UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de un Software para la Generación de Mallas Estructuradas no Ortogonales con Aplicación a la Simulación Matemática de Yacimientos

MATERIAL DIDÁCTICO

Que para obtener el título de

Ingeniero Petrolero

P R E S E N T A

Iván Eduardo Beltrán González

ASESOR DE MATERIAL DIDÁCTICO

Dr. Víctor Leonardo Teja Juárez

Ciudad Universitaria, Cd. Mx., 2022

Agradecimientos

A mis padres, Paula González y Anatolio Beltrán, por su apoyo incondicional, amor y valores. Son mi mayor orgullo e inspiración, siempre les estaré agradecidos por darme la vida y diversos aprendizajes.

A mi asesor de material didáctico, Dr. Víctor Leonardo Teja Juárez, cuyo apoyo, paciencia y sabiduría me permitió aprender sobre temas complejos, disfrutar de la programación y mejorar como persona e ingeniero.

A mi hermana, Brenda Beltrán, por ser una persona ejemplar y demostrar que con esfuerzo y dedicación se puede ir más allá de los límites.

A mis amigos y compañeros de clase, Héctor, Carlos, Javier, Ubiliado y David, gracias por haberlos conocido, reído con ustedes y sufrido en las materias difíciles. La universidad se tornó más amena cuando los conocí.

A mis profesores, por trasmitir su conocimiento y experiencia en cada clase. Sin duda, todas las pruebas que aplican semestre a semestre permiten a los alumnos a mejorar, aprender y generar trabajos de calidad.

A la Facultad de Ingeniería y la Universidad Nacional Autónoma de México por permitirme estudiar en sus aulas y vivir increíbles experiencias.

Este trabajo se desarrolló como parte del proyecto PAPIIT IA106820, participando en el desarrollo del modelo numérico y facilitando los recursos computacionales para llevarlo a cabo.

Índice

Lista de figuras

Lista de tablas

Nomenclatura

Símbolo	Representa
α, β, γ	Coeficientes para expresar el sistema curvilíneo
x, y	Coordenadas dentro del dominio físico
ξ, η	Coordenadas dentro del dominio computacional
f	Flujo fraccional
ϕ	Porosidad
k	Permeabilidad
p_c	Presión capilar
ρ	Densidad
s	Saturación
q	Gasto
μ	Viscosidad

Resumen

El presente material didáctico brinda las herramientas teóricas y prácticas para desarrollar aplicaciones de escritorio con el lenguaje de programación Python. Estos programas de cómputo están enfocados en la simulación matemática de yacimientos, donde la principal característica es la incorporación de mallas estructuradas no ortogonales en dos dimensiones a un problema de ingeniería de yacimientos como la inyección de agua.

Los primeros dos capítulos son la base teórica necesaria para elaborar un algoritmo que genera las mallas estructuradas no ortogonales en 2D. Posteriormente, se hace una breve introducción al lenguaje Python y se dan ejemplos de cómo crear interfaces gráficas, las cuales son la base para la elaboración de MallaPy_2D.

Finalmente, se realiza un acople entre la aplicación MallaPy_2D y un código de simulación numérica de yacimientos, el cual fue migrado del leguaje C++ a Python, como resultado da otra aplicación de escritorio llamada SMYPy_UNAM, un software con la capacidad de realizar simulaciones de yacimientos sobre mallas estructuradas no ortogonales. Para comprobar el funcionamiento del software, se realizan corridas de simulación donde se varían los parámetros del yacimiento y se analizan los gráficos resultantes.

Introducción

Una de las formas más precisas de analizar y comprender un problema de ingeniería es mediante la simulación matemática, la cual engloba los modelos: físico, matemático, numérico y computacional. Esta disciplina permite aplicar los fundamentos de la matemática computacional, donde se parte de un modelo físico en el cual se describen las consideraciones necesarias para resolver un problema de ingeniería, posteriormente, se estructuran las ecuaciones necesarias en el modelo matemático y finalmente se implementan algoritmos numéricos que se pueden resolver mediante un lenguaje de programación.

Una de las partes fundamentales de la simulación matemática es la malla, dependiendo de la distribución de los nodos y elementos, ésta puede ser estructurada o no estructurada. La primera categoría implica una mejor manipulación de los nodos y elementos, además de una fácil adaptación con los algoritmos de simulación, debido a la distribución reticular, por otro lado, la segunda categoría requiere mayor uso de memoria y complejidad en el código.

Con la llegada de lenguajes de programación de alto nivel como Python, hoy en día se pueden aplicar técnicas de generación de mallas de forma práctica. De igual manera, es posible generar interfaces de usuario que pueden automatizar los procesos de la generación de mallas de una forma sencilla e interactiva.

Objetivo

Desarrollar una interfaz gráfica portable y sencilla que permita aplicar al alumno los conceptos teóricos y prácticos de la generación de mallas estructuradas no ortogonales en la simulación matemática de yacimientos. Esto con la finalidad de fomentar la creación de herramientas digitales que funjan como complemento a los análisis de simulación matemática de yacimientos.

Objetivos particulares

1. Investigar y analizar la técnica de generación de mallas por coordenadas de cuerpo ajustado.
2. Migrar un código de mallado existente del lenguaje de programación Fortran al lenguaje Python.
3. Desarrollar una interfaz gráfica para uso de la técnica de mallado por coordenadas de cuerpo ajustado.
4. Integrar la interfaz gráfica desarrollada a un simulador bifásico.

Alcance

El alumno podrá generar una interfaz gráfica de usuario con el lenguaje Python, capaz de generar mallas estructuradas no ortogonales y adaptarlas a un problema de simulación de yacimientos. Este software dispondrá de elementos interactivos para dibujar, modificar y visualizar mallas. De igual manera, también se contemplan componentes para variar los parámetros del problema de simulación de una manera intuitiva y sencilla.

1. Conceptos fundamentales

En este capítulo se explican los conceptos e información necesaria para comprender el contexto de la simulación matemática de yacimientos (SMY).

