qquad (10)$$

Con la aplicación de la ecuación (5) es posible obtener la permeabilidad relativa al aceite para cada una de las celdas que componen el enmallado de simulación; previamente es necesario conocer la distribución de la saturación de agua celda a celda en el modelo y luego aplicar la ecuación antes mencionada. La aplicabilidad

de esta ecuación se encuentra entre los rangos de saturación de agua irreducible y saturación de agua máxima($S_{wirr} = 0.15$; $S_{wmax} = 0.75$).

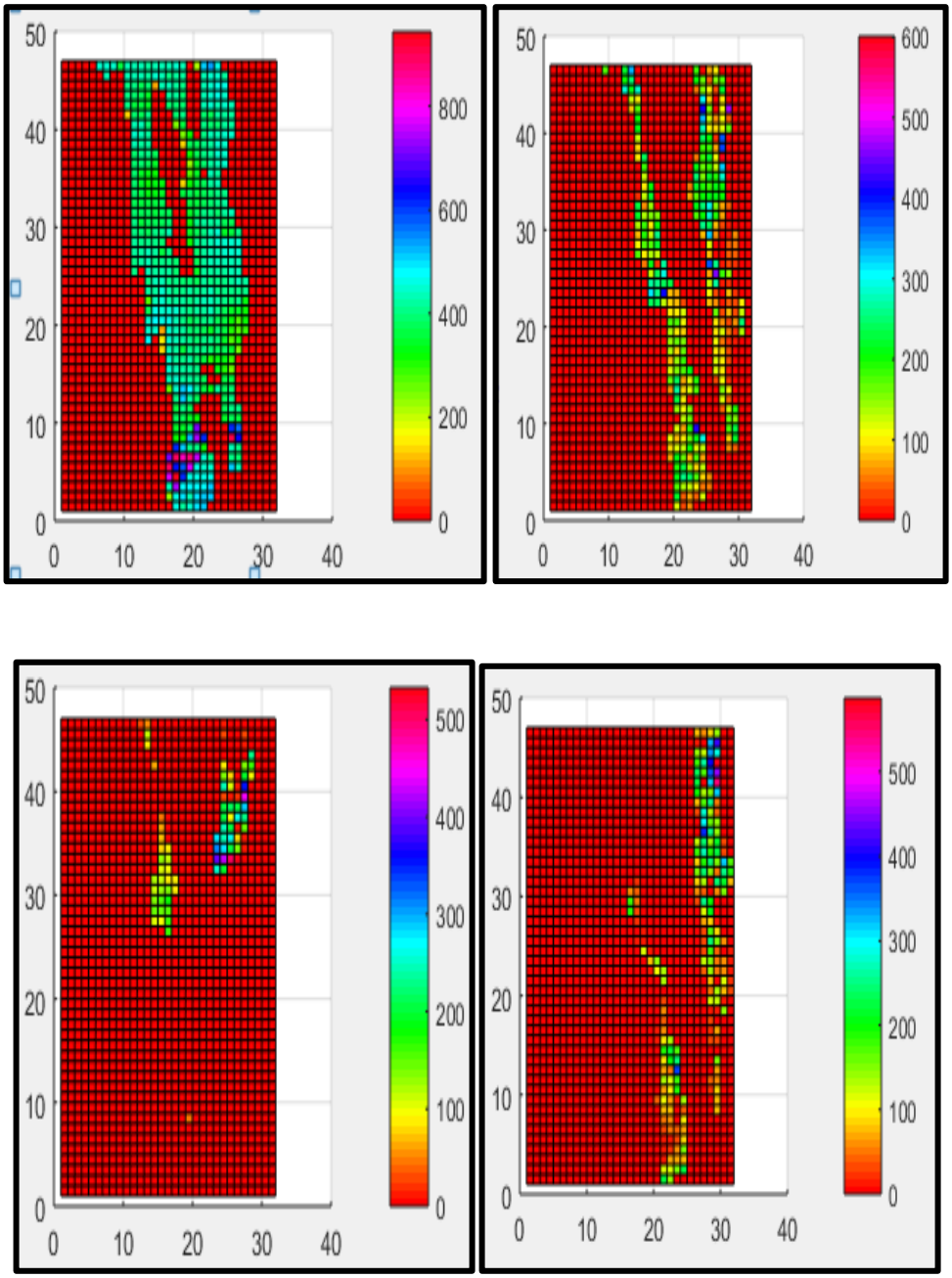
Debido a que en el formato de exportación le asigna valores de cero para las celdas inactivas (Null Blocks) para dichas celdas no es aplicada la ecuación y se le asigna un valor de cero a la propiedad de permeabilidad efectiva para esa celda, igualmente para las celdas que se encuentran por debajo del contacto agua-aceite (DWOC) las cuales tienen un valor de saturación de uno se le asigna un valor de cero a la propiedad de permeabilidad efectiva al aceite debido a que esta fase no está presente en ninguna de estas celdas. De esta manera es posible almacenar los valores de permeabilidad relativa al aceite en los rangos aplicados para esta ecuación para cada celda.

Luego de obtener la permeabilidad relativa al aceite para cada celda, se realiza la exportación del grid de simulación los valores de permeabilidad absoluta para cada una de las celdas, estas matrices tienen igual dimensión. Para determinar la permeabilidad efectiva al aceite se aplica la siguiente ecuación:

$$K_{Efectivaoil} = K_{raceite} * K_{absoluta} \qquad (11)$$

De esta forma se realiza un producto punto entre las dos matrices y se obtiene la permeabilidad efectiva al aceite celda a celda y luego es almacenada en una matriz de igual dimensiones que el enmallado de simulación; en un archivo MAT-file.

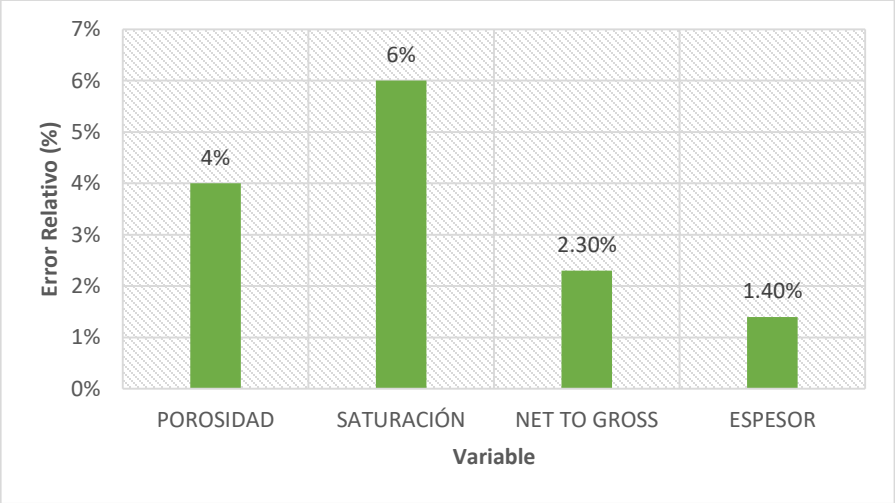
Figura 17. Distribución de la Permeabilidad efectiva al aceite en las capas 15, 28, 35,50.



A través de las imágenes comparativas de los datos almacenados en Matlab y CMG respectivamente, es posible observar un alto grado de similitud y correspondencia del almacenamiento de los datos en cada una de las herramientas, es decir, se obtiene un buen almacenamiento representativo de los datos en Matlab respecto a

los almacenados en el simulador numérico CMG. Sin embargo, durante el procedimiento de exportación y almacenamiento de los datos es posible tener un margen de error debido al manejo y escalamiento de los datos de una herramienta a otra. Para la verificación del almacenamiento de los datos se realizó un cálculo de un error relativo aritmético para cada una de las variables exportadas y almacenadas. El cálculo del error relativo aritmético se realizó celda a celda para la totalidad de capas que componen el modelo de simulación, obteniendo finalmente un porcentaje promedio del error presentado por cada variable durante su exportación y almacenamiento a la herramienta. La siguiente figura muestra el cálculo del error relativo para cada variable.

Figura 18. Error Relativo de Almacenamiento de Variables.



El error relativo se presenta debido a que durante la lectura y el almacenamiento de los datos en Matlab se pudo presentar la omisión de la lectura de algunos decimales en los valores numérico de las variables, igualmente se pudo presentar un error en la correspondencia espacial de los datos exportados respecto a los almacenados. En conclusión, es posible decir que para las cuatro variables se presentan errores relativos bajos, esto se debe a que la exportación y el almacenamiento de los datos fueron realizados de forma directa entre Excel y Matlab, omitiendo la manipulación manual de los datos por parte de los autores del proyecto.

4.2. CALCULO DEL PROMEDIO ARITMETICO PARA CADA PROPIEDAD.

El cálculo promedio de cada una de las propiedades exportadas hace referencia a un valor numérico en el cual se tiene en cuenta el número de capas que definen el intervalo de perforación en el cual se quiere calcular el promedio de determinada propiedad. El número de intervalos puede ir desde una capa hasta la totalidad de las capas que conforman el grid de simulación.

Para el desarrollo de cálculo del promedio aritmético se estableció que estos no serían estáticos y que podían variar, la variación de estos intervalos puede ser establecida y ejecutada a través de un Script desarrollado en Matlab. El script funciona de manera que el usuario pueda realizar cálculos promedio desde la capa una hasta la capa en la cual el desee establecer su punto base de perforación. Este intervalo de perforación debe tener en cuenta la condición de la profundidad del contacto Agua-Aceite establecida en el grid de simulación. En el desarrollo de nuestra metodología se establecieron intervalos de 10 capas, 15 capas y 20 capas, donde se tuvo en cuenta la profundidad del contacto Agua-Aceite.

A pesar de que el script desarrollado nos permite realizar cálculos para cualquier intervalo de perforación, se han establecido previamente estos intervalos mencionados por razones tales como: La profundidad del contacto Agua-aceite ($DWOC = 10827{,}3\ ft$), el costo de perforación de los pozos y la distribución de las propiedades en el enmallado de simulación.

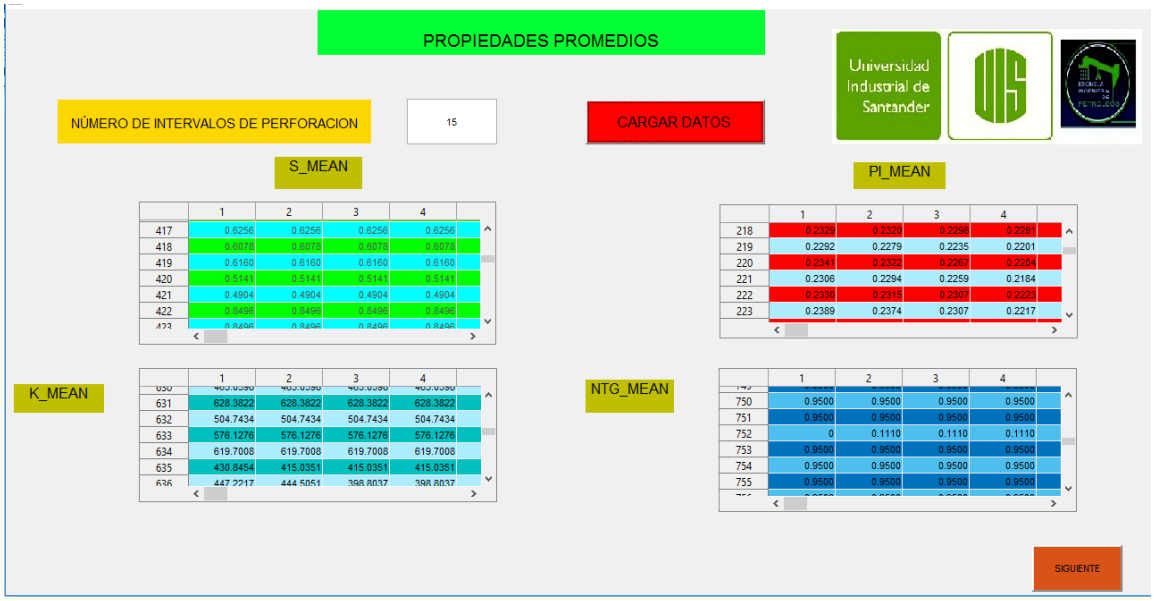
Para el caso donde se tiene los intervalos establecidos de 10 capas, 15 capas y 20 capas la distribución de intervalos en la totalidad del enmallado de simulación queda como lo muestra la siguiente tabla.

Tabla 12. Distribución de intervalos de cálculo de propiedad promedio.

INTERVALOS	10 CAPAS	15 CAPAS	20 CAPAS
Sección 1	1 – 10	1 – 15	1 – 20
Sección 2	2 – 11	2 – 16	2 – 21
Sección 3	3 – 12	3 – 17	3 – 22
Sección 4	4 – 13	4 – 18	4 – 23
Sección *n*	$N – (n + 9)$	$N – (n + 14)$	$N – (n + 19)$

Para ingresar o establecer el número de intervalo de perforación que el usuario requiera, debe ingresar el número de intervalo en el botón de *edit text* presente en la interfaz de la herramienta. En la siguiente figura muestra un ejemplo de ingreso del intervalo de perforación de 15, por parte del usuario.

Figura 19. Ingreso de Número de intervalo de perforación.



El cálculo del promedio aritmético de cada variable se realizó aplicando las siguientes fórmulas para cada propiedad en particular:

$$K_{mean} = \sum_{i=1}^{n} \frac{K_1 * h_1 + K_2 * h_2 + K_3 * h_3 + \cdots + K_n * h_n}{h_1 + h_2 + h_3 + \cdots + h_n} \quad (12)$$

Donde

$K_{mean} = Permeabilidad\ Promedio\ en\ enesimo\ intervalo, mD$

$K_1 = Valor\ de\ permeabilidad\ en\ la\ primera\ celda\ del\ intervalo, mD$

$h_1 = Espesor\ Promedio\ de\ la\ primera\ capa\ del\ intervalo, ft$

La anterior expresión se aplica para el cálculo de la permeabilidad promedio ((K_{mean}), para el calculo de las otras propiedades aplicamos un procedimiento similar, por ejemplo, calculo promedio aritmético de la porosidad:

$$PI_{mean} = \sum_{i=1}^{n} \frac{\emptyset_1 * h_1 + \emptyset_2 * h_2 + \emptyset_3 * h_3 + \cdots + \emptyset_n * h_n}{h_1 + h_2 + h_3 + \cdots + h_n} \quad (13)$$

Donde

$K_{mean} = Porosidad\ Promedio\ en\ enesimo\ intervalo, mD$

$K_1 = Valor\ de\ porosidad\ en\ la\ primera\ celda\ del\ intervalo, mD$

$h_1 = Espesor\ Promedio\ de\ la\ primera\ capa\ del\ intervalo, ft$

En el cálculo de los promedios aritméticos de cada propiedad para su posterior almacenamiento en matrices de forma individual, se tuvo en cuenta que los archivos de exportación de CMG (Computer Modelling Group) asignan un valor de cero para aquellas celdas inactivas presentes en el grid de simulación, estas celdas son conocidas como celdas nulas o inactivas.

Teniendo esto presente se suscribió en una parte del código donde la longitud del vector de espesores $[h_1 + h_2 + h_3 + \cdots + h_n]$, denominador del cálculo del promedio aritmético, fuese modificada según la presencia de celdas inactivas en el cálculo del promedio aritmético de respectiva propiedad.

El cálculo y almacenamiento de los múltiples promedios aritméticos generados por su respectiva propiedad se suscribió el siguiente fragmento dentro del código de programación de Matlab:

Tabla 13. Script almacenamiento promedios aritméticos para intervalo de 10 capas.

Matriz de Permeabilidad Efectiva (K_mean)	Matriz de Saturación Aceite (S_mean)

kefectiva = reshape(kefectiva, M*N, L);
for i = 0:52
tmp = kefectiva(:,(1:10)+i);
tmp2 = h((1:10)+i);
for j = 1:M*N
[x,y,v] = find(tmp(j,:));
if isempty(v)
K_mean(j,i+1) = 0;
else
K_mean(j,i+1) = sum(v(:).*tmp2(y))/sum(tmp2(y));
end
end
end
	saturacion = reshape(saturacion, M*N, L);
	for i = 0:52
	tmp = saturacion(:,(1:10)+i);
	tmp2 = h((1:10)+i);
	for j = 1:M*N
	[x,y,v] = find(tmp(j,:));
	if isempty(v)
	S_mean(j,i+1) = 0;
	else
	S_mean(j,i+1) = sum(v(:).*tmp2(y))/sum(tmp2(y));
	end
	end
	end

Tabla 14. Script almacenamiento promedios aritméticos para intervalo de 15 capas.

Matriz de Permeabilidad Efectiva (K_mean)	Matriz de Saturación Aceite (S_mean)

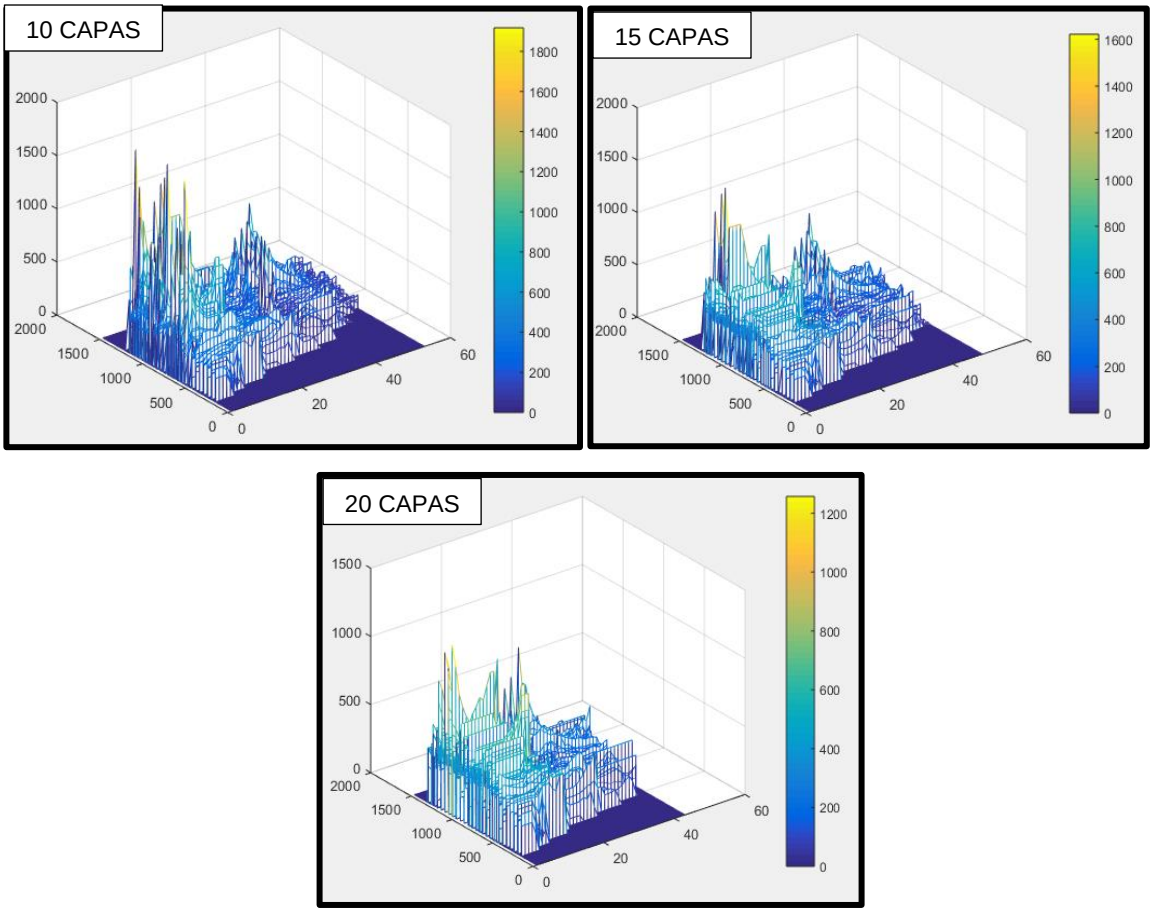
kefectiva = reshape(kefectiva, M*N, L);
for i = 0:47
tmp = kefectiva(:,(1:15)+i);
tmp2 = h((1:15)+i);
for j = 1:M*N
[x,y,v] = find(tmp(j,:));
if isempty(v)
K_mean(j,i+1) = 0;
else
K_mean(j,i+1) = sum(v(:).*tmp2(y))/sum(tmp2(y));
end
end
end
	saturacion = reshape(saturacion, M*N, L);
	for i = 0:47
	tmp = saturacion(:,(1:15)+i);
	tmp2 = h((1:15)+i);
	for j = 1:M*N
	[x,y,v] = find(tmp(j,:));
	if isempty(v)
	S_mean(j,i+1) = 0;
	else
	S_mean(j,i+1) = sum(v(:).*tmp2(y))/sum(tmp2(y));
	end
	end
	end

Es posible observar el pequeño cambio que se debe realizar en el script para cuando se quiera modificar el intervalo de perforación, se debe cambiar el ciclo del `for` y se debe modificar la longitud del vector (h). Por ejemplo, para cuando se tiene intervalos de 10 capas el ciclo for es: `for i = 0:52,` y para cuando se tiene intervalos de 15 capas el ciclo for es `for i = 0:47.`

El ciclo for inicia desde cero y termina en el número de intervalos que resulten en el enmallado de simulación, dependiendo del número de capas que incluya cada intervalo.

De igual manera el script nos permite observar gráficamente el cálculo y distribución del promedio aritmético para cada uno de los intervalos perforados generado por las consecutivas operaciones entre el total de celdas constituyentes del grid de simulación.

Figura 20. Valores de K_mean (mD) para intervalo de perforación 10, 15, 20 capas Almacenados en Matlab.



De forma lógica la magnitud del promedio para la propiedad de permeabilidad efectiva al aceite disminuye a medida que aumenta el intervalo de perforación. Sin embargo, igualmente es posible observar que los valores más altos de esta propiedad se encuentran distribuidos en posiciones muy similares dentro del plano

para los diferentes intervalos de perforación, generando así zonas particulares de buenos valores de esta propiedad.

Figura 21. Valores de S_mean (mD) para intervalo de perforación 10, 15, 20 capas Almacenados en Matlab.

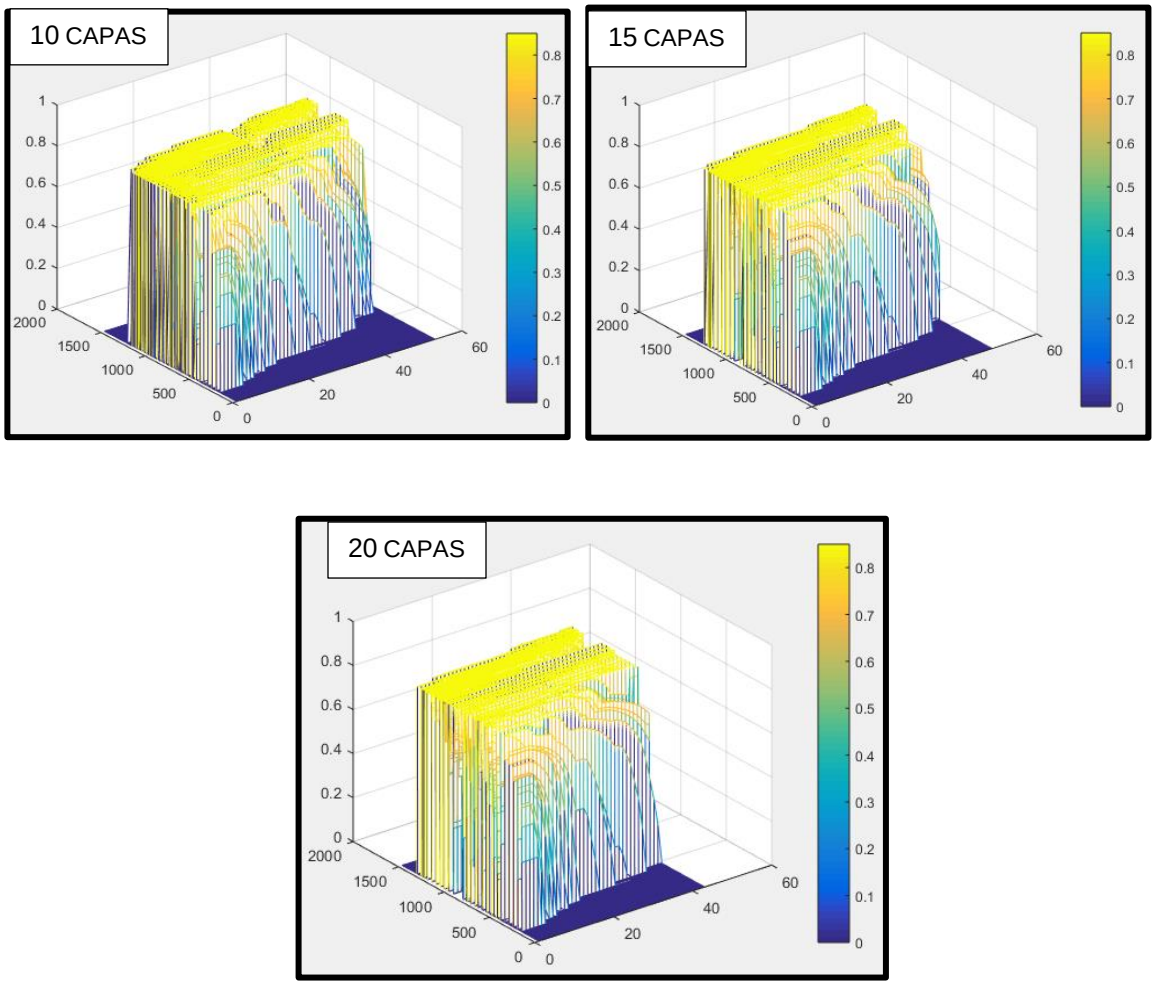
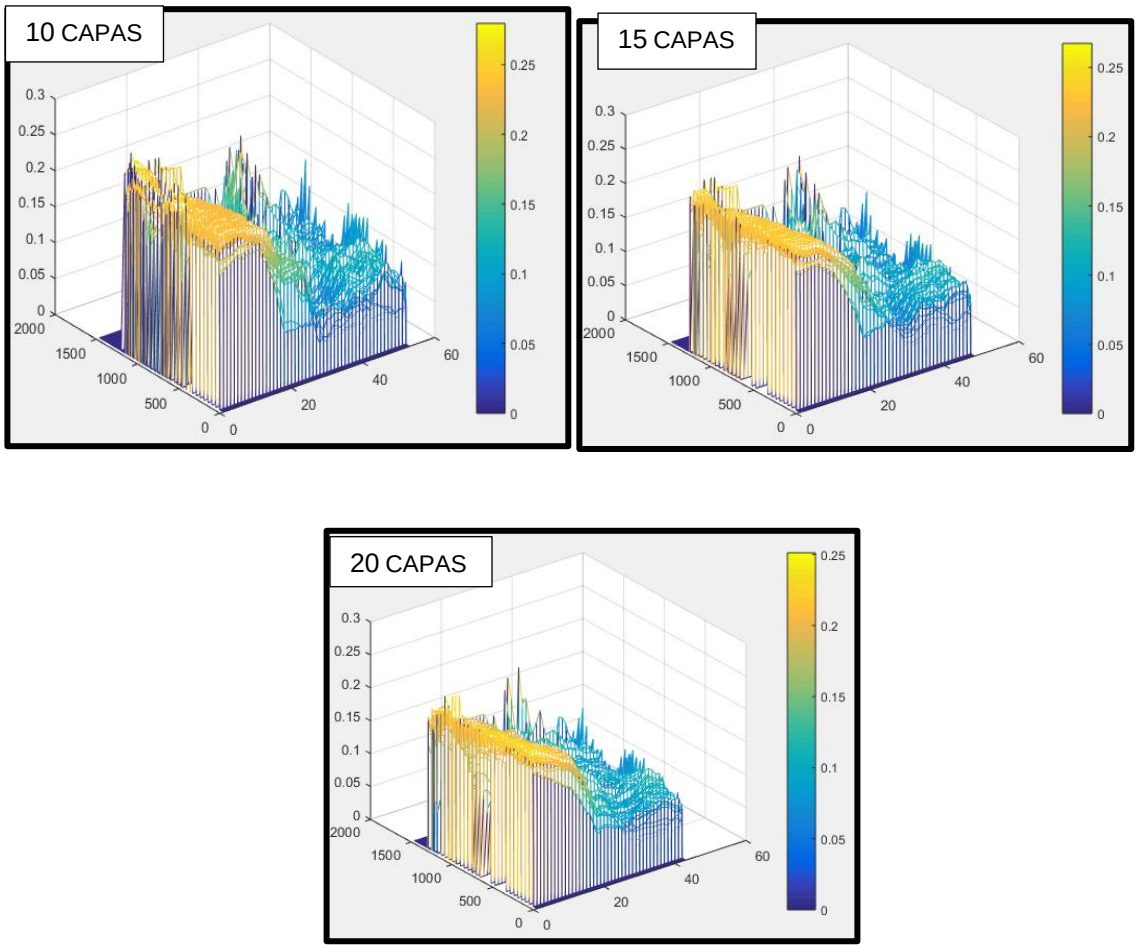


Figura 22. Valores de PI_mean (mD) para intervalo de perforación 10, 15, 20 capas Almacenados en Matlab



4.3. CALCULO DE LOS POTENCIALES.

El cálculo de los diferentes potenciales es una de las principales etapas que conforman esta metodología, ya que la evaluación y determinación de cada uno de ellos involucra en su totalidad el conjunto de variables implícitas en la determinación de las ubicaciones potenciales de los pozos.

Los potenciales son definidos a partir de cada una de las matrices promedios calculada para cada una de las propiedades anteriormente mencionadas. Con cada una de las matrices que almacenan los respectivos promedios de cada propiedad se procede a hacer un producto matricial punto a punto entre estas para hallar el valor del Potencial presentes en cada uno de los intervalos.

El valor numérico del potencial nos ayuda a establecer los diferentes sectores y las diferentes ubicaciones favorables para cada uno de los pozos. A continuación, se presenta la ecuación utilizada para el cálculo del potencial:

$$Potencial = [S_{mean}] * [PI_{Mean}] * [NTG_{Mean}] * \left[K_{efectivaaceite}\right] \quad (14)$$

Para las zonas donde se presenten buenos potenciales es posible afirmar que se debe a una buena magnitud y distribución de las propiedades de Porosidad, Saturación, Net To Gross y Permeabilidad Efectiva al Aceite; caso inverso para cuando se presentan potenciales magnitud numérica baja.

La variación de la magnitud y localización de los potenciales se debe a la distribución espacial de cada una de las propiedades y al número de capas pertenecientes a determinado intervalo de evaluación. A partir del cálculo de los potenciales para los diferentes intervalos de perforación propuestos, es posible observar gráficamente su magnitud y ubicación dentro de las dimensiones del grid del modelo de simulación. Igualmente, los potenciales muestran diferentes magnitudes en cada una de las secciones para un mismo intervalo de perforación, esto se debe a la variación vertical de las propiedades en determinado intervalo de evaluación.

A continuación, se presenta gráficamente la ubicación y magnitud de los potenciales para las diferentes secciones presentes en un intervalo de perforación. Las secciones hacen referencia a número de capas involucrados en el intervalo, es decir, la sección 1 hace referencia a las capas de la uno a la capa 10, la sección 2 hace referencia a la capa 2 hasta la capa 11 como se muestra en la tabla 11.

Figura 23.Comparación de la magnitud y ubicación de los potenciales para el intervalo de perforación de 10 capas.