1.1 Simulación Matemática de Yacimientos (SMY)

La simulación matemática de yacimientos se refiere a la construcción y operación de un modelo computacional que emule a un yacimiento real (PetroWiki, 2018). Esto con el objetivo de predecir el comportamiento futuro del yacimiento y encontrar los medios para aumentar la recuperación de hidrocarburos. Para desarrollar un software de SMY es necesario partir de la siguiente metodología:

Figura 1. Metodología para desarrollar un simulador de yacimientos. Modificado de (Hernández & Dominguez, 1984).

En el presente material didáctico se hace énfasis en mejorar el modelo computacional. En el cual se propondrá una interfaz gráfica de usuario con la capacidad de modificar los parámetros de simulación y en particular, la malla del yacimiento.

1.1.1 Modelo físico

Se definen las consideraciones necesarias para poder modelar tanto matemática como numéricamente el problema de ingeniería, mismo que dependerá del método de recuperación secundaria o mejorada que se desea implementar.

1.1.2 Modelo matemático

Se parte de las consideraciones asumidas en el modelo físico. De igual manera, se definen las ecuaciones diferenciales con sus respectivas condiciones iniciales y de frontera.

1.1.3 Modelo numérico

Debido a la complejidad de las ecuaciones que constituyen el modelo matemático, el modelo numérico plantea una solución más amigable, de tal manera que estas ecuaciones se puedan resolver con ayuda de un ordenador.

Uno de los mayores retos es convertir el set de ecuaciones diferenciales continuas a una forma más simple y discreta que pueda ser programada en una computadora. Es ahí, donde técnicas de discretización como el de diferencias o volumen finitos toman relevancia. La discretización se lleva a cabo aplicando formulaciones matemáticas las cuales dependen del método que se elija, por ejemplo, en el método de diferencias finitas las fórmulas de discretización de las derivadas de primer y segundo orden se obtienen mediante la aplicación de expansión en series de Taylor alrededor de un punto seleccionado.

1.1.4 Modelo computacional

Consiste en la elaboración de un programa de cómputo o una serie de programas que resuelvan las ecuaciones del modelo numérico mediante un algoritmo previamente seleccionado. La calidad de los datos que se proveen a la entrada del programa, las correlaciones, el tipo de mallado, etc., serán determinantes para realizar un buen análisis de simulación.

1.1.5 Clasificación de los simuladores de yacimientos

En la figura 2 se ilustra una clasificación general de los simuladores. No obstante, factores como el número de fases, dimensiones del modelo de simulación o coordenadas pueden llegar a determinar nuevas categorías.

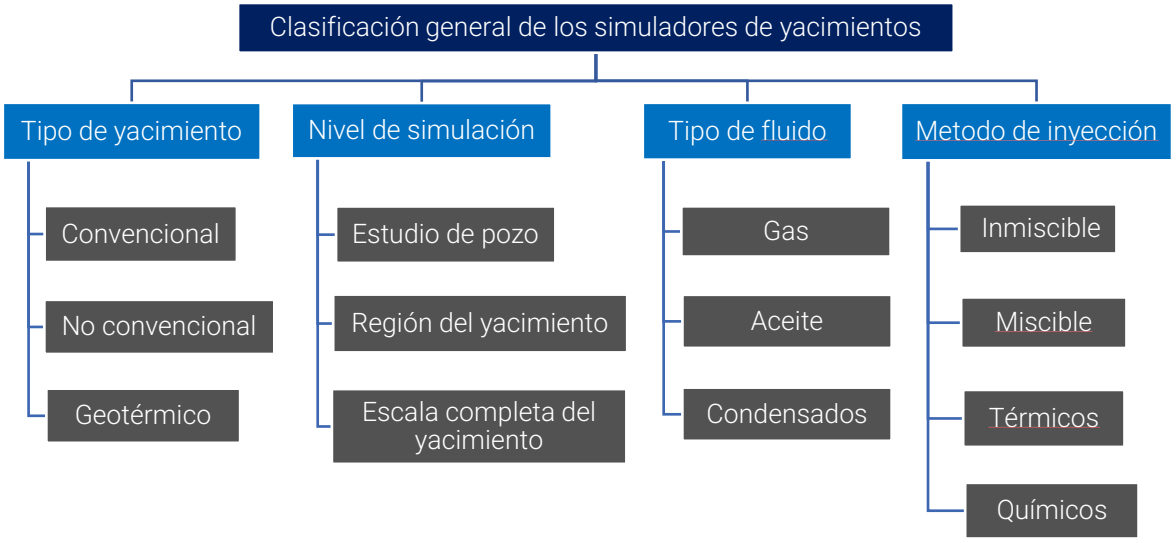


Figura 2. Clasificación de los simuladores de yacimientos. Modificado de (Hernández & Dominguez, 1984).

1.1.6 Software relacionado a SMY

El software existente para realizar un análisis de SMY puede ser privado, comercial o de licencia libre. Este último desarrollado sin fines de lucro y apoyado por la comunidad de desarrolladores independientes o instituciones educativas. En la tabla 1 se ilustran algunos de los simuladores más utilizados en la industria petrolera.

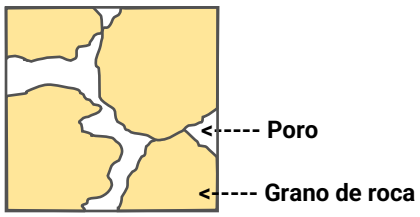
Tabla 1. Descripción de los softwares de simulación más usados