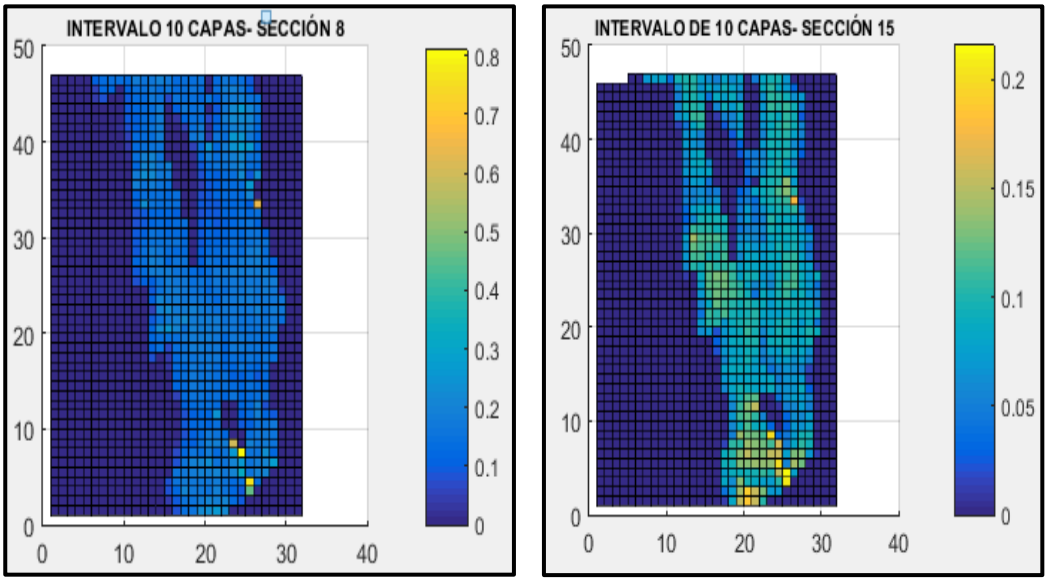


Figura 24. Comparación de la magnitud y ubicación de los potenciales para el intervalo de perforación de 15 capas.

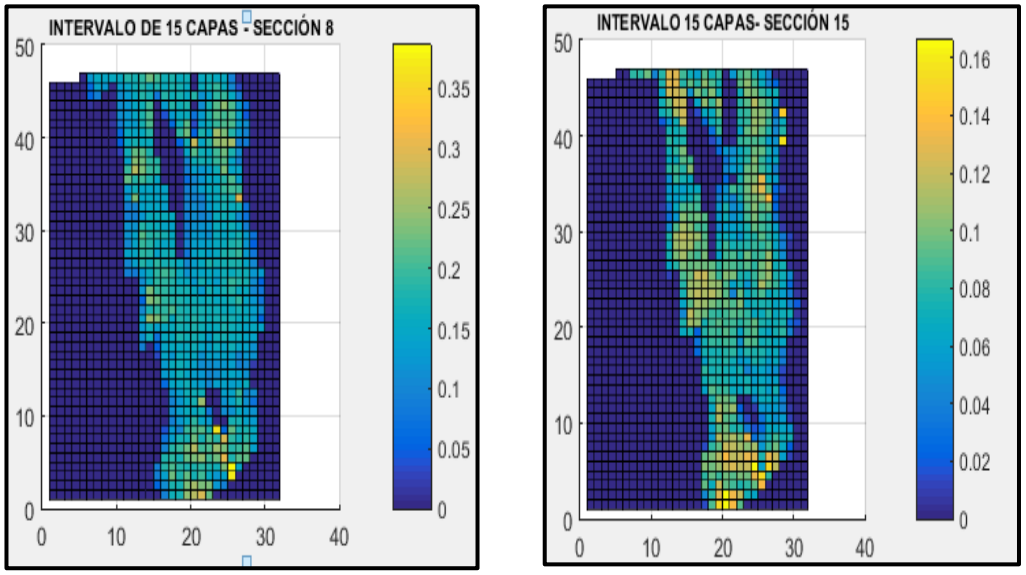
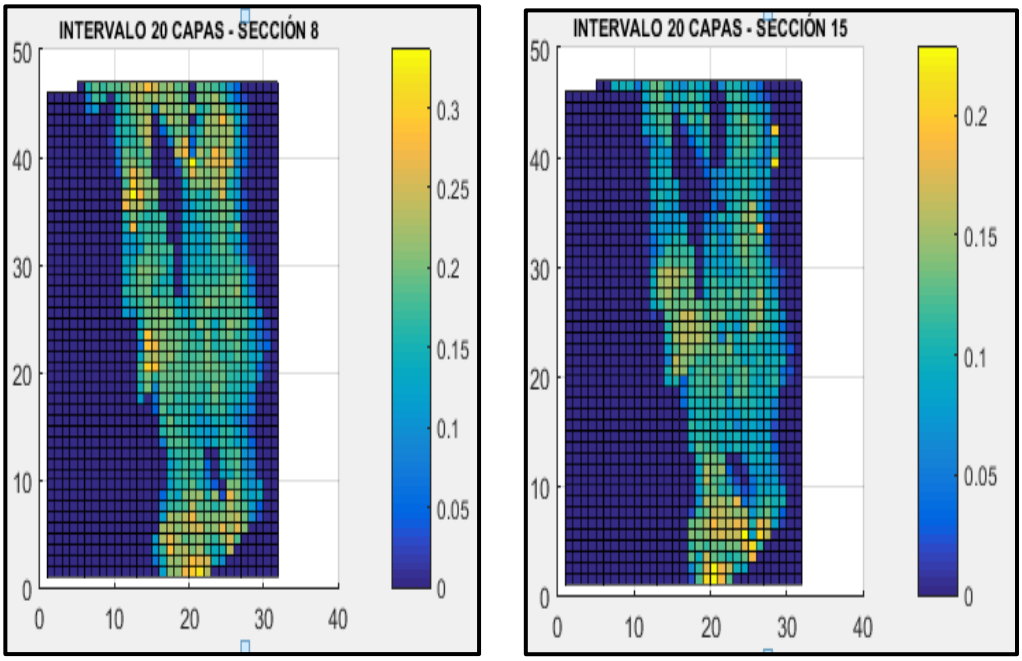


Figura 25. Comparación de la magnitud y ubicación de los potenciales para el intervalo de perforación de 20 capas.



A través del desarrollo de la herramienta es posible identificar un número específico de potenciales presentes en el modelo de simulación; el número de potenciales puede variar según el usuario requiera. De igual forma es posible conocer las coordenadas de ubicación para determinado número de potenciales. En el desarrollo de la metodología se identificaron las diferentes ubicaciones para diferentes potenciales tales como: 100, 200, 300 y 400. La variación del número de potenciales permite observar las principales zonas donde se encuentran ubicados determinados números de potenciales dentro del enmallado de simulación.

La identificación y localización de las zonas con mayor número de ubicaciones potenciales para los pozos, permite realizar una discretización de las posibles ubicaciones que ofrece la totalidad de la dimensión del enmallado de simulación, generando una ventana favorable para una buena y rápida ejecución de la estrategia de ubicación de los pozos.

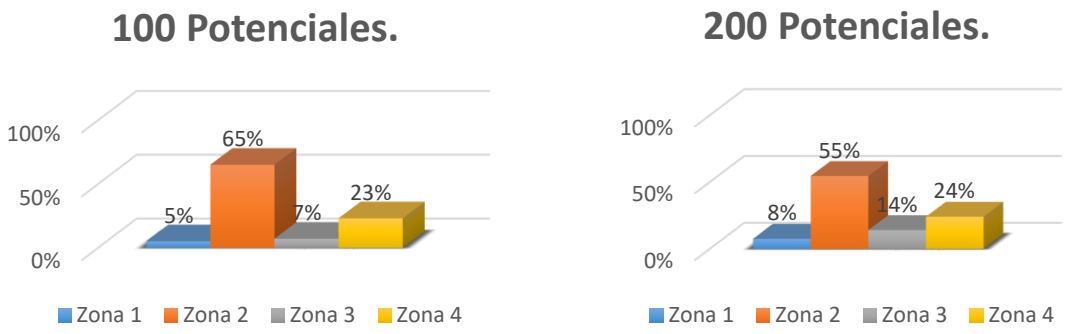
A continuación, se presentan las zonas con mayores ubicaciones potenciales a partir de los resultados obtenidos por la herramienta.

Tabla 15. Zonas con mayor número de ubicaciones potenciales.

COORDENADA	Celdas en X	Celdas en Y
Zona 1	11-16	18-24
Zona 2	17-28	1-13
Zona 3	11-16	31-47
Zona 4	17-28	31-47

La definición de estas zonas se realizó a través de un conteo de todas las ubicaciones (coordenadas) resultantes para determinado número de potenciales, posteriormente cada ubicación fue asignada a su zona correspondiente; La agrupación de las ubicaciones en cada una de sus zonas permite observar la distribución de ubicaciones para cada zona. A continuación, se presenta la distribución de ubicaciones para cada zona y para cada uno de los potenciales.

Figura 26. Distribución de ubicaciones potenciales por zona para intervalos de 10 capas.



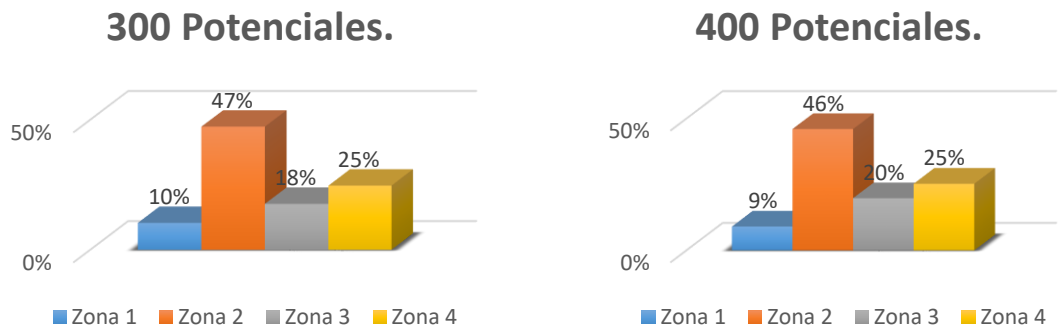
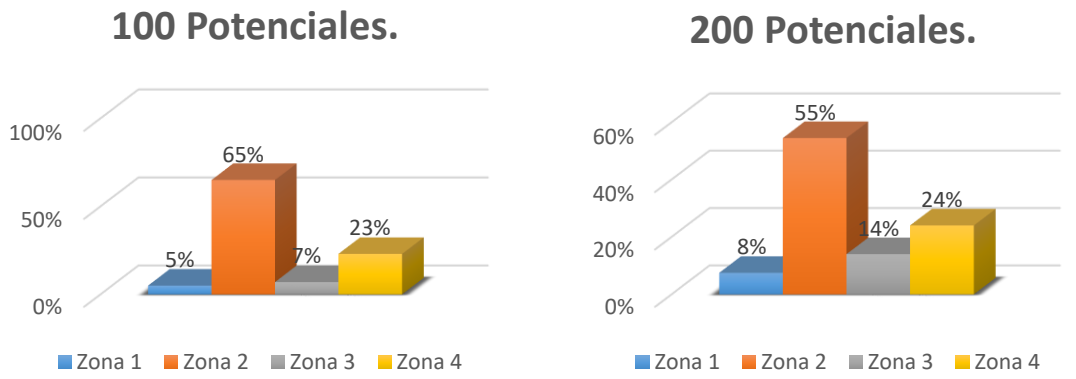


Figura 27. Distribución de ubicaciones potenciales por zona para intervalos de 15 capas.



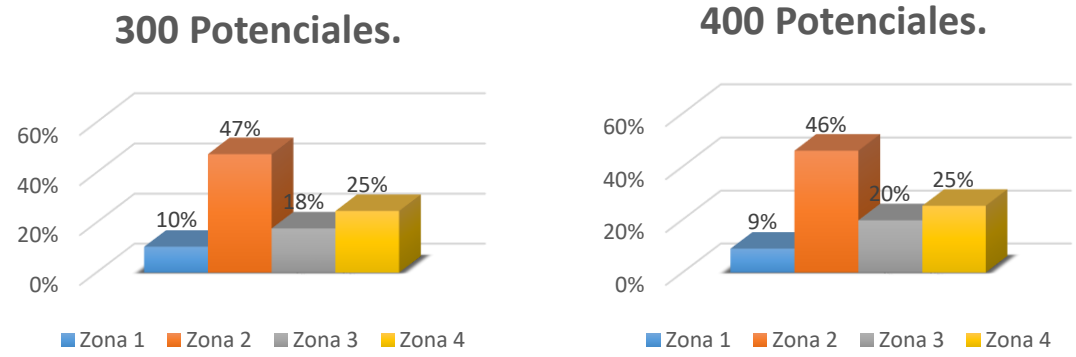
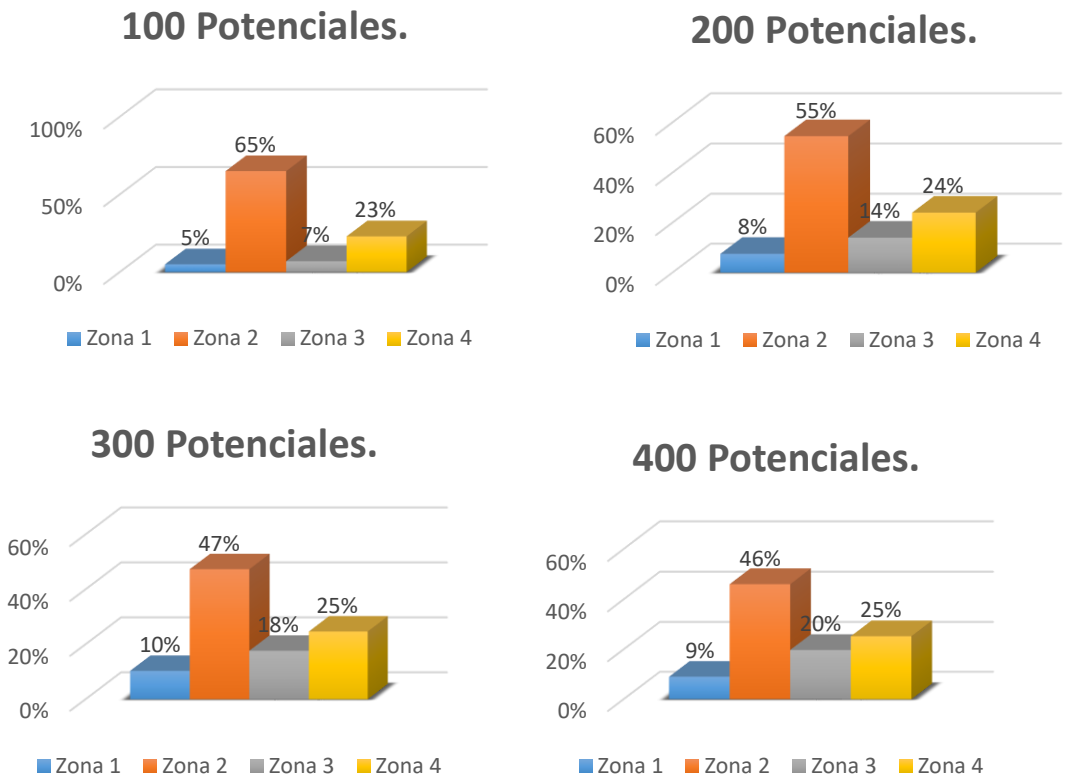


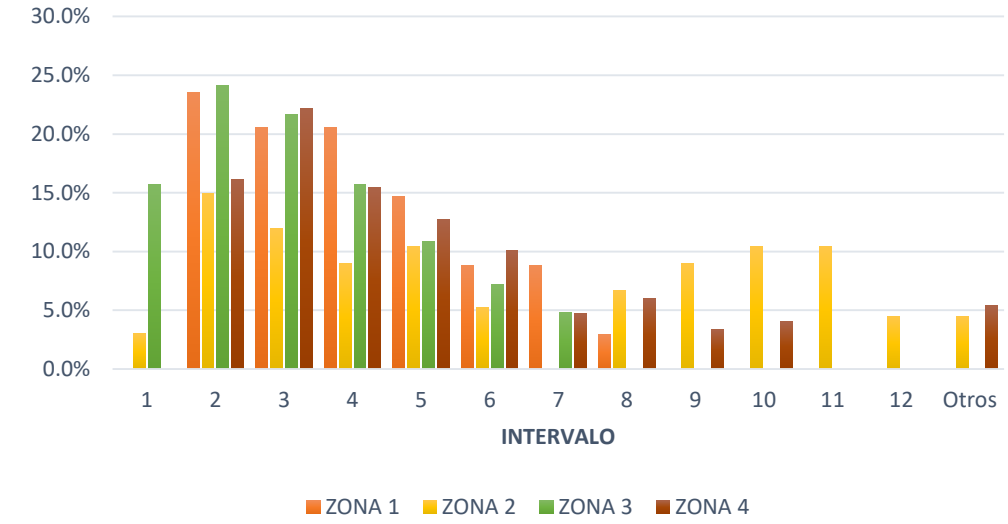
Figura 28. Distribución de ubicaciones potenciales por zona para intervalos de 20 capas.



Con la identificación de la distribucion porcentual de las ubicaciones potenciales en cada una de sus zonas, para cada uno de los potenciales y para cada uno de los intervalos de perforacion propuestos, es posible concluir que las zonas donde se presentan mayor numero de ubicaciones potenciales son la Zona 2 y Zona 4. De esta manera nos ayuda a direccionar la ubicación de los pozos propuestos en el desarrollo de la estrategia.

Con el objetivo de continuar identificando las coordenadas implicitas en la ubicación potencial de los pozos, la aplicación de la herramienta tambien permite identificar los intervalos de perforación más comunes dentro de los intervalos propuestos para el desarrollo de la estrategia. A continuación se presenta la distribución de los intervalos para cada zona.

Figura 29. Distribución de intervalos de perforación potenciales por zona para intervalos de 10 capas



.

Figura 30. Distribución de intervalos de perforación potenciales por zona para intervalos de 15 capas.

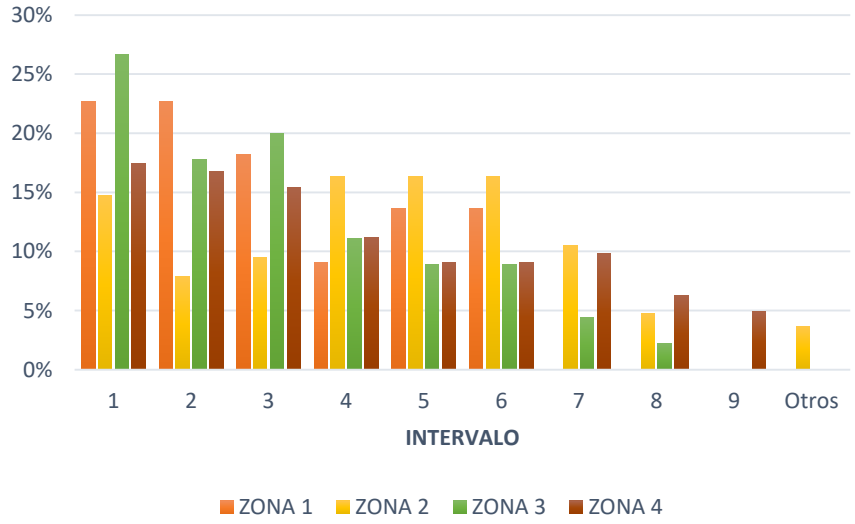
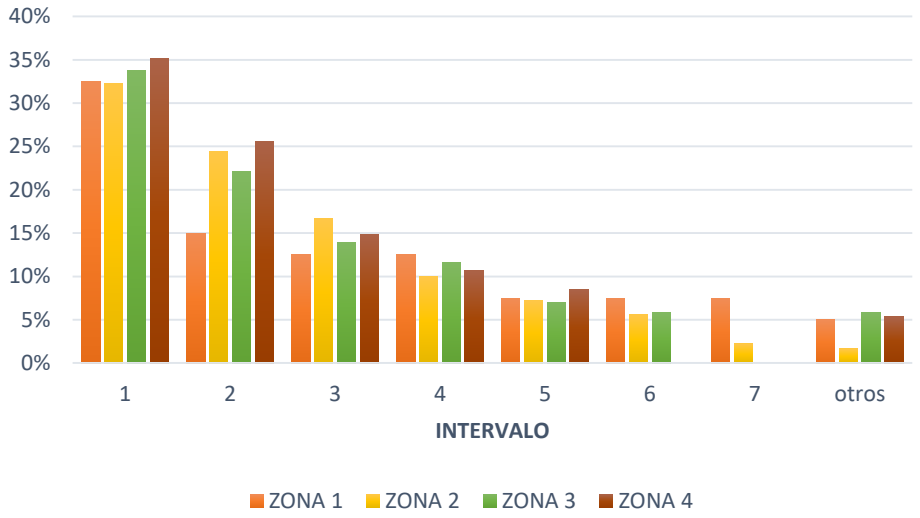


Figura 31. Distribución de intervalos de perforación potenciales por zona para intervalos de 20 capas.



A partir de la identificación de la distribución de intervalo para cada una de las zonas es posible definir una ventana de intervalo para cada una de ellas. Para la sección de 10 capas se presenta intervalos desde el 1 hasta el intervalo número 12, donde los intervalos de mayor frecuencia para esta sección son: 2, 3,4 y 5 con 18, 19, 14 y 12% respectivamente.

Para la sección de 15 capas se presenta intervalos desde el 1 hasta el intervalo número 9, donde los intervalos de mayor frecuencia para esta sección son: 1,2, 3,4 5 y 6 con 17, 13, 13, 13, 12 y 12% respectivamente. Igualmente se definieron los intervalos para la sección de 15 capas los cuales van desde el 1 hasta el intervalo 7, donde los intervalos de mayor frecuencia para esta sección son: 1,2, 3 y 4 con 33, 23, 15 y 10% respectivamente. A pesar de la variación en las secciones de perforación es posible observar que los intervalos de mayor frecuencia en las ubicaciones potenciales para los pozos son 1, 2, 3,4 y 5 para las diferentes zonas.

La ubicación de las diferentes zonas e intervalos donde se encuentran las ubicaciones de mayores potenciales para los pozos ayudan a orientar y direccionar la selección de las ubicaciones donde posteriormente se realizará la evaluación de la función del Valor Presente Neto. La selección de estas coordenadas se realizó de un conjunto de cuatrocientos potenciales, teniendo en cuenta que no se seleccionaran coordenadas iguales o parecidas, de esta manera evitar la interferencia entre pozos ubicados en el enmallado de simulación. A continuación, se presentan las coordenadas seleccionadas con su respectiva sección de perforación.

Tabla 16.Coordenadas seleccionadas para la sección de 10 capas.

10 CAPAS			
POZO #	UBICACIÓN (x,y,z)	ZONA #	PROFUNDIDAD (ft)
1	25, 4, 12	2	10030.323
2	24, 7, 8	2	10110.531
3	21, 3, 2	2	10315.145
4	21, 11, 9	2	10305.604
5	20, 6, 2	2	10338.948
6	24, 40, 2	4	10364.147
7	24, 37, 4	4	10404.272
8	26, 22, 2	4	10364.663
9	26, 12, 3	2	10580.855
10	17, 21, 3	1	10501.729
11	18, 3, 2	2	10613.814
12	21, 37, 2	4	10657.751
13	26, 33, 8	4	10390.949
14	27, 9, 4	2	10637.045
15	15, 20, 2	1	10582.168
16	19, 38, 3	4	10744.953
17	14, 22, 5	1	10798.623
18	11, 36, 1	3	10744.631
19	12, 33, 1	3	10605.641
20	14, 46, 4	3	10731.873
21	18, 40, 5	4	10792.238
22	25, 26, 2	4	10402.453
23	17, 43, 3	4	10713.764

Tabla 17. Coordenadas seleccionadas para la sección de 15 capas.

15 CAPAS			
POZO #	UBICACIÓN (x,y,z)	ZONA #	PROFUNDIDAD (ft)
1	24, 5, 4	2	10087.659
2	23, 8, 4	2	10193.072
3	23, 6, 4	2	10208.224
4	25, 39, 1	4	10384.662
5	24, 43, 1	4	10448.139
6	22, 2, 1	2	10536.623
7	20, 6, 4	2	10564.459
8	17, 21, 1	1	10600.154
9	23, 40, 2	4	10650.926
10	23, 42, 1	4	10613.479
11	21, 11, 3	2	10297.4
12	13, 28, 1	1	10684.111
13	27, 7, 1	2	10677.464
14	26, 34, 5	4	10407.647
15	12, 36, 1	3	10798.085
16	20, 39, 4	4	10897.448
17	17, 4, 1	2	10826.459
18	14, 23, 1	1	10786.612
19	28, 6, 3	2	10702.029
20	14, 20, 1	1	10809.516
21	14, 46, 1	3	10797.488
22	18, 40, 1	4	10825.047
23	26, 36, 3	4	10412.207

Tabla 18. Coordenadas seleccionadas para la sección de 20 capas.

20 CAPAS			
POZO #	UBICACIÓN (x,y,z)	ZONA #	PROFUNDIDAD (ft)
1	26, 22, 2	4	10602.205
2	25, 4, 12	2	10140.957
3	23, 8, 4	2	10208.541
4	21, 11, 3	2	10297.4
5	26, 3, 8	2	10491.588
6	22, 2, 1	2	10536.623

20 CAPAS			
7	21, 3, 2	2	10546.041
8	24, 37, 4	4	10623.453
9	24, 40, 2	4	10624.885
10	17, 21, 1	1	10600.154
11	26, 34, 5	4	10438.904
12	23, 40, 2	4	10650.926
13	23, 42, 1	4	10613.479
14	24, 43, 1	4	10448.139
15	27, 7, 1	2	10677.464
16	18, 3, 2	2	10846.907
17	21, 37, 2	4	10909.307
18	12, 33, 1	3	10856.596
19	28, 6, 3	2	10702.029
20	12, 36, 1	3	10798.085
21	17, 4, 1	2	10924.885
22	19, 38, 3	4	11007.098
23	11, 36, 1	3	11007.098

El limite de perforación fue establecido hasta una sección de 20 capas, a pesar de que la herramienta es posible simular para intervalos mayores. La selección de este límite se debe a que para intervalos mayores a 20 capas las ubicaciones potenciales son muy similares a las de 20 capas e igualmente para intervalos mayores a 20 es posible observar una disminución del Valor presente Neto para la misma ubicación perforada en una sección de 20 capas y mayores a 20 capas, esto se debe a que para esas ubicaciones potenciales cuando se perfora secciones mayores a 20 capas el corte de agua aumenta súbitamente por la cercanía de punto base de perforación y el contacto agua aceite.

Finalmente se seleccionaron 23 ubicaciones potenciales de las 400 arrojadas por la herramienta para cada una de las secciones de perforación, fue necesario conocer 400 ubicaciones potenciales debido a que dentro del conjunto de las 400 ubicaciones se presentan ubicaciones potenciales de igual coordenadas (x, y) pero diferentes intervalos lo que reduce la selección de las ubicaciones potenciales.

4.3.1. Evaluación de valor presente neto: La evaluación del Valor Presente Neto está sujeto a un conjunto de factores tales como: Producción de Aceite y Agua, Parámetros económicos y condiciones operativas establecidas para el desarrollo de la estrategia.

Inicialmente para realizar la evaluación del Valor Presente Neto se seleccionaron las ubicaciones de los pozos para su respectiva evaluación económica, las ubicaciones seleccionadas son las mostradas en la tabla 15, 16 y 17 para secciones de 10,15 y 20 capas respectivamente; cada una de ella con un conjunto de 23 pozos. La determinación del Valor Presente Neto se realizó pozo a pozo, es decir, se realizó una corrida de simulación para cada uno de los pozos sin la presencia de otro pozo en el enmallado de simulación hasta completar las 23 ubicaciones potenciales seleccionadas.

Al finalizar cada una de las simulaciones se exportaron los datos de producción acumulada mensual de Aceite y Agua para cada para cada pozo potencial y se almacenaron en archivos de Excel. Los diferentes cálculos inmersos en la evaluación del Valor Presente Neto se realizaron en Excel debido a que en el desarrollo de la herramienta fue propuesto un módulo de lectura de estos archivos y no un módulo de cálculo directo por parte de la herramienta, por lo tanto, fue necesario realizar estos cálculos con la ayuda de Excel. La exportación de los datos de producción mensual acumulada se realizó a través de la ejecución de un archivo .RWD (anexo A) en la herramienta Result Report de Launcher. Los parámetros económicos establecidos para la evaluación del Valor Presente Neto se muestran en la siguiente tabla:

Tabla 19. Parámetros establecidos para el cálculo de Valor Presente Neto.

INDICADORES ECONOMICOS			GASTOS OPERATIVOS [USD/STB]		
REGALIAS	8%	Mensual	Tratamiento químico Crudo		5
IMPUESTO DE RENTA	20%	Mensual	Tratamiento Químico Agua		2.8
INFLACIÓN	5%	Anual	Lifting Cost		12
INFLACIÓN	0.41%	Mensual	Otros Gastos		4.5
iOPORTUNIDAD	12%	Anual	Costo de Perforación.	820	USD/ft
i OPORTUNIDAD	0.95%	Mensual	Amortización Contable	0.144	MMUSD
PRECIO DEL BARRIL	45	US$/bbl			

El proceso de evaluación económica desarrollado en Excel se inició con el cálculo de los ingresos, posteriormente se calcularon todos los gastos operativos o egresos los cuales incluyen las perforaciones de los pozos, los tratamientos químicos del aceite y agua producidos y el Lifting Cost; posteriormente se calculó el flujo de caja mensual para cada pazo con el objetivo final de calcular el Valor Presente Neto (VPN) para cada uno de los pozos a través de la siguiente ecuación:

$$VPN = \sum_{n=0}^{t} \frac{Fujo\ de\ Caja_t}{(1+td)^t}$$

Donde;

$Fujo\ de\ Caja = Ingresos - Egresos.$

$Ingresos = P^o(t) * {q_o}^{produciodo}(t).$

$Egresos = {C_w}^{prod}(t) * {q_w}^{prod}(t) + {C_o}^{prod}(t) * {q_o}^{prod}(t).$

$td = Tasa\ de\ Interes.$

$t = Tiempo\ de\ Evalución\ del\ Proyecto\ (12\ años)$

La evaluación del Valor Presente Neto se realizó a diferentes escenarios de condiciones operativas de los pozos, es decir, se realizó el cálculo del Valor Presente Neto para diferentes restricciones de tasa de líquido en superficie (STL); con el objetivo de establecer una conjugación favorable de la zona, la sección de perforación y la restricción operativa del pozo para establecer el mejor escenario de localización y condiciones operativas para los pozos que conlleve a un desarrollo favorable de la estrategia de ubicación.

Las tasas de líquido en superficie (STL) evaluadas son 8000 Bbl/Dia, 10000 Bbl/Dia y 12000 Bbl/Dia, el criterio de selección de estas tasas se debe a las condiciones operativas de las facilidades de superficie disponibles que presenta el campo simulado en el modelo, de igual manera se estableció la variación de estas tasas a juicio ingenieril para observar el comportamiento de la producción de agua y aceite respecto a la variación de las tasas para cada uno de los pozos, lo cual concluyo que la producción de aceite y agua presentaron un comportamiento proporcional al aumento de las tasas. Igualmente, el comportamiento del Valor Presente Neto fue proporcional al aumento de las tasas (TSL).

La determinación del Valor Presente Neto para las diferentes tasas establecidas se realizó a través de la simulación pozo a pozo para cada una de las tasas, lo cual estableció 207 corridas de simulación en su totalidad. A continuación, se presenta la variación del Valor Presente Neto para cada una de las tasas en la respectiva ubicación de cada uno de los pozos.

Tabla 20. Evaluación de Valor Presente Neto para sección de 10 capas.