Software	Licencia	Características
CMG COMPUTER MODELLING GROUP LTD. **CMG Suite**	Comercial	La suite de CMG incluye simuladores como IMEX, GEM, STARS, que permiten realizar análisis de simulación para aceite negro, métodos térmicos, composicionales, inyección de químicos, etc.
tNavigator® **tNavigator**	Comercial	Desarrollado por Rock Flow Dynamics y escrito en C++, tNavigator incluye un entorno capaz de realizar análisis estáticos y dinámicos del yacimiento.
ECL **Eclipse Suite**	Comercial	Desarrollado por Schulumberger. Eclipse, es hoy en día uno de los softwares más utilizados en la industria petrolera, la suite incluye simuladores de aceite negro, composicionales, etc.
Nexus® **Nexus**	Comercial	Este software es distribuido por Halliburton y permite realizar análisis de simulación donde existe tanto aceite como gas. Las características del simulador prometen rapidez y certeza en los análisis de SMY.
Black Oil Applied Simulation Tool (Boast)	Libre	Desarrollado por la National Energy Technology Laboratory (NETL). BOAST ofrece un simulador de Aceite negro, tridimensional y trifásico. Fue desarrollado en el lenguaje FORTRAN y se puede descargar la aplicación desde su sitio web. → Enlace
The Open Porous Media (OPM)	Libre	Es un simulador desarrollado por ingenieros noruegos. Tiene un repositorio en Github donde los usuarios pueden acceder al recurso y un sitio oficial con la documentación del simulador. → Enlace

1.2 Propiedades de la roca

1.2.1 Porosidad

Determina la capacidad de almacenamiento de un yacimiento y se define como la fracción entre volumen de poro y el volumen total de la roca (AAPG_Wiki, Porosity, 2022).



$$\phi = \frac{volumen\ poroso}{volument\ total\ de\ roca} \quad (1)$$

Figura 3. Vista microscópica de los poros y granos de una roca.

Donde:

$\phi = porosidad\ [\%, fracción]$

Cuando el volumen de poros interconectados, presentes en una roca, contribuyen al flujo de fluidos en un yacimiento se define como porosidad efectiva. Por otro lado, la porosidad total es el espacio poroso total presente en la roca, sin importar si contribuye o no al flujo de fluidos.

1.2.2 Permeabilidad

Es la capacidad de una capa de roca para trasmitir fluidos como agua o aceite. Cuando existe un solo fluido o fase, se le conoce como permeabilidad absoluta, comúnmente se representa con la letra *k* y su unidad estándar es el Darcy (AAPG_Wiki, Permeability, 2022).

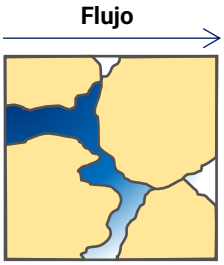


Figura 4. Flujo de fluidos a través de los poros interconectados de la roca

Permeabilidad efectiva

Es la capacidad de flujo preferencial o de transmisión de un fluido particular cuando existen otros fluidos inmiscibles en el yacimiento (Schlumberger, Effective Permeability, 2022).

Permeabilidad relativa

Es un término adimensional implementado para adaptar la ecuación de Darcy a las condiciones de flujo multifásico. Se define como la relación entre la permeabilidad efectiva de un fluido en particular y la permeabilidad absoluta de ese (Schlumberger, Relative Permeability, 2022).

$$k_{r\alpha} = \frac{k_e}{k_A} \tag{2}$$

Donde:

$k_{r\alpha}$ *= permeabilidad relativa de la fase [adim]*

k_e *= permeabilidad efectiva [md]*

k_A *= permeabilidad absoluta [md]*

1.3 Propiedades de los fluidos

1.3.1 Densidad

Es una propiedad física que se define como masa por unidad de volumen. Se suele expresar como g/cm^3 o lb_m/ft^3. En la industria petrolera la densidad del aceite se puede obtener de la gravedad especifica, el factor de volumen, la relación gas aceite, etc.

1.3.2 Viscosidad

Es la propiedad que indica la resistencia de un fluido al momento de fluir. Se suele expresar en centipoise (cp), el cual es equivalente a un milipascal-segundo. Cuando se trata de aceite, los principales factores que afectan su viscosidad son: su composición, temperatura, gas disuelto y presión.

1.4 Propiedades de la interacción roca-fluido

1.4.1 Saturación

Se define como la cantidad relativa de agua, petróleo y gas presente en los poros de una roca, comúnmente se expresa en porcentaje y representa con la letra S y el subíndice de la fase. Si las tres fases están presentes, se cumple la siguiente expresión:

$$S_w + S_o + S_g = 1 \tag{3}$$

Donde:

S_w = saturación de agua

S_o = saturación de aceite

S_g = saturación de gas

1.4.2 Presión capilar

Se define como la diferencia de presión entre dos fluidos inmiscibles que ocupan el mismo espacio poroso. Si la fase mojante es agua y la fase no mojante es aceite:

$$p_{cow} = p_o - p_w \tag{4}$$

Donde:

p_{cow} = presión capilar en las fases agua-aceite [psi]

p_w = presión en la fase mojante (agua) [psi]

p_o = presión en la fase no mojante (aceite) [psi]

Si se analiza la relación entre presión capilar y saturación de agua en un yacimiento dado, se puede descifrar dos fenómenos: imbibición y drene (ver fig. 5). La imbibición ocurre cuando hay un incremento de la fase no mojante, por ejemplo, cuando se inyecta agua al yacimiento. Mientras que el proceso de drene describe un decremento de la fase mojante.

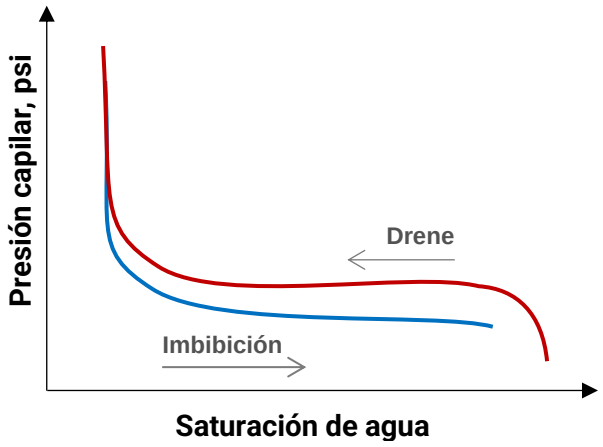


Figura 5. fenómeno de imbibición y drene. Modificado de (petrowiki.com).

2. Generación de mallas estructuradas en 2D

En este capítulo se abordan los conceptos teóricos relacionados a la generación de mallas estructuradas en 2D, de igual manera, se hace énfasis en dos métodos para generar este tipo de mallas. El primero es la Interpolación Transfinita y el segundo, el método de coordenadas de cuerpo ajustado que implementa ecuaciones diferenciales elípticas (Thomson, Soni, & Weatherill, 1999).

2.1 Mallas estructuradas

Una malla se define como la representación discreta de un medio continuo (dominio). Las mallas se componen por nodos y elementos. Éstas se pueden clasificar en: **Estructuradas y no estructuradas.**

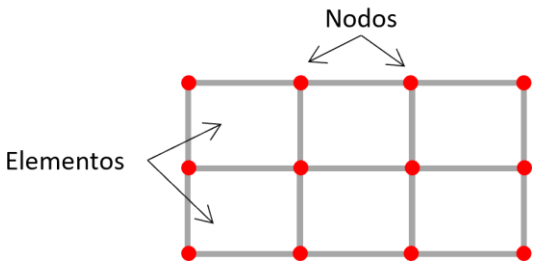


Figura 6. Ejemplo de malla. Modificado de (google.com)

Las mallas estructuradas son mallas con conectividad implícita que siguen un patrón reticular donde los nodos y elementos son fáciles de identificar. Los elementos de este tipo de mallas, por lo regular, tienen forma de cuadriláteros (2D) y hexaedros (3D).

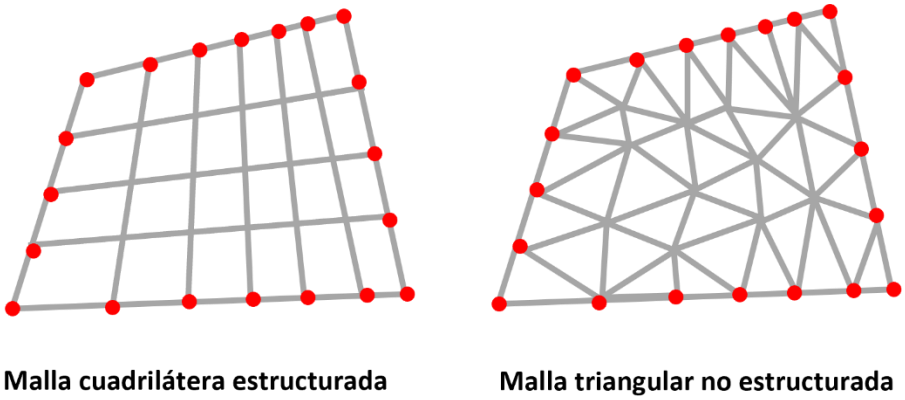


Figura 7. Ejemplo de mallas numéricas. Modificado de (Colchado, 2022)

Las mallas estructuradas son preferibles para un simulador de yacimientos, esto debido a su fácil ordenamiento, numeración y adaptación a los diferentes algoritmos de simulación, por ejemplo, IMPES o completamente mediante la iteración de Newton-Raphson. Por otro lado, una malla no estructurada también se puede implementar en un problema de simulación matemática de yacimientos, si así se desea, pero la complejidad de los algoritmos aumenta considerablemente. Para fines de este material didáctico, solo se contempla el uso de mallas estructuradas.

2.2 Métodos para generar mallas estructuradas en 2D

Existen distintos métodos de mallado estructural. Los más comunes son los algebraicos y los basados en ecuaciones diferenciales. A continuación, se mencionan algunos de ellos.

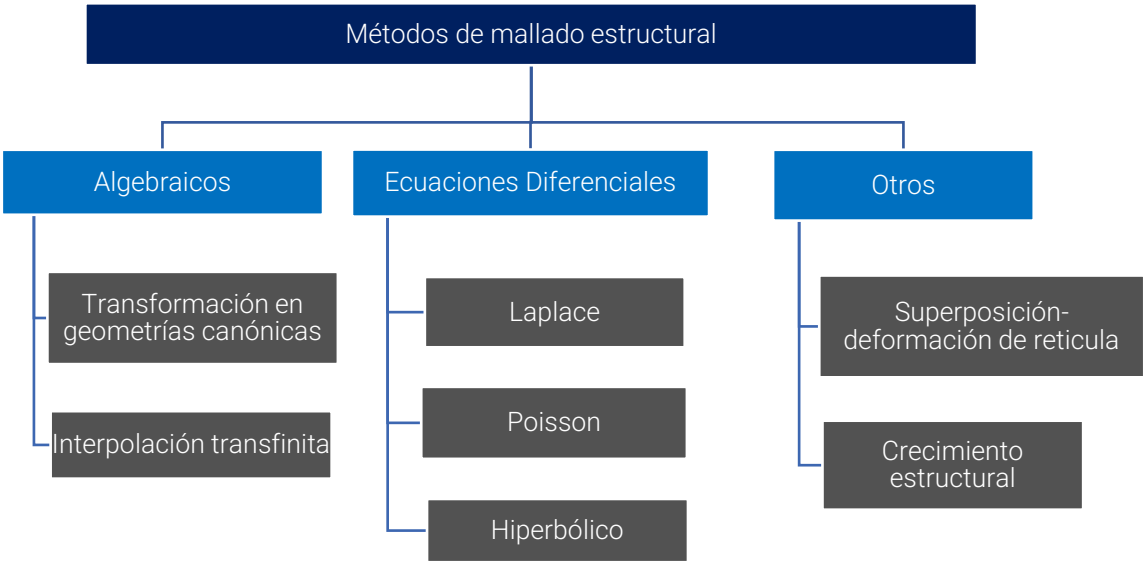


Figura 8. Mapa conceptual de los métodos de mallado estructural (Ruiz & López).

Cabe mencionar que tanto los métodos algebraicos, como los basados en ecuaciones diferenciales tienen la premisa de cambiar el sistema de coordenadas, de tal forma que sea más fácil generar los puntos que conformarán el cuerpo de la malla. Además, es posible combinar los métodos antes mencionados, por ejemplo, interpolación transfinita con uno de ecuaciones diferenciales. En este sentido, el primero funge como una aproximación inicial y el segundo para suavizar la malla.