10 CAPAS						
POZO #	UBICACIÓN (x,y,z)	ZONA #	PROFUNDIDAD (ft)	VALOR PRESENTE NETO (MMUS$)		
				TASA 8000 (bpd)	TASA 10000(bpd)	TASA 12000(bpd)
1	25, 4, 12	2	10030.323	329.993	417.285	504.576
2	24, 7, 8	2	10110.531	329.840	417.13	504.28
3	21, 3, 2	2	10315.145	329.44	416.74	504.03
4	21, 11, 9	2	10305.604	329.467	416.758	503.904
5	20, 6, 2	2	10338.948	329.403	416.694	503.986
6	24, 40, 2	4	10364.147	329.355	416.646	503.938
7	24, 37, 4	4	10404.272	329.279	416.570	503.862
8	26, 22, 2	4	10364.663	329.354	416.646	501.804
9	26, 12, 3	2	10580.855	322.463	401.172	461.894
10	17, 21, 3	1	10501.729	328.885	398.768	449.207
11	18, 3, 2	2	10613.814	309.652	369.689	421.393
12	21, 37, 2	4	10657.751	281.376	336.087	370.229
13	26, 33, 8	4	10390.949	318.516	331.511	331.512
14	27, 9, 4	2	10637.045	255.914	269.174	277.487
15	15, 20, 2	1	10582.168	235.641	253.105	264.409
16	19, 38, 3	4	10744.953	169.871	202.139	215.829
17	14, 22, 5	1	10798.623	149.389	177.970	205.124
18	11, 36, 1	3	10744.631	149.431	177.911	204.913
19	12, 33, 1	3	10605.641	180.985	180.985	180.985
20	14, 46, 4	3	10731.873	119.970	134.993	147.858
21	18, 40, 5	4	10792.238	81.328	98.903	116.603
22	25, 26, 2	4	10402.453	105.114	105.114	105.114
23	17, 43, 3	4	10713.764	72.764	72.761	72.761

Tabla 21. Evaluación de Valor Presente Neto para sección de 15 capas.

15 CAPAS						
POZO #	UBICACIÓN (x,y,z)	ZONA #	PROFUNDIDAD (ft)	VALOR PRESENTE NETO (MMUS$)		
				TASA 8000 (bpd)	TASA 10000 (bpd)	TASA 12000 (bpd)
1	24, 5, 4	2	10087.659	329.884	417.175	504.467
2	23, 8, 4	2	10193.072	329.682	416.974	504.265
3	23, 6, 4	2	10208.224	329.653	416.945	504.236
4	25, 39, 1	4	10384.662	329.316	416.608	503.899
5	24, 43, 1	4	10448.139	329.195	416.423	500.864
6	22, 2, 1	2	10536.623	329.026	415.254	496.878
7	20, 6, 4	2	10564.459	328.908	414.463	496.518
8	17, 21, 1	1	10600.154	327.321	407.070	481.400
9	23, 40, 2	4	10650.926	325.808	405.552	479.832
10	23, 42, 1	4	10613.479	322.675	396.781	465.116
11	21, 11, 3	2	10297.4	329.483	410.033	451.674
12	13, 28, 1	1	10684.111	310.174	375.865	434.865
13	27, 7, 1	2	10677.464	280.360	342.635	388.296
14	26, 34, 5	4	10407.647	322.961	345.220	345.215
15	12, 36, 1	3	10798.085	238.454	283.347	324.281
16	20, 39, 4	4	10897.448	201.968	246.026	287.576
17	17, 4, 1	2	10826.459	187.861	221.454	251.094
18	14, 23, 1	1	10786.612	177.724	213.255	247.178
19	28, 6, 3	2	10702.029	197.247	202.237	204.915
20	14, 20, 1	1	10809.516	150.100	173.660	194.842
21	14, 46, 1	3	10797.488	108.606	123.955	137.114
22	18, 40, 1	4	10825.047	70.535	88.240	105.628
23	26, 36, 3	4	10412.207	87.532	87.532	87.532

Tabla 22. Evaluación de Valor Presente Neto para sección de 20 capas.

20 CAPAS						
POZO #	UBICACIÓN (x,y,z)	ZONA #	PROFUNDIDAD (ft)	VALOR PRESENTE NETO (MMUS$)		
				TASA 8000 (bpd)	TASA 10000(bpd)	TASA 12000(bpd)
1	26, 22, 2	4	10602.205	328.900	416.192	509.111
2	25, 4, 12	2	10140.957	329.782	417.073	504.365
3	23, 8, 4	2	10208.541	329.653	416.944	504.236
4	21, 11, 3	2	10297.4	329.483	416.774	504.066
5	26, 3, 8	2	10491.588	329.112	416.403	503.337
6	22, 2, 1	2	10536.623	329.026	416.317	503.331
7	21, 3, 2	2	10546.041	329.008	416.247	502.489
8	24, 37, 4	4	10623.453	328.347	413.868	497.901
9	24, 40, 2	4	10624.885	328.415	413.573	496.414
10	17, 21, 1	1	10600.154	320.714	398.633	471.497
11	26, 34, 5	4	10438.904	329.212	409.197	466.719
12	23, 40, 2	4	10650.926	316.859	392.879	464.067
13	23, 42, 1	4	10613.479	307.750	376.522	439.949
14	24, 43, 1	4	10448.139	308.856	377.609	426.643
15	27, 7, 1	2	10677.464	256.792	316.210	374.949
16	18, 3, 2	2	10846.907	246.364	298.739	345.237
17	21, 37, 2	4	10909.307	216.639	266.459	314.131
18	12, 33, 1	3	10856.596	210.718	252.619	291.106
19	28, 6, 3	2	10702.029	218.760	266.941	287.643
20	12, 36, 1	3	10798.085	192.712	233.854	271.492
21	17, 4, 1	2	10924.885	136.643	166.179	192.758
22	19, 38, 3	4	11007.098	62.521	80.176	98.164
23	11, 36, 1	3	11007.098	62.521	80.176	98.164

Al finalizar la evaluación de los parámetros económicos y operativos para cada una de las ubicaciones potenciales en su respectiva sección de perforación, es posible observar el comportamiento del Valor Presente Neto según la zona en la cual este ubicado determinado pozo. El Valor Presente Neto es el criterio más importante en la selección de la ubicación (x,y) y del intervalo de perforación en el cual se ubicará el pozo, ya que de esta forma se seleccionan los pozos con mayor Valor Presente Neto para ser aplicados en el desarrollo de la estrategia y así lograr el escenario más favorable económicamente para el proyecto.

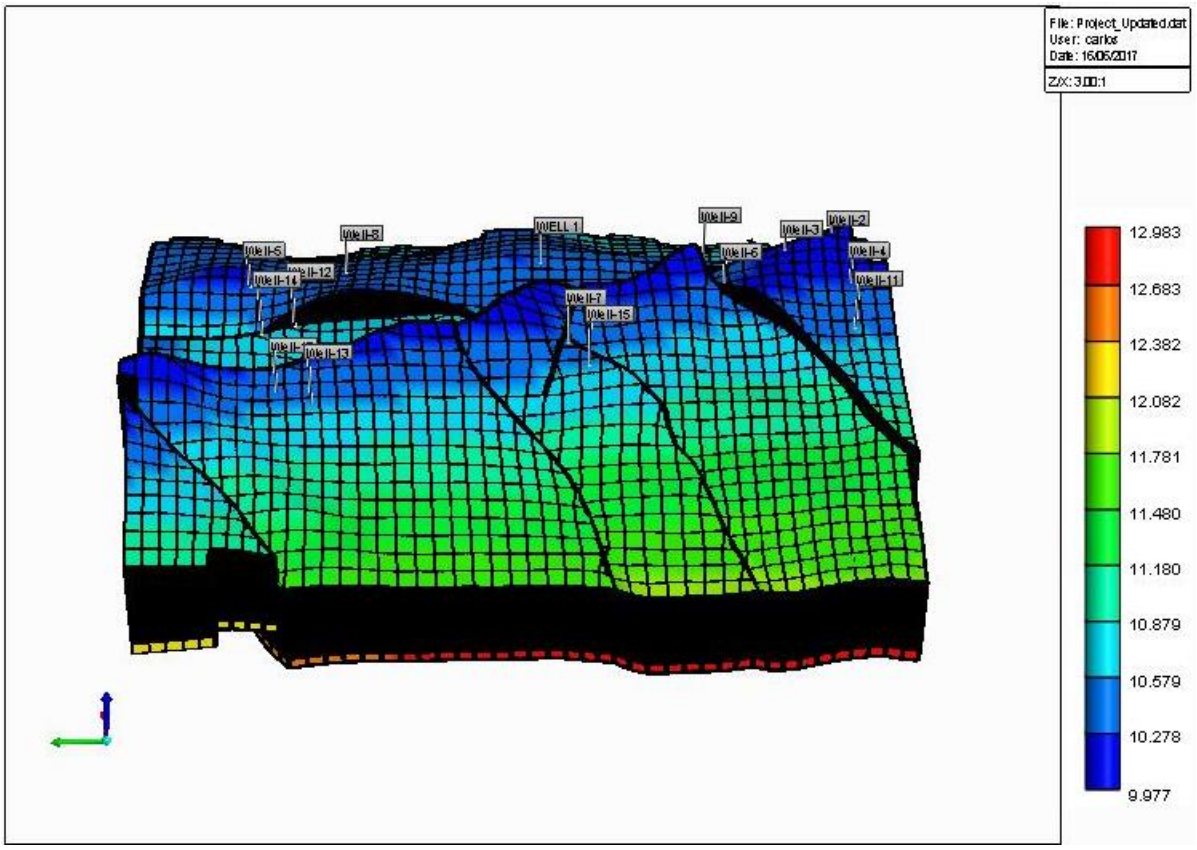
Tabla 23. Pozos seleccionados para la estrategia de ubicación.

ESTRATEGIA DE SELECCION			
POZO	UBICACIÓN (x,y,z)	ZONA	INTERVALO DE PERFORACION
1	26, 22, 2	4	20
2	25, 4, 12	2	10
3	24, 7, 8	2	10
4	21, 3, 2	2	10
5	24, 40, 2	4	10
6	21, 11, 3	2	20
7	17, 21, 1	1	15
8	26, 34, 5	4	20
9	26, 12, 3	2	10
10	13, 28, 1	1	15
11	18, 3, 2	2	10
12	21, 37, 2	4	10
13	12, 36, 1	3	15
14	20, 39, 4	4	15
15	15, 20, 2	1	10

La selección de los 15 pozos con mayores Valor Presente Neto permite el desarrollo de la estrategia de ubicación en el escenario más favorable económicamente para la aplicación de la misma. En la evaluación del Valor Presente Neto de forma particular, es decir, pozo a pozo los efectos de depleción de la presión de yacimiento son minúsculos comparados con los efectos de depleción de la presión del yacimiento para cuando se ubican los 15 pozos de la estrategia; debido a que la razón de drenaje pozo a pozo de los fluidos es muy pequeña respecto a la totalidad presente en el modelo de simulación, sin embargo, la evaluación particular del Valor Presente Neto nos ayuda a conocer los pozos de mejor potencial productivo y mejor escenario económico en el tiempo de evaluación de la estrategia.

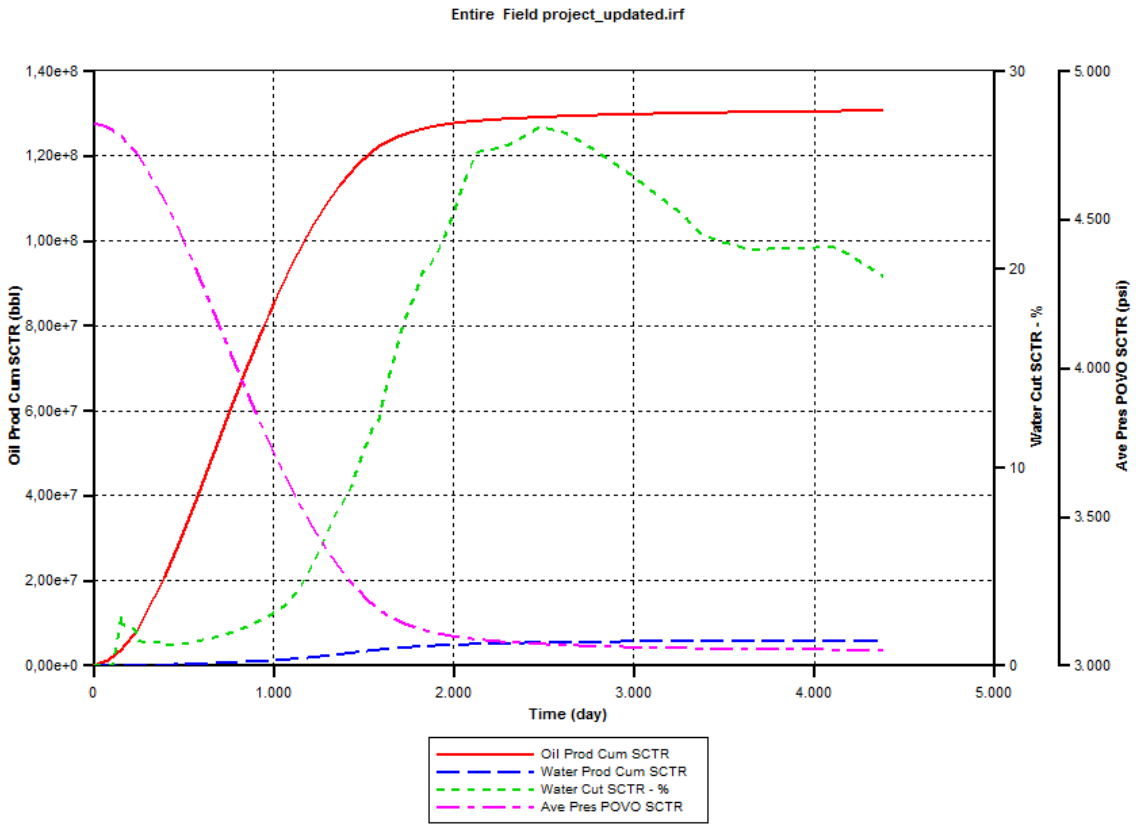
Luego de seleccionar los pozos que presentan el mejor escenario económico se realiza la simulación para cuando se han ubicado los 15 pozos productores verticales propuestos por la estrategia. El desarrollo de estos pozos está sujeto a la disponibilidad operativa de los equipos de perforación y movilización con los que cuenta el campo. El campo cuenta con dos taladros de perforación, que tienen una razón de perforación de $165\ ft/dia$. Igualmente, la capacidad de movilización de los equipos de perforación en el campo es de 15 días. Estos factores determinan las fechas de cierre y apertura de los pozos según el desarrollo de las perforaciones. Posteriormente de haber establecido el programa de perforación de los 15 pozos es posible realizar la simulación para el conjunto de los pozos en sus respectivas ubicaciones. A continuación, se presenta la ubicación de los 15 pozos en el enmallado de simulación.

Figura 32. Pozos Seleccionados para la estrategia.



La imagen presenta la ubicación final de los 15 pozos seleccionados para el desarrollo de la estrategia. De esta manera es posible realizar la ejecución de la simulación para este escenario. La finalización de la simulación permite observar el comportamiento productivo y otras variables que se presentan para estas ubicaciones. A continuación, se presenta el comportamiento de la Producción Acumulada de Aceite y agua, el Corte de agua y la Presión promedio del yacimiento para cuando la estrategia es aplicada.

Figura 33. Comportamiento de Variables de Campo.