2.2.1 Interpolación Transfinita

Consiste en aplicar una serie de ecuaciones algebraicas que generen los puntos internos de la malla en base a los puntos de sus fronteras. Para que esto sea posible, es necesario, transformar el dominio físico (x, y) a uno computacional $(\xi,\ \eta)$.

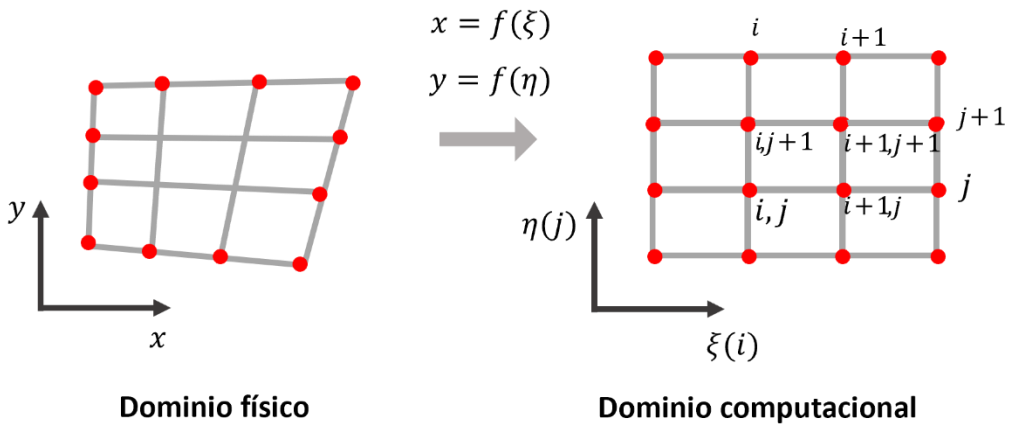


Figura 9. Transformación del dominio físico a computacional. Modificado de (google.com).

Al cambiar de coordenadas cartesianas (x, y) a curvilíneas $(\xi,\ \eta)$, se obtienen las ventajas de una malla rectangular. Esto debido a que cualquier punto en el plano cartesiano tiene un equivalente con los ejes curvilíneos $(\xi,\ \eta)$. Sin embargo, se debe ser especifico en la distribución de puntos en las cuatro fronteras del dominio físico, ya que cualquier cambio puede influir en la malla resultante.

Para comprender de mejor forma el rol que toman las fronteras en el mallado, considere un dominio bidimensional representado por cuatro curvas, dos para $\xi\ =\ constante$ y dos para $\eta\ =\ constante$.

Las curvas se definen de la siguiente manera:

$$\begin{aligned} Curva\ C1: R(0,\eta) \quad & Curva\ C3: R(1,\eta) \\ Curva\ C2: R(\xi,0) \quad & Curva\ C4: R(\xi,1) \end{aligned} \tag{5}$$

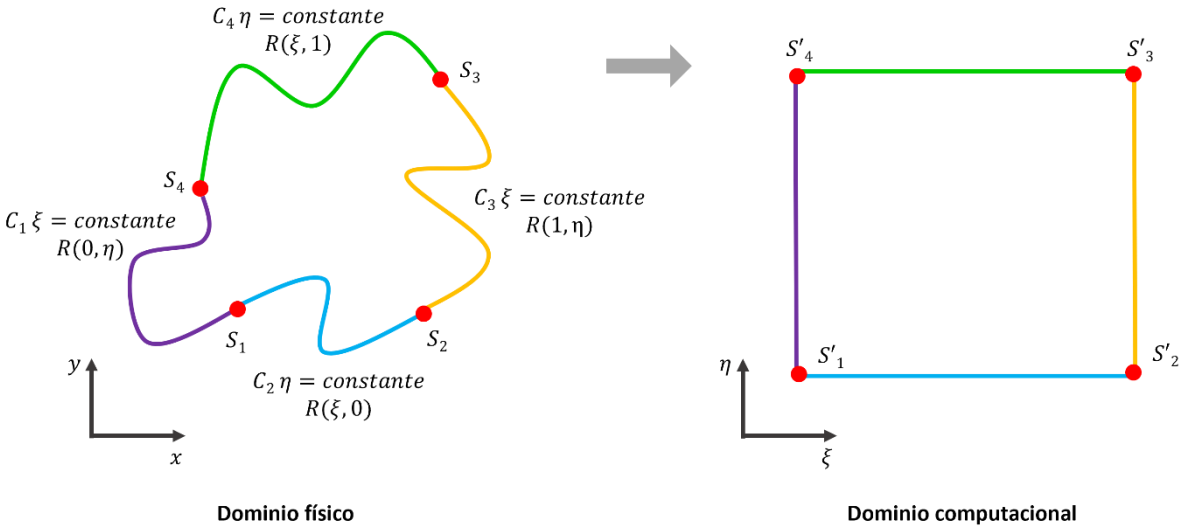


Figura 10. Contornos de malla en dominios físico y computacional. Modificado de (Selmin, 2022).

De la figura 10, se identifica que las cuatro curvas intersecan los puntos del dominio:

$$S_1 = R(0,\eta) \cap R(\xi,0) \quad S_2 = R(\xi,0) \cap R(1,\eta)$$
$$S_3 = R(1,\eta) \cap R(\xi,1) \quad S_4 = R(\xi,1) \cap R(1,\eta) \tag{6}$$

En términos generales, cada vez que se desee implementar un método estructural de mallado, se debe de realizar al menos cuatro cortes al dominio físico y a cada uno de ellos asignarles una frontera. Para el ejemplo de la figura 10, se puede asumir que el contorno de la malla inicia en el punto S1 y avanza en sentido antihorario. Lo que da origen a las siguientes relaciones:

Frontera	Puntos	Curva
Sur	S1 – S2	C2
Este	S2 – S3	C3
Norte	S3 – S4	C4
Oeste	S4 – S1	C1

Una vez identificadas las fronteras del dominio, en este caso curvas, se procede a delimitar los puntos en las fronteras, de modo que los nodos en *x* sean igual a los puntos en las fronteras Sur y Norte; mientras que los nodos en *y*, a las fronteras Este y Oeste. Para lo anterior se pueden implementar métodos de interpolación cúbicos o lineales si se tratase de polilíneas. Finalmente, se aplican las ecuaciones de interpolación transfinita.