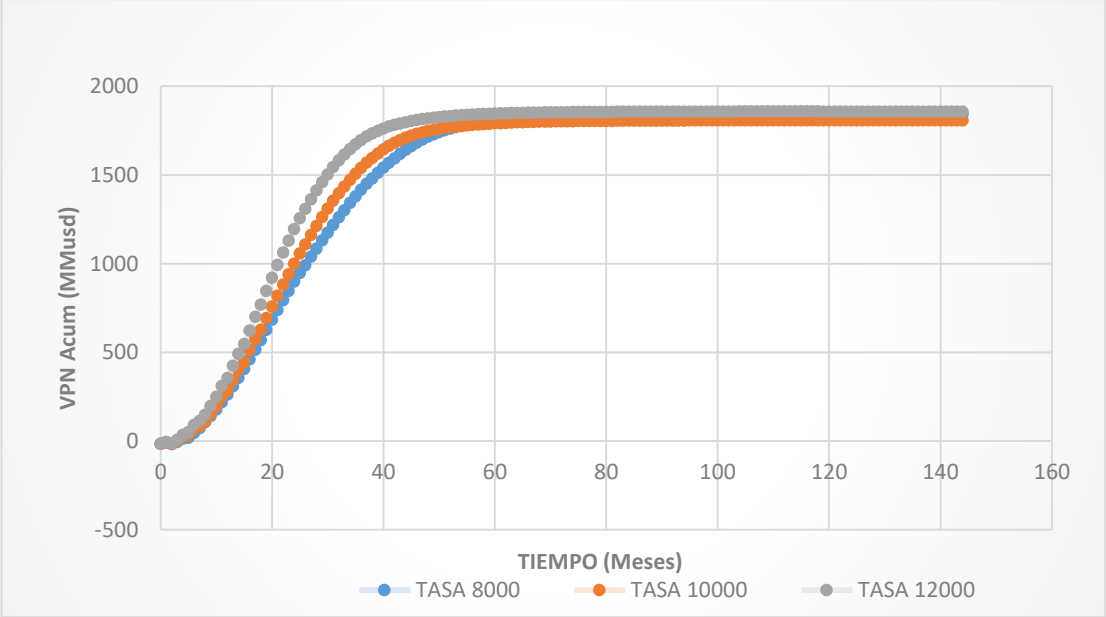


El comportamiento de las diferentes graficas observadas en el plot muestra que: la producción acumulada de aceite presenta un crecimiento lineal aproximadamente los primeros 2000 días de simulación, luego de estos 2000 días presenta aumentos muy pequeños y paulatinos por lo que presenta una línea horizontal. El comportamiento de la producción acumulada de aceite posterior a los 2000 días es descrito por una línea horizontal debido a que para ese tiempo la Presión promedio del yacimiento es de 3100 psi. Para esta presión promedio de yacimiento el diferencial de presión en fondo de pozo es de 285 psi, respecto al BHP de 2815 psi establecida previamente; la magnitud pequeña de este diferencial de presión disminuye y limita la capacidad de producción de cada uno de los pozos causando una directa disminución de su producción y de esa forma disminuyendo el aumento lineal de la producción acumulada de aceite. Igualmente, el plot muestra se obtiene

una baja producción de agua durante el tiempo de evaluación, generando cortes de agua máximos de aproximadamente de 27 %.

Igualmente, con la finalización de la simulación es posible realizar la evaluación económica para la ubicación de estos 15 pozos en el modelo de simulación. A continuación, se presenta la evaluación económica para la estrategia.

Figura 33. Valor Presente Neto Acumulado para las tres Tasas simuladas.



La figura 35 presenta el comportamiento del Valor Presente para cada una de las tasa de líquido simuladas, es posible observar que el aumento que presenta el Valor Presente Neto Acumulado para los primeros 50 meses aproximadamente es proporcional al aumento de las tasa y para meses posteriores el aumento en el Valor Presente Neto Acumulado es bajo y esto es causado porque a partir de esta fecha se presenta una disminución en la producción acumulada del campo para las diferentes tasas por la depleción de la presión del yacimiento causada por el drenaje de sus fluidos (Aceite y Agua). A continuación, se presenta la tabla del Valor Presente Acumulado al final del tiempo de aplicación de la estrategia.

Tabla 24. Valor Presente Neto Acumulado Final.

VALOR PRESENTE NETO DEL CAMPO			
NUMERO DE POZOS	VALOR PRESENTE NETO (MMUS$)		
15	TASA 8000 bbl/día	TASA 10000 bbl/día	TASA 12000 bbl/día
	1589.739061	1646.953416	1696.677462
TIR (%)	18	21	23

5. CONCLUSIONES.

- La implementación de la metodología propuesta en el presente trabajo permite economizar importante tiempo de cómputo en modelos de simulación numérica, debido a que permite direccionar la ubicación de los pozos en las zonas con mayor potencial de almacenamiento y flujo de los fluidos, eliminado el uso de simulaciones realizadas aleatoriamente para conocer estas ubicaciones.

- El desarrollo de la metodología propuesta permite definir zonas potenciales para ubicación de pozos productores. Para este caso las zonas más potenciales son: Zona 2 y Zona 4, con un porcentaje de pozos ubicados de 40 y 35 % respectivamente.

- El desarrollo de la metodología propuesta permite definir intervalos de perforación más potenciales para ubicación de pozos productores. Para este caso los intervalos de perforación más potenciales son: 10 capas, 15 capas y 20 capas, con un porcentaje de pozos ubicados en esos intervalos 60, 27 y 13% respectivamente.

- El almacenamiento de cada una de las propiedades en Matlab generó una representación muy similar y compatible respecto a los datos almacenados en el modelo base de simulación del software CMG, lo que hace posible realizar la conjugación de ambas herramientas informáticas para una correcta evaluación de la función objetivo.

- La distribución espacial y la magnitud de las cuatro propiedades: Porosidad, Permeabilidad efectiva al Aceite, Saturación de Aceite y Net to Gross. Es uno de los principales factores que definen el performance de la función objetivo

evaluado y por lo tanto son una directriz de la ubicación de los pozos dentro del modelo base de simulación.

- Las magnitudes de los potenciales calculados en Matlab representan que las capas o layers con las mejores distribuciones y magnitudes, se encuentran ubicadas entre los intervalos 2 y 17 lo que corresponde a las capas número 2 hasta el número 2, afirmado gráficamente por CMG.

- Los intervalos de perforación de máximos potenciales arrojados por la herramienta se encuentran a profundidades superiores al contacto Agua-Aceite, lo cual genera un escenario favorable económicamente para el desarrollo de la estrategia.

6. RECOMENDACIONES.

- Aplicar la metodología presentada realizando la simulación de los pozos en las ubicaciones en conjunto, es decir, que no se realice pozo a pozo si no que sea aplicada para todo el conjunto de pozos que se quiera ubicar; así observar el efecto sobre el Valor Presente Neto de cada pozo y campo.

- Aplicar la metodología de la ubicación de pozos productores verticales desarrollada en este trabajo, presentando diferentes escenarios donde se tenga en cuenta diferentes precios del crudo, para así generar una comparación del escenario presentado por esta metodología y los otros posibles escenarios a desarrollar.

- Desarrollar mediante otro proyecto de grado la evaluación del Valor presente Neto incluyendo el costo de instalación y diseño de las facilidades de superficie necesarias para el transporte y manejo de los volúmenes de Aceite y Agua producidos, a partir de las ubicaciones seleccionadas con el desarrollo de la metodología propuesta previamente.

BIBLIOGRAFÍA

ABUKHAMSIN, Ahmed Y. *"Optimization of Well Design and Location in a Real Field".* Testis de Maestro. Universidad de Stanford. 2009.

ADENIJI, Adewale. IDIGBE, K.I. *"Fast Algorithm for Calculating Present Values and Rate of Return on Investment"* .SPE 172437. 2014.

ARCHER, Rosalind Ann. ZAKERI, Golbon. VAUDREY, Tobi. Cogita. *Splines as an Optimization Tool in Petroleum Engineering.* SPE 95601. 2005.

ASADOLLAHI, Masoud. NAEVDAL, Geir. *Waterflooding Optimization Using Gradient Based Methods.* SPE 125331. 2009.

AVANSI, Guilherme Daniel. SCHIOZER, Denis Jose. SUSLICK, Saul. RISSO, Fernanda Vaz. *Assinted Procedures for Definition of Production Strategy and Economic Evaluation Using Proxy Models.* SPE 122298. 2009.

AWOTUNDE, Abeeb A. NARANJO, Carlos. W*ell Placement Optimization Constrained to Minimum Well Spacing*. SPE 169272. 2014.

B.L. BECKNER *et al. Field Development Planning Using Simulated Annealing - Optimal Economic Well Scheduling and Placement.* 1995.

BADRU, O. KABIR, C.S. *"Well Placement Optimization in Field Development".*SPE 84191. 2003.

BITTENCOURT, Antonio C. HOME, Roland N. *"Reservoir Development and Design Optimization".* SPE 38895. 1997.

DING, Yu. *Optimization of Well Placement Using Evolutionary Algorithms.* SPE 113525. 2008.

EMERICK, Alexandre A. SILVA, Eugênio. MESSER, Bruno. ALMEIDA, Luciana F. SZWARCMAN, Dilza. PACHECO, Marco Aurelio C. VELLASCO, Marley M.B.R. *"Well Placement Optimization Using a Genetic Algorithm with Nonlinear Constraints"* SPE 118808. 2009.

ERMOLAEV, Alexander. KUVICHKO, Alexander. *"Non-Regular Well Placement Optimization"* SPE 166903. 2013.

FOROUZANFAR, Fahim*. LI, Gaoming. REYNOLD, Albert.* "*A Two-Stage Well Placement Optimization Method Based on Adjoint Gradient". SPE 135304.* 2010.

FARSHI, Mohammand Moravvej. *Improving Genetic Algorithms for Optimum Well Placement*, Master's Report, Department of Energy Resources Engineering, Stanford University, California. 2008.

FOROUZANFAR, Fahim. LI, Gaoming. REYNOLDS, Albert Coburn. "*A Two-Stage Well Placement Optimization Method Based on Adjoint Gradient"* SPE 135304. 2010.

GÜYAGÜLER, Baris. *Optimization of Well Placement and Assessment of Uncertainty.* Tesis de Doctorado. Universidad de Stanford. 2002.

JESMANI, Mansoureh. BELLOUT, Mathias C. HANEA, Remus. FOSS, Bjarne. *Particle Swarm Optimization Algorithm for Optimum Well Placement Subject to Realistic Field Development Constraints.* SPE 175590. 2015.

LE RAVALEC, Mickaele. *Optimizing Well Placement with Quality Maps Derived From Multi-Fidelity Meta-Models.* SPE 154416. 2012.

LEE, Tai-Yong. KRAVARIS, Costas. SEINFELD, Jhon H. *History Matching by Spline Approximation and Regularization in Single-Phase Area Reservoirs.* SPE 13931. 1986.

MANSOUREH Jesmani. MATHIAS C. Bellout et al. *Particle Swarm Optimization Algorithm for Optimun well placement subject to Realistic Field Development constrains.* SPE 175590. 2015

MORALES Adrian Nicolas. NASRABADI, Hadi. ZHU, Ding. *A New Modified Algorithm for Well Placement optimization under Geological uncertainties.* SPE 143617. 2011

OZDOGAN, Umut. HORNE, Roland N. *Optimization of Well Placement Under Time-Dependent Uncertainty.* SPE 90091. 2004.

PAN, Yan. HORNE, Roland*. Improved Methods for Multivariate Optimization of Field Development Scheduling and Well Placement Desing. SPE 49055. 1998*

SARMA, Pallav. CHEN, Wen H. *"Efficient Well Placement Optimization with Gradient-Based Algorithms and Adjoint Models.* SPE 112257. 2008.

UNIVERSIDAD DE LOS ANDES. "*Tutorial De Matlab".* (en línea), 06 de Marzo de 2017. Disponible en Internet: https://pentagono.uniandes.edu.co/tutorial/Matlab/tutorial_matlab.pdf.

VIRGINIA, Johnson. LEAH, Rogers. "*Applying Soft Computing Methods to Improve the Computational Tractability of a Subsurface Simulation Optimization Problem"*. 2001.

WIEGAND, Klaus. *"Resorvoir Model Optimization Under Uncertainty"*. ExxonMobil URC. 2006.

YAN, Xia. REYNOLDS, Albert C. *"An Optimation Algorithm Based on Combining Finite-Differnce Approximations and Stochastic Gradients".* SPE 163613. 2013.

YANG, Won Y. CAO, Wenwu. CHUNG, Tae-Sang. Y MORRIS Jonh. *"Applied Numerical Methods Using MATLAB".* 2005.

YETEN, Burak., DURLOFSKY, Louis. AZIZ, Khalid. *Optimization of Nonconventional Well Type, Location and Trajectory.* SPE 7756. 2002

ZANDVLIET et al. SARMA Y CHEN. *Adjoint-Based Well-Placement Optimization under Production Constraints. SPE 105797. 2008.*

ZUBAREV, Denis Igorevich. *Pros and Cons of Applying Proxy-Models as a Substitute for Full Reservoir Simulations.* SPE 124815. 2009.

ANEXOS

Anexo A. Archivo .RWD.

```
FILES 'Project_Updated.irf'
output 'Project_Updated1.xls'
SPREADSHEET
**UBICACION
TABLE-FOR
 *COLUMN-FOR *PARAMETERS 'Cumulative Oil SC' *WELL 'U-41' *FILES 'Project_Updated.irf'
 *COLUMN-FOR *PARAMETERS 'Cumulative Gas SC' *WELL 'U-41' *FILES 'Project_Updated.irf'
 *COLUMN-FOR *PARAMETERS 'Cumulative Water SC' *WELL 'U-41' *FILES 'Project_Updated.irf'
TABLE-END
```

Anexo B. Código de Almacenamiento de las Variables.

```
clear all
close all
clc

load('Net To Gross/EXPORT-NTG/matlab.mat')

ntg = zeros(47, 32, 62);
for i=1:62
    eval(['ntg(:,:,i) = ',['NTG', num2str(i)],';']);
end
save ntg.mat ntg

clear all
close all
clc

load('POROSIDAD/Mat/matlab.mat')

porosity = zeros(47, 32, 62);
for i=1:62
    eval(['porosity(:,:,i) = ',['porosity', num2str(i)],';']);
end
save porosity.mat porosity

clear all
close all
clc

load('SATURACION/EXPORT/matlab.mat')

saturacion = zeros(47, 32, 62);
```

```
for i=1:62
    eval(['saturacion(:,:,i) = ',['SATURACION', num2str(i)],';']);
end
save saturacion.mat saturacion

clear all
close all
clc

load('Kefectivaaceite/EXPORT/matlab.mat')

kefectivanueva = zeros(47, 32, 62);
for i=1:62
    eval(['kefectivanueva(:,:,i)          =          ',['KEFECTIVAACEITE',
num2str(i)],';']);
end
save kefectivanueva.mat kefectivanueva
```

Anexo C. Códigos de Programación y Desarrollo de la Herramienta Wles 1.0.

HOJA 1

```
    existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before pozos_OpeningFcn gets called.  An
%     unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pozos_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help pozos
% Last Modified by GUIDE v2.5 28-Jun-2017 22:51:18
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pozos_OpeningFcn, ...
                   'gui_OutputFcn',  @pozos_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before pozos is made visible.
function pozos_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pozos (see VARARGIN)
% Choose default command line output for pozos
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
axes(handles.axes3)
handles.imagen=imread('ingpetroleos.jpg');
imagesc(handles.imagen)
axis off
% UIWAIT makes pozos wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = pozos_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pozos1;
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pozos2;



% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pozos3;
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pozos4;
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pozos5;
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pozos1;
```

HOJA 2.

```
function varargout = pozos1(varargin)
% POZOS1 MATLAB code for pozos1.fig
%      POZOS1, by itself, creates a new POZOS1 or raises the existing
%      singleton*.
%      H = POZOS1 returns the handle to a new POZOS1 or the handle to
%      the existing singleton*.
%      POZOS1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in POZOS1.M with the given input arguments.
%
%      POZOS1('Property','Value',...) creates a new POZOS1 or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
```

```
%      applied to the GUI before pozos1_OpeningFcn gets called.  An
%     unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pozos1_OpeningFcn via varargin.
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLE
% Edit the above text to modify the response to help pozos1
% Last Modified by GUIDE v2.5 29-Jun-2017 13:16:0
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pozos1_OpeningFcn, ...
                   'gui_OutputFcn',  @pozos1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before pozos1 is made visible.
function pozos1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pozos1 (see VARARGIN)
% Choose default command line output for pozos1
handles.output = hObject;
% Update handles structure
```

```
guidata(hObject, handles);
axes(handles.axes4)
handles.imagen=imread('ingpetroleos.jpg');
imagesc(handles.imagen)
axis off
% UIWAIT makes pozos1 wait for user response (see UIRESUME)
% uiwait(handles.figure1)
% --- Outputs from this function are returned to the command line.
function varargout = pozos1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
filename = uigetfile('*.mat');
command = sprintf('load(''%s'')', filename);
evalin('base',command);
A=importdata('porosity.mat');
set(handles.uitable1,'data',A);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%imagenes cmg
fileFolder                                                                    =
fullfile('C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\POROSIDAD
IMAGEN\CMG')
dirOutput = dir(fullfile(fileFolder))
fileNames = {dirOutput.name}
numFrames = numel(fileNames)
name=fileNames{3};
```

```
content                                                                =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\POROSIDADIMAGEN\CM
G';
fullname = [content,'\',name];
I =imread(fullname);
% % Preallocate the array
sequence = zeros([size(I) (numFrames-2)],class(I));
% sequence(:,:,1) = I;
% % Create image sequence array
for p = 3:numFrames
name=fileNames{p};
content                                                                =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\POROSIDADIMAGEN\CM
G';
fullname = [content,'\',name];

sequence(:,:,:,p-2) = imread(fullname);
end
implay(sequence)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%imagenes matlab
%imagenes cmg
fileFolder                                                             =
fullfile('C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\POROSIDAD
IMAGEN\MATLAB')
dirOutput = dir(fullfile(fileFolder))

fileNames = {dirOutput.name}
numFrames = numel(fileNames)
name=fileNames{3};
content                                                                =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\POROSIDADIMAGEN\MA
TLAB';
fullname = [content,'\',name];
I =imread(fullname);
```

```
% % Preallocate the array
sequence = zeros([size(I) (numFrames-2)],class(I));
% sequence(:,:,1) = I;
% % Create image sequence array
for p = 3:numFrames
name=fileNames{p};
content                                                                =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\POROSIDADIMAGEN\MA
TLAB';
fullname = [content,'\',name];
sequence(:,:,:,p-2) = imread(fullname);
end
implay(sequence)
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
filename = uigetfile('*.mat');
command = sprintf('load(''%s'')', filename);
evalin('base',command);
B=importdata('saturacion.mat');
set(handles.uitable2,'data',B);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%imagenes cmg
fileFolder                                                             =
fullfile('C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\saturacio
nIMAGEN\CMG')
dirOutput = dir(fullfile(fileFolder))
fileNames = {dirOutput.name}
numFrames = numel(fileNames)
name=fileNames{3};
content                                                                =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\saturacionIMAGEN\C
MG';
```

```
fullname = [content,'\',name];

I =imread(fullname);
% % Preallocate the array
sequence = zeros([size(I) (numFrames-2)],class(I));
% sequence(:,:,1) = I;
% % Create image sequence array
for p = 3:numFrames
name=fileNames{p};
content                                                                =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\saturacionIMAGEN\C
MG';
fullname = [content,'\',name];

sequence(:,:,:,p-2) = imread(fullname);
end
implay(sequence)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%imagenes MATLAB
fileFolder                                                             =
fullfile('C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\saturacio
nIMAGEN\MATLAB')
dirOutput = dir(fullfile(fileFolder)
fileNames = {dirOutput.name}
numFrames = numel(fileNames)
name=fileNames{3};
content                                                                =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\saturacionIMAGEN\M
ATLAB';
fullname = [content,'\',name];
I =imread(fullname);
% % Preallocate the array
sequence = zeros([size(I) (numFrames-2)],class(I));
% sequence(:,:,1) = I;
% % Create image sequence array
```

```
for p = 3:numFrames
name=fileNames{p};
content                                                                    =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\saturacionIMAGEN\M
ATLAB';
fullname = [content,'\',name];
sequence(:,:,:,p-2) = imread(fullname);
end
implay(sequence)
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
filename = uigetfile('*.mat');
command = sprintf('load(''%s'')', filename);
evalin('base',command);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%imagenes cmg
fileFolder                                                                 =
fullfile('C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\ntgIMAGEN
\CMG')
dirOutput = dir(fullfile(fileFolder))
fileNames = {dirOutput.name}
numFrames = numel(fileNames)
name=fileNames{3};
content                                                                    =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\ntgIMAGEN\CMG';
fullname = [content,'\',name];


I =imread(fullname);
% % Preallocate the array
sequence = zeros([size(I) (numFrames-2)],class(I));
% sequence(:,:,1) = I;
% % Create image sequence array
```

```
for p = 3:numFrames
name=fileNames{p};
content                                                                    =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\ntgIMAGEN\CMG';
fullname = [content,'\',name];
sequence(:,:,:,p-2) = imread(fullname);
end
implay(sequence)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%imagenes MATLAB
fileFolder                                                                 =
fullfile('C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\ntgIMAGEN
\MATLAB')
dirOutput = dir(fullfile(fileFolder))
fileNames = {dirOutput.name}
numFrames = numel(fileNames)
name=fileNames{3};
content                                                                    =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\ntgIMAGEN\MATLAB';
fullname = [content,'\',name];
I =imread(fullname);
% % Preallocate the array
sequence = zeros([size(I) (numFrames-2)],class(I));
% sequence(:,:,1) = I;
% % Create image sequence array
for p = 3:numFrames
name=fileNames{p};
content                                                                    =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\ntgIMAGEN\MATLAB';
fullname = [content,'\',name];
sequence(:,:,:,p-2) = imread(fullname);
end
implay(sequence)
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
filename = uigetfile('*.mat');
command = sprintf('load(''%s'')', filename);
evalin('base',command);
fileFolder                                                              =
fullfile('C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\Kefectiva
nuevaIMAGEN')
dirOutput = dir(fullfile(fileFolder))
fileNames = {dirOutput.name}
numFrames = numel(fileNames)
name=fileNames{3};
content                                                                 =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\KefectivanuevaIMAG
EN';
fullname = [content,'\',name];
I =imread(fullname);
% % Preallocate the array
sequence = zeros([size(I) (numFrames-2)],class(I));
% sequence(:,:,1) = I;
% % Create image sequence array
for p = 3:numFrames
name=fileNames{p};
content                                                                 =
'C:\Users\USUARIO\Desktop\DATOS\guide\IMAGENESVARIBLES\KefectivanuevaIMAG
EN';
fullname = [content,'\',name];
sequence(:,:,:,p-2) = imread(fullname);
end
implay(sequence)
% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
filename = uigetfile('*.mat');
command = sprintf('load(''%s'')', filename);
evalin('base',command);
% --- Executes when entered data in editable cell(s) in uitable1.
function uitable1_CellEditCallback(hObject, eventdata, handles)
% hObject    handle to uitable1 (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) edited
%   PreviousData: previous data for the cell(s) edited
%   EditData: string(s) entered by the user
%   NewData: EditData or its converted form set on the Data property. Empty
if Data was not changed
%    Error: error string when failed to convert EditData to appropriate
value for Data
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pozos2;
```

HOJA 3.

```
function varargout = pozos2(varargin)
% POZOS2 MATLAB code for pozos2.fig
%      POZOS2, by itself, creates a new POZOS2 or raises the existing
%      singleton*.
%      H = POZOS2 returns the handle to a new POZOS2 or the handle to
%      the existing singleton*.
%
%      POZOS2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in POZOS2.M with the given input arguments.
%
%      POZOS2('Property','Value',...) creates a new POZOS2 or raises the
```

```
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before pozos2_OpeningFcn gets called.  An
%     unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pozos2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help pozos2
% Last Modified by GUIDE v2.5 28-Jun-2017 23:11:46
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pozos2_OpeningFcn, ...
                   'gui_OutputFcn',  @pozos2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end


if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before pozos2 is made visible.
function pozos2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pozos2 (see VARARGIN)
```

```
% Choose default command line output for pozos2
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
axes(handles.axes3)
handles.imagen=imread('ingpetroleos.jpg');
imagesc(handles.imagen)
axis off
% UIWAIT makes pozos2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = pozos2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double
kefectivanueva=importdata('kefectivanueva.mat');
porosity=importdata('porosity.mat');
saturacion=importdata('saturacion.mat');
espesor=importdata('ESPESOR.mat');
ntg=importdata('ntg.mat');

val=get(hObject,'String');
newval=str2double(val);
handles.edit1=newval;
guidata(hObject,handles);
```

```
[M, N, L] = size(ntg);
int_rng   = 18;
%cargar un if
%Z = 10 Intervalo ; for = 0:52 ; kefectivanueva ((1:10)+i)
%Z = 15 Intervalo ; for = 0:47  ; kefectivanueva ((1:15)+i)
%Z = 20 Intervalo ; for = 0:42 ; kefectivanueva ((1:20)+i)
h = espesor(:);

kefectivanueva = reshape(kefectivanueva, M*N, L);
for i = 0:62-handles.edit1
     tmp  = kefectivanueva(:,(1:handles.edit1)+i);
     tmp2 = h((1:handles.edit1)+i);
     for j = 1:M*N
        [x,y,v] = find(tmp(j,:));
        if isempty(v)
           K_mean(j,i+1) = 0;
        else
           K_mean(j,i+1) = sum(v(:).*tmp2(y))/sum(tmp2(y));
        end
     end

end

ntg = reshape(ntg, M*N, L);
for i = 0:62-handles.edit1
     tmp = ntg(:,(1:handles.edit1)+i);
     tmp2 = h((1:handles.edit1)+i);
     for j = 1:M*N
        [x,y,v] = find(tmp(j,:));
        if isempty(v)
           NTG_mean(j,i+1) = 0;
        else
           NTG_mean(j,i+1) = sum(v(:).*tmp2(y))/sum(tmp2(y));

        end
     end
```

```
end


porosity = reshape(porosity, M*N, L);
for i = 0:62-handles.edit1
     tmp = porosity(:,(1:handles.edit1)+i);
     tmp2 = h((1:handles.edit1)+i);
     for j = 1:M*N
        [x,y,v] = find(tmp(j,:));
        if isempty(v)
           PI_mean(j,i+1) = 0;
        else
           PI_mean(j,i+1) = sum(v(:).*tmp2(y))/sum(tmp2(y));
        end
     end

end

saturacion = reshape(saturacion, M*N, L);
for i = 0:62-handles.edit1
     tmp = saturacion(:,(1:handles.edit1)+i);
     tmp2 = h((1:handles.edit1)+i);
     for j = 1:M*N
        [x,y,v] = find(tmp(j,:));
        if isempty(v)
           S_mean(j,i+1) = 0;
        else
           S_mean(j,i+1) = sum(v(:).*tmp2(y))/sum(tmp2(y));

        end
     end

end
```

```
save hoja2.mat,(K_mean(j,i+1))
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
filename = uigetfile('*.mat');
command = sprintf('load(''%s'')', filename);
evalin('base',command);
A=importdata('S_mean.mat');
set(handles.uitable1,'data',A);
B=importdata('PI_mean.mat');
set(handles.uitable4,'data',B);
D=importdata('K_mean.mat');
set(handles.uitable3,'data',D);
E=importdata('NTG_mean.mat');
set(handles.uitable5,'data',E);
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes when entered data in editable cell(s) in uitable1.
function uitable1_CellEditCallback(hObject, eventdata, handles)
% hObject    handle to uitable1 (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) edited
%   PreviousData: previous data for the cell(s) edited
%   EditData: string(s) entered by the user
```

```
%   NewData: EditData or its converted form set on the Data property. Empty
if Data was not changed
%    Error: error string when failed to convert EditData to appropriate
value for Data
% handles    structure with handles and user data (see GUIDATA)
% --- Executes during object creation, after setting all properties.
function uitable1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uitable1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pozos3;
```

HOJA 4

```
function varargout = pozos3(varargin)
% POZOS3 MATLAB code for pozos3.fig
%      POZOS3, by itself, creates a new POZOS3 or raises the existing
%      singleton*.
%      H = POZOS3 returns the handle to a new POZOS3 or the handle to
%      the existing singleton*.
%      POZOS3('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in POZOS3.M with the given input arguments.
%      POZOS3('Property','Value',...) creates a new POZOS3 or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before pozos3_OpeningFcn gets called.  An
%    unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pozos3_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```
% Edit the above text to modify the response to help pozos3
% Last Modified by GUIDE v2.5 29-Jun-2017 00:14:41
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pozos3_OpeningFcn, ...
                   'gui_OutputFcn',  @pozos3_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end


if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before pozos3 is made visible.
function pozos3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pozos3 (see VARARGIN)
% Choose default command line output for pozos3
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
axes(handles.axes5)
handles.imagen=imread('ingpetroleos.jpg');
imagesc(handles.imagen)
axis off
% UIWAIT makes pozos3 wait for user response (see UIRESUME)
```

```
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = pozos3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double
S_mean=importdata('S_mean.mat');
PI_mean=importdata('PI_mean.mat');
K_mean=importdata('K_mean.mat');
NTG_mean=importdata('NTG_mean.mat');
ntg=importdata('ntg.mat');
val=get(hObject,'String');
newval=str2double(val);
handles.edit1=newval;
guidata(hObject,handles);
[M, N, L] = size(ntg);
ntg = reshape(ntg, M*N, L);
Potential = ((K_mean.*PI_mean).*S_mean).*NTG_mean;
set(handles.uitable1,'data',Potential)

axes(handles.axes3)
```

```
mesh(reshape(Potential(:,30), M, N));
% mesh(reshape(Potential(:,12), M, N));
% axes(handles.axes1);
P  = handles.edit1;
max_pot = zeros(P, 1);
[temp, idx] = sort(abs(Potential(:)), 'descend');
[cor, int] = ind2sub([M*N L], idx);
[x, y]     = ind2sub([M N], cor);
Coordenadas=[y(1:P) x(1:P) int(1:P)];
set(handles.uitable2,'data',Coordenadas)
A = Potential;
B = max(max(A));
C = A/B
axes(handles.axes1)
handles.imagen=surf(reshape(C(:,15), M, N));
imagesc(handles.imagen)

surf(reshape(C(:,25), M, N))
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pozos4;
```

HOJA 5

```
function varargout = pozos4(varargin)
% POZOS4 MATLAB code for pozos4.fig
%      POZOS4, by itself, creates a new POZOS4 or raises the existing
%      singleton*.
%
%      H = POZOS4 returns the handle to a new POZOS4 or the handle to
%      the existing singleton*.
%
%      POZOS4('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in POZOS4.M with the given input arguments.
%
%      POZOS4('Property','Value',...) creates a new POZOS4 or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before pozos4_OpeningFcn gets called.  An
```

```
%     unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pozos4_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help pozos4

% Last Modified by GUIDE v2.5 22-Jul-2017 17:09:34

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pozos4_OpeningFcn, ...
                   'gui_OutputFcn',  @pozos4_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before pozos4 is made visible.
function pozos4_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pozos4 (see VARARGIN)
% Choose default command line output for pozos4
```

```
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
axes(handles.axes2)
handles.imagen=imread('ingpetroleos.jpg');
imagesc(handles.imagen)
axis off
% UIWAIT makes pozos4 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = pozos4_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen   ('C:\Users\USUARIO\Desktop\DATOS\guide\DATOS   DE   PRODUCCION\10
LAYER')
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('C:\Users\USUARIO\Desktop\DATOS\guide\DATOS DE VPN\10 LAYER')
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
winopen   ('C:\Users\USUARIO\Desktop\DATOS\guide\DATOS   DE   PRODUCCION\15
LAYER')
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('C:\Users\USUARIO\Desktop\DATOS\guide\DATOS DE VPN\15 LAYER')
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen   ('C:\Users\USUARIO\Desktop\DATOS\guide\DATOS   DE   PRODUCCION\10
LAYER')
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('C:\Users\USUARIO\Desktop\DATOS\guide\DATOS DE VPN\20 LAYER')

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double
% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a
double
% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a
double
% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a
double
% --- Executes during object creation, after setting all properties.
```

```
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a
double
% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a
double
% --- Executes during object creation, after setting all properties.
```

```
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pozos5;
function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a
double
% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

HOJA 6

```
function varargout = pozos5(varargin)
% POZOS5 MATLAB code for pozos5.fig
%      POZOS5, by itself, creates a new POZOS5 or raises the existing
%      singleton*.