Para encontrar las ecuaciones que realizan la tarea de la interpolación transfinita, considere que un esquema de interpolación unidimensional puede expresarse para un dominio bidimensional de la siguiente manera:

$$\mathrm{R}_a(\xi,\eta) = (1-\xi)\mathrm{R}(0,\eta) + \xi\mathrm{R}(1,\eta), \qquad 0 \leq \xi \leq 0 \tag{7}$$

Donde la ecuación (7) representa la interpolación, $\xi = constante$, similar para $\eta = constante$, la cual se expresa como:

$$R_b(\xi,\eta) = (1-\eta)R(\xi,0) + \eta R(\xi,1), \qquad 0 \leq \eta \leq 0 \tag{8}$$

Haciendo uso del operador booleano para sumar la ecuación (7) y (8), queda:

$$R_a(\xi,\eta) \oplus R_b(\xi,\eta) = R_a + R_b - R_a * R_b \tag{9}$$

Aplicando álgebra a (9) y asignando los subíndices l, r, b, t, a las fronteras left, right, bottom, top. La ecuación se puede reescribir como:

$$\begin{aligned} R(\xi,\eta) = {} & (1-\xi)R_l(\eta) + \xi R_r(\eta) + (1-\eta)R_b(\xi) + (\eta)R_t(\xi) - (1-\xi)(1-\eta)R_b(0) \\ & - (1-\xi)\eta R_t(0) - (1-\eta)\xi R_b(1) - (\xi\eta)R_t(1) \end{aligned} \tag{10}$$

Lo que es equivalente a:

$$\begin{aligned} x(\xi,\eta) = {} & (1-\xi)x_l(\eta) + \xi x_r(\eta) + (1-\eta)x_b(\xi) + (\eta)x_t(\xi) - (1-\xi)(1-\eta)x_b(0) \\ & - (1-\xi)\eta x_t(0) - (1-\eta)\xi x_b(1) - (\xi\eta)x_t(1) \end{aligned} \tag{11}$$

$$\begin{aligned} y(\xi,\eta) = {} & (1-\xi)y_l(\eta) + \xi r_r(\eta) + (1-\eta)y_b(\xi) + (\eta)y_t(\xi) - (1-\xi)(1-\eta)y_b(0) \\ & - (1-\xi)\eta y_t(0) - (1-\eta)\xi y_b(1) - (\xi\eta)y_t(1) \end{aligned} \tag{12}$$

2.2.2 Generación de mallas por medio de ecuaciones diferenciales

Para generar una malla estructurada 2D se resuelve un sistema de ecuaciones diferenciales parciales (EDPs), con el fin de localizar los puntos al interior del dominio físico. Estos sistemas pueden ser: **elípticos, parabólicos e hiperbólicos.**

Cabe aclarar que el desarrollo matemático que viene a continuación fue adaptado de la tesis de maestría: Generación de Mallas Numéricas para Geometrías Irregulares y Complejas (Teja, L., 2011).

Nótese la siguiente expresión:

$$a\,\frac{\partial^2\phi}{\partial x^2}+b\,\frac{\partial^2\phi}{\partial x\partial y}+c\,\frac{\partial^2\phi}{\partial y^2}+d\,\frac{\partial\phi}{\partial x}+e\,\frac{\partial\phi}{\partial y}+f\phi+g=0 \tag{13}$$

Donde a, b, c, d, e, f y g pueden ser funciones de las variables independientes *x* e *y*, además de la variable independiente ϕ. Esta ecuación se clasifica de acuerdo con los coeficientes:

$$Elíptica\colon\quad b^2-4ac<0$$

$$Parabólica\colon\quad b^2-4ac=0$$

$$Hiperbólica\colon\quad b^2-4ac>0$$

Para construir el sistema de EDPs con dos familias de líneas ortogonales se parte de la analogía de transferencia de calor en sólidos. Donde la primera y segunda familia de isolíneas de temperatura satisfacen las ecuaciones Laplacianas siguientes:

$$\nabla^2\xi=0 \tag{14}$$

$$\nabla^2\eta=0 \tag{15}$$

Tomando la equivalencia entre el plano físico y computacional se tienen:

$$\mathrm{x}=\mathrm{x}(\xi,\eta),\quad \mathrm{y}=\mathrm{y}(\xi,\eta),\qquad \text{para}\quad 0\le\xi\le 1,\quad 0\le\eta\le 1$$

Aplicando la regla de la cadena:

$$\frac{\partial^2}{\partial x^2}=\frac{\partial^2}{\partial\xi^2}\left(\frac{\partial\xi}{\partial x}\right)^2+2\frac{\partial^2}{\partial\xi\partial\eta}\left(\frac{\partial\xi}{\partial x}\frac{\partial\eta}{\partial x}\right)+\frac{\partial^2}{\partial\eta^2}\left(\frac{\partial\eta}{\partial x}\right)^2 \tag{16}$$

$$\frac{\partial^2}{\partial y^2}=\frac{\partial^2}{\partial\xi^2}\left(\frac{\partial\xi}{\partial y}\right)^2+2\frac{\partial^2}{\partial\xi\partial\eta}\left(\frac{\partial\xi}{\partial y}\frac{\partial\eta}{\partial y}\right)+\frac{\partial^2}{\partial\eta^2}\left(\frac{\partial\eta}{\partial y}\right)^2 \tag{17}$$

Adaptando (16) y (17) con el operador Laplaciano, se puede escribir la siguiente expresión:

$$\nabla^2 = \frac{\partial^2}{\partial \xi^2}\left(\left(\frac{\partial \xi}{\partial x}\right)^2 + \left(\frac{\partial \xi}{\partial y}\right)^2\right) + 2\frac{\partial^2}{\partial \xi \partial \eta}\left(\frac{\partial \xi}{\partial x}\frac{\partial \eta}{\partial x} + \frac{\partial \xi}{\partial y}\frac{\partial \eta}{\partial y}\right) + \frac{\partial^2}{\partial \eta^2}\left(\left(\frac{\partial \eta}{\partial x}\right)^2 + \left(\frac{\partial \eta}{\partial y}\right)^2\right) \quad (18)$$