%      H = POZOS5 returns the handle to a new POZOS5 or the handle to
%      the existing singleton*.
%      POZOS5('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in POZOS5.M with the given input arguments.
%      POZOS5('Property','Value',...) creates a new POZOS5 or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before pozos5_OpeningFcn gets called.  An
%     unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pozos5_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help pozos5
% Last Modified by GUIDE v2.5 26-Jun-2017 16:10:03
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pozos5_OpeningFcn, ...
                   'gui_OutputFcn',  @pozos5_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before pozos5 is made visible.
function pozos5_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pozos5 (see VARARGIN)
% Choose default command line output for pozos5
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
axes(handles.axes3)
handles.imagen=imread('ingpetroleos.jpg');
imagesc(handles.imagen)
axis off
axes(handles.axes1)
handles.imagen=imread('graficahoja5.png');
imagesc(handles.imagen)
axis off
% UIWAIT makes pozos5 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = pozos5_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('C:\Users\USUARIO\Desktop\DATOS\guide\DEFINICIONESTRATEGICA')
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('C:\Users\USUARIO\Desktop\DATOS\guide\DEFINICIONESTRATEGICA\PRODU
CCION')
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('C:\Users\USUARIO\Desktop\DATOS\guide\DEFINICIONESTRATEGICA\VALOR
PRESENTE NETO')
% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('C:\Users\USUARIO\Desktop\DATOS\guide\DEFINICIONESTRATEGICA')
% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: place code in OpeningFcn to populate axes1
```

Anexo D. Instructivo de Usuario

REQUISITOS DE EJECUCION DE LA HERRAMIENTA WLES 1.0

Se requiere que el equipo en el cual se quiera ejecutar la interfaz gráfica tenga instalado previamente el programa Matlab, el cual ejecutara la interfaz. Igualmente se debe almacenar todos los archivos, imágenes y documentos que están inmersos en la ejecución y lectura por parte de la interfaz.

DESCRIPCIÓN DE LA HERRAMIENTA.

La herramienta Well Location Evaluation software (WLES) es compuesta principalmente por seis (6) hojas, donde cada una de ellas ejecuta cálculos contiguos para un desarrollo favorable de la estrategia y de la interfaz. Cada una de las hojas es creada a partir de dos archivos: .Mat y un .Fig estos dos extensiones son ejecutables por Matlab. El archivo .Mat contiene el scrip del código de programación y el .Fig crea la ventana grafica de la herramienta.

Previamente a la ejecución de cada una de las hojas el usuario debe abrir la carpeta que contiene los archivos, gráficas y documentos en el directorio de Matlab (Current Folder) de la siguiente manera.

1. Abrir Matlab (Haciendo doble click el icono de Matlab)

2. Hacer click en el botón  de la barra de herramientas de Matlab.

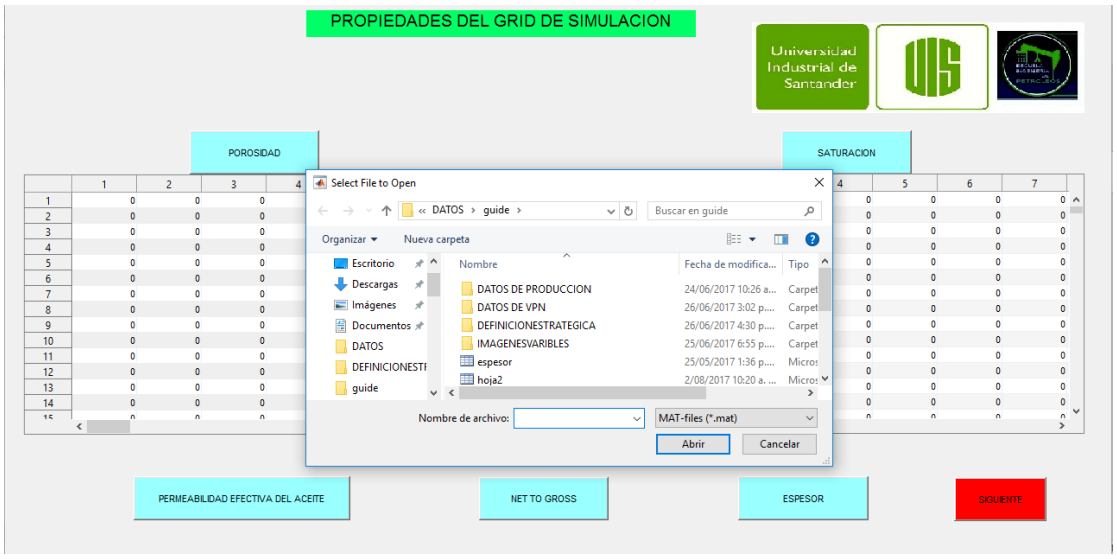
3. Seleccionar la carpeta donde se almacenan los archivos que se quieren ejecutar en la interfaz.

A continuación se presentan las configuraciones de cada una de las hojas de la interfaz y la acción de cada uno de los botones que contiene cada una de ellas.

Hoja 1.

En la hoja 1, es una hoja de presentación. Donde se presenta el nombre del proyecto desarrollado, los autores del mismo y cinco (5) botones los cuales indican la secuencia de los cálculos realizados para el desarrollo de la estrategia de ubicación de los pozos.

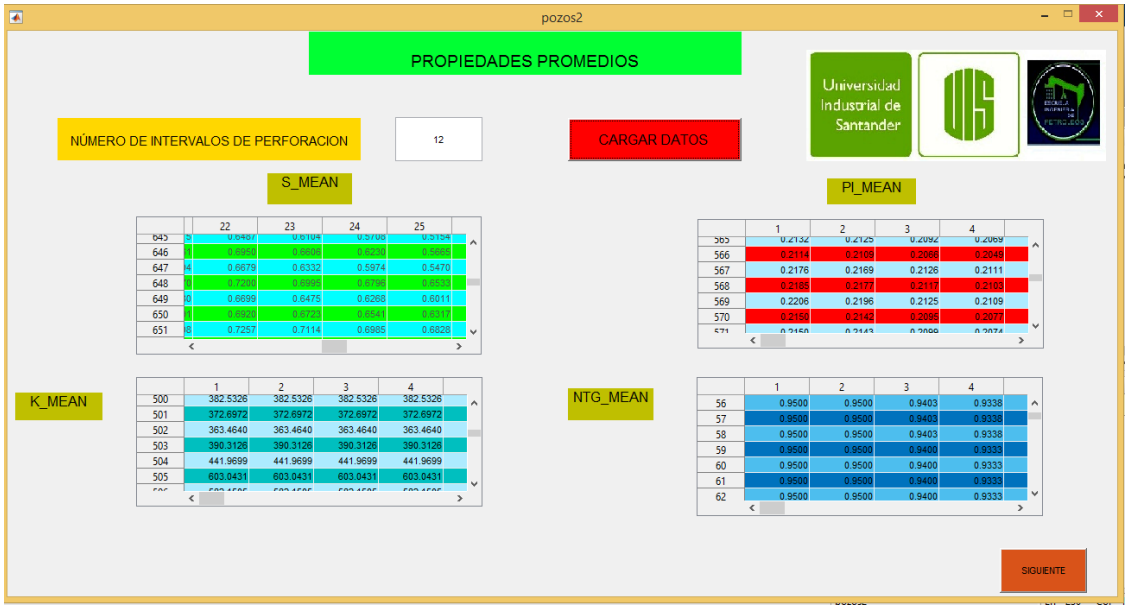
Hoja 2



La hoja 2 cuenta con cinco botones principales; POROSIDAD, SATURACIÓN, PERMEABILIDAD EFCETIVA AL ECEITE, NET TO GROSS y ESPESOR.

Estos botones le permiten al usuario cargar los archivos donde están almacenadas cada una de la variables que fueron exportadas del modelo de simulación a la interfaz, para posteriormente ser utilizada en los calculo a realizar.

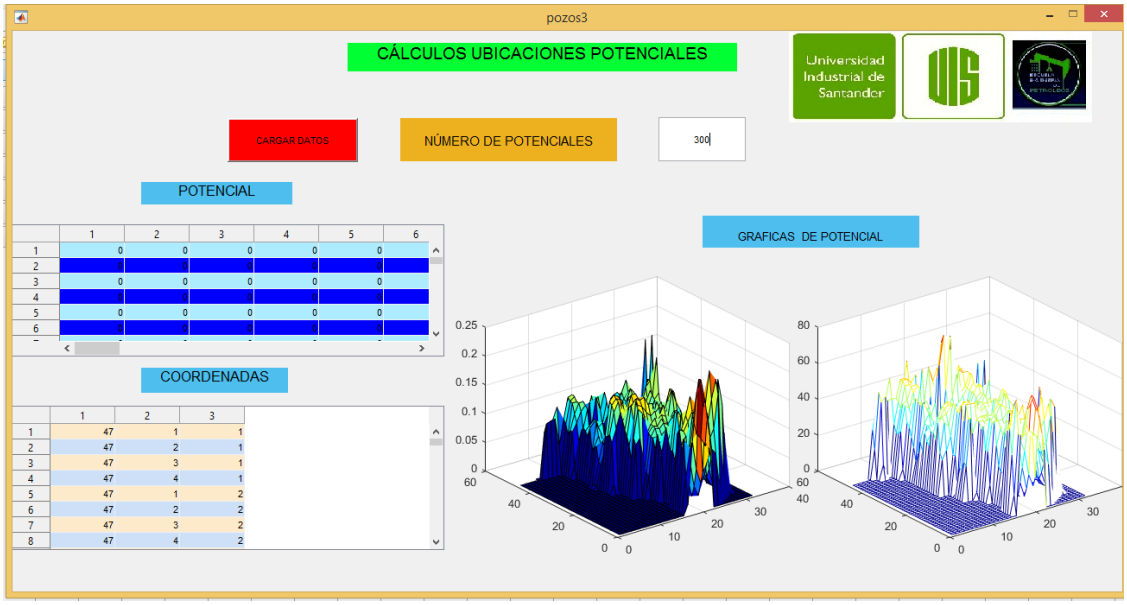
Hoja 3.



La hoja 3 contiene cuatro botones de tipo *Static Text* los cuales son: S_MEAN, PI_MEAN, K_MEAN, NTG_MEAN. Estos botones se refieren a las matrices promedios de cada variable dependiendo del intervalo de perforación que el usuario establezca.

Para establecer el intervalo de perforación el usuario debe escribir el intervalo que él requiera, lo debe hacer el botón de *Edit Text*. La ventana presenta el ejemplo donde se estableció intervalo de 10 capas. Posteriormente el usuario debe hacer click en el botón *cargar datos* donde se desplegará una ventana que permite al usuario seleccionar el documento que almacena las matrices promedios, en este caso llamado Hoja 2 de extensión .Dat. Finalmente en las tablas de las ventanas se observan las matrices calculadas.

Hoja 4

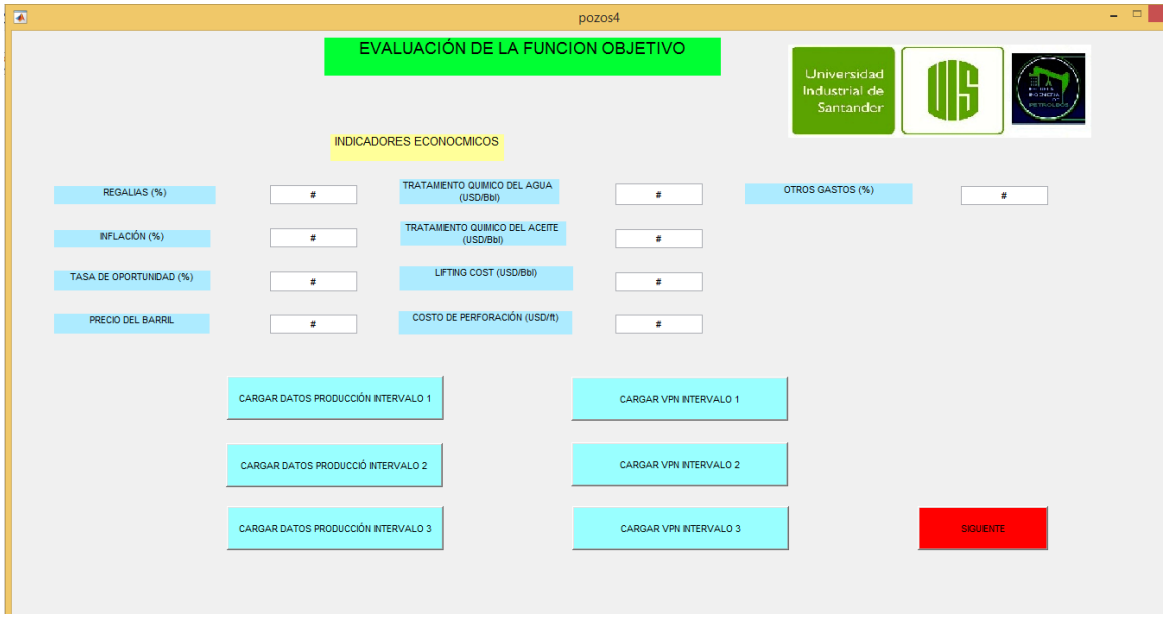


El principal objetivo de esta hoja es establecer el número de potenciales que el usuario quiere conocer, le permite al usuario imprimir y conocer diferentes cantidades de potenciales y de esta manera conocer las diferentes coordenadas de mayor potencial para las ubicaciones de los pozos productores, como lo propone la estrategia.

El botón *cargar datos* le permite al usuario cargar los datos almacenados en cada una de las matrices promedios, consecutivamente el usuario de escribir el número de potenciales que quiere conocer y luego accionar el botón de Enter de su equipo. Finalmente, el usuario podrá observar los valores numéricos de los potenciales y las respectivas coordenadas en las cuales se encuentran esto potenciales; igualmente el usuario puede observar gráficamente la distribución de los potenciales, para esto el usuario debe hacer uso de la línea 113 y escribir el intervalo que quiere observar.

Por ejemplo: `mesh(reshape(Potential(:,12), M, N));` en esta línea el usuario observa la distribución de potenciales en el intervalo 12, si requiere observar un intervalo diferente debe escribir el número de intervalo que requiere observar.

Hoja 5.



La hoja 4 es una hoja de lectura de archivos, es decir, en la ejecución de esta hoja el usuario podrá observar la evaluación económica realizada en el desarrollo del proyecto. La hoja muestra los factores económicos y operativos establecidos por los autores para el cálculo del Valor Presente Neto de cada uno de los pozos perforados y para el campo, en cada uno de los intervalos de perforación previamente seleccionados.

La acción de los botones *Cargas producción intervalo 1, Cargas producción intervalo 2,* Cargas *producción intervalo 3,* le permiten al usuario inicialmente cargar los datos de producción de cada pozo o de todo el campo como el usuario requiera, estos datos se encuentran en un archivo .exe previamente almacenados. Posteriormente el usuario debe cargar los archivos de Valor Presente Neto haciendo

uso de los botones *Cargas VPN intervalo 1, Cargas VPN intervalo 2, Cargas VPN intervalo 3,* de esta manera se puede observar el comportamiento económico mensual del proyecto durante todo el tiempo de evaluación económica del mismo; en estos archivos es posible observar el Flujo de Caja y el Valor presente neto de cada pozo y de todo el campo principalmente. Son archivos .exe previamente calculado y almacenados.

Hoja 6.

La hoja 4 es una hoja de lectura de archivos, es decir, esta hoja le permite al usuario observar los pozos seleccionados para la definición de la estrategia, esto lo hace a través de la lectura de un archivo .Dat donde se encuentran escritos los pozos seleccionados. El botón Datos de Producción le permite al usuario cargar la producción acumulada mensual del campo luego de la aplicación de la estrategia; el botón Ubicación de Pozos le permite al usuario leer el archivo .Dat generado por la simulación, este archivo debe ser copiado a la carpeta donde se encuentran los documentos de la interfaz. Por último el botón Cargar VPN que le permite al usuario seleccionar el documento (.Exe) donde se realizó la evaluación económica de la estrategia.

LIMITACIONES DE LA HERRAMIENTA WLES 1.0

La herramienta WLES 1.0 presenta limitaciones tales como:

- Es posible utilizar y ejecutar la herramienta cuando se tiene un modelo de simulación tipo Corner Point, el cual fue el tipo de malla utilizada para el desarrollo del proyecto.
- La evaluación de las ubicaciones potenciales es realizada solo a un tiempo determinado, es decir, la evaluación se realiza para determinada fecha en la cual fueron exportadas las variables del modelo base de simulación numérica.
- La herramienta no tiene la capacidad de reproducir condiciones estructurales establecidas en el modelo de simulación. Condiciones como, por ejemplo: Fallas Geológicas.