La ecuación (18) se puede reescribir de la siguiente manera:

$$\alpha \frac{\partial^2 x}{\partial \xi^2} - 2\beta \frac{\partial^2 x}{\partial \xi\, \partial \eta} + \gamma \frac{\partial^2 x}{\partial \eta^2} = -|J|^2\left(\frac{\partial x}{\partial \xi}P + \frac{\partial x}{\partial \eta}Q\right), \quad (19)$$

$$\alpha \frac{\partial^2 y}{\partial \xi^2} - 2\beta \frac{\partial^2 y}{\partial \xi\, \partial \eta} + \gamma \frac{\partial^2 y}{\partial \eta^2} = -|J|^2\left(\frac{\partial y}{\partial \xi}P + \frac{\partial y}{\partial \eta}Q\right) \quad (20)$$

Donde P y Q son funciones para controlar el espaciamiento de los nodos de la malla y J es el Jacobiano.

Por otro lado, α, β, γ se pueden expresar de la siguiente manera:

$$\alpha\begin{bmatrix}\frac{\partial^2 x}{\partial \xi^2}\\ \frac{\partial^2 y}{\partial \xi^2}\end{bmatrix} + 2\beta\begin{bmatrix}\frac{\partial^2 x}{\partial \xi\, \partial \eta}\\ \frac{\partial^2 y}{\partial \xi\, \partial \eta}\end{bmatrix} + \gamma\begin{bmatrix}\frac{\partial^2 x}{\partial \eta^2}\\ \frac{\partial^2 y}{\partial \eta^2}\end{bmatrix} = 0 \quad (21)$$

Partiendo de las siguientes aproximaciones:

$$\frac{\partial x}{\partial \xi} \approx \frac{x_{i+1,j} - x_{i-1,j}}{2\Delta\xi} \qquad \frac{\partial y}{\partial \eta} \approx \frac{y_{i+1,j} - y_{i,j-1}}{2\Delta\eta}$$

$$\frac{\partial^2 x}{\partial \xi^2} \approx \frac{x_{i+1,j} - 2x_{i,j} + x_{i-1,j}}{\Delta\xi^2} \qquad \frac{\partial^2 y}{\partial \eta^2} \approx \frac{y_{i,j+1} - 2y_{i,j} + y_{i,j-1}}{\Delta\eta^2}$$

$$\frac{\partial^2 x}{\partial \xi \eta} \approx \frac{x_{i+1,j+1} - x_{i+1,j-1} - x_{i-1,j+1} + x_{i-1,j-1}}{4\Delta\xi\Delta\eta}$$

$$\frac{\partial^2 y}{\partial \xi \eta} \approx \frac{y_{i+1,j+1} - y_{i+1,j-1} - y_{i-1,j+1} + y_{i-1,j-1}}{4\Delta\xi\Delta\eta}$$

De las ecuaciones (19) y (20), se puede expresar lo siguiente:

$$\alpha'\left[x_{i+1,j} - 2x_{i,j} + x_{i-1,j}\right] + \gamma'\left[x_{i,j+1} - 2x_{i,j} + x_{i,j-1}\right] - 2\beta'\left[x_{i+1,j+1} - x_{i-1,j+1} - x_{i+1,j-1} + x_{i-1,j-1}\right]$$

$$\alpha'\left[y_{i+1,j} - 2y_{i,j} + y_{i-1,j}\right] + \gamma'\left[y_{i,j+1} - 2y_{i,j} + y_{i,j-1}\right] - 2\beta'\left[y_{i+1,j+1} - y_{i-1,j+1} - y_{i+1,j-1} + y_{i-1,j-1}\right] \quad (22)$$

Se asume que $\Delta\xi = \Delta\eta = 1$. Por lo que α', β', γ' de (22) se pueden reescribir de la siguiente manera:

$$\alpha' = \frac{\left(x_{i,j+1} - x_{i,j-1}\right)^2 + \left(y_{i,j+1} - y_{i,j-1}\right)^2}{(2)^2} \quad (23)$$

$$\gamma' = \frac{\left(x_{i+1,j} - x_{i-1,j}\right)^2 + \left(y_{i+1,j} - y_{i-1,j}\right)^2}{(2)^2} \quad (24)$$

$$\beta' = \frac{\left(x_{i,j+1} - x_{i-1,j}\right) + \left(x_{i,j+1} - x_{i,j-1}\right)}{(4)^2} + \frac{\left(y_{i+1,j} - y_{i-1,j}\right) + \left(y_{i,j+1} - y_{i,j-1}\right)}{(4)^2} \quad (25)$$

Para resolver las ecuaciones (19) y (20), considere una variable genérica ϕ que representa a x, y. Por lo cual, se obtienen la siguiente expresión:

$$\alpha\phi_{\xi\xi} + \gamma\phi_{\eta\eta} - 2\beta + \frac{1}{J^2}(P\phi_{\xi} + Q\phi_{\eta}) = 0 \quad (26)$$

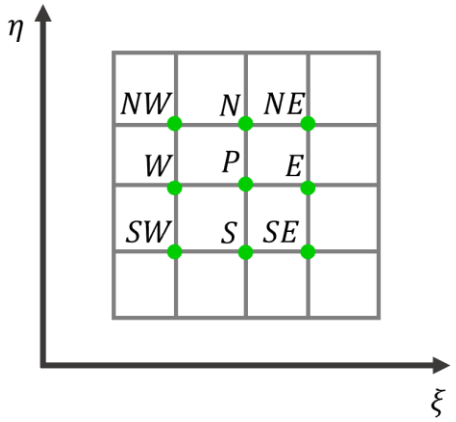


Figura 11. Plano físico transformado. Modificado de (Teja, L., 2011)

Aplicando diferencias finitas a la ecuación (26) y considerando la distribución de los puntos de la figura 11, se obtienen las siguientes expresiones:

$$\phi_{\xi\xi} = \frac{\phi_E + \phi_W - 2\phi_P}{\Delta\xi^2} \tag{27}$$

$$\phi_{\xi\xi} = \frac{\phi_E + \phi_W - 2\phi_P}{\Delta\xi^2} \tag{28}$$

$$\phi_{\eta\eta} = \frac{\phi_N + \phi_S - 2\phi_P}{\Delta\eta^2} \tag{29}$$

$$\phi_{\xi\eta} = \frac{\phi_{NE} + \phi_{SW} - 2\phi_{SE} - \phi_{NW}}{4\Delta\xi\Delta\eta} \tag{30}$$

$$\phi_{\xi} = \frac{\phi_E - \phi_W}{2\Delta\xi} \tag{31}$$

$$\phi_{\eta} = \frac{\phi_N - \phi_S}{2\Delta\eta} \tag{32}$$

Sustituyendo las ecuaciones (27), (28), (29), (30), (31) y (32) en (26), se obtiene,

$$\begin{aligned}\alpha\left(\frac{\phi_E + \phi_W - 2\phi_P}{\Delta\xi^2}\right) + \gamma\left(\frac{\phi_N + \phi_S - 2\phi_P}{\Delta\eta^2}\right) - 2\beta\left(\frac{\phi_{NE} + \phi_{SW} - 2\phi_{SE} - \phi_{NW}}{4\Delta\xi\Delta\eta}\right) \\ + \frac{P}{J^2}\left(\frac{\phi_E - \phi_W}{2\Delta\xi}\right) + \frac{Q}{J^2}\left(\frac{\phi_N - \phi_S}{2\Delta\eta}\right) = 0\end{aligned} \tag{33}$$

Agrupando los términos de la ecuación (33), se obtiene,

$$\begin{aligned}\phi_P\left(\frac{2\alpha}{\Delta\xi^2} + \frac{2\gamma}{\Delta\eta^2}\right) = \phi_E\left(\frac{\alpha}{\Delta\xi^2} + \frac{P}{2J^2\Delta\xi}\right) + \phi_W\left(\frac{\alpha}{\Delta\xi^2} - \frac{P}{2J^2\Delta\xi}\right) + \phi_N\left(\frac{\gamma}{\Delta\eta^2} + \frac{Q}{2J^2\Delta\eta}\right) \\ + \phi_S\left(\frac{\gamma}{\Delta\eta^2} - \frac{Q}{2J^2\Delta\eta}\right) - \phi_{NE}\frac{\beta}{2\Delta\xi\Delta\eta} + \phi_{SE}\frac{\beta}{2\Delta\xi\Delta\eta} \\ + \phi_{NW}\frac{\beta}{2\Delta\xi\Delta\eta} - \phi_{SW}\frac{\beta}{2\Delta\xi\Delta\eta}\end{aligned} \tag{34}$$

Por simplicidad, la ecuación (34) se puede expresar de la siguiente manera:

$$A_P\phi_P = A_E\phi_E + A_W\phi_W + A_N\phi_N + A_S\phi_S + A_{NE}\phi_{NE} + A_{SE}\phi_{SE} + A_{NW}\phi_{NW} + A_{SW}\phi_{SW} \tag{35}$$

Si $\Delta\xi = \Delta\eta = 1$, los coeficientes de la ecuación (35) se pueden obtener de la siguiente manera:

$$
\begin{aligned}
A_E &= 2(\alpha + \gamma) \\
A_E &= \alpha + \frac{P}{2J^2} \\
A_w &= \alpha - \frac{P}{2J^2} \\
A_N &= \gamma + \frac{Q}{2J^2} \\
A_S &= \gamma - \frac{Q}{2J^2} \\
A_{NE} &= -\frac{\beta}{2} \\
A_{SE} &= \frac{\beta}{2} \\
A_{NW} &= \frac{\beta}{2} \\
A_{SW} &= -\frac{\beta}{2}
\end{aligned}
\tag{36}
$$

Para fines de este material didáctico se asume $P = Q = 0$, lo que significa que no existe atracción de líneas en las fronteras de la malla. Por ende, las expresiones anteriores se simplifican a:

$$
\begin{aligned}
A_P &= 2(\alpha + \gamma) \\
A_E &= \alpha \\
A_w &= \alpha \\
A_N &= \gamma \\
A_S &= \gamma \\
A_{NE} &= -\frac{\beta}{2} \\
A_{SE} &= \frac{\beta}{2} \\
A_{NW} &= \frac{\beta}{2} \\
A_{SW} &= -\frac{\beta}{2}
\end{aligned}
\tag{37}
$$

Para aplicar las ecuaciones en la generación de mallas numéricas en dos dimensiones, primero se genera una malla inicial con Interpolación Transfinita, posteriormente, se calculan los parámetros α, β, γ, se resuelven los coeficientes $AP, AE, AW, \ldots, ASW$, se ensambla una matriz y finalmente se aplica un solver para obtener la malla refinada.

3. Desarrollo de la herramienta MallaPy_2D

En este capítulo se desarrolla el software MallaPy_2D, mismo que reúne los conceptos explicados en el capítulo 2 y permite la generación de mallas no ortogonales. Lo anterior se realiza con ayuda del lenguaje de programación Python en su versión 3.10.1.

3.1 Introducción a Python

Python es un lenguaje de programación multiparadigma, interactivo y con una sintaxis extremadamente amigable. Fue creado a finales de los años 1980s por Guido van Rossum, como sucesor del lenguaje ABC. Desde su versión 0.9 en 1991, Python se ha convertido en uno de los lenguajes de programación más populares y utilizados por la comunidad científica y tecnológica.

3.1.1 ¿Qué se puede hacer con Python?

Entre los usos más populares de este lenguaje destaca el análisis y ciencia de datos, desarrollo web, inteligencia artificial, educación, etc. Cada uno de estos campos tiene una librería o framework como común denominador.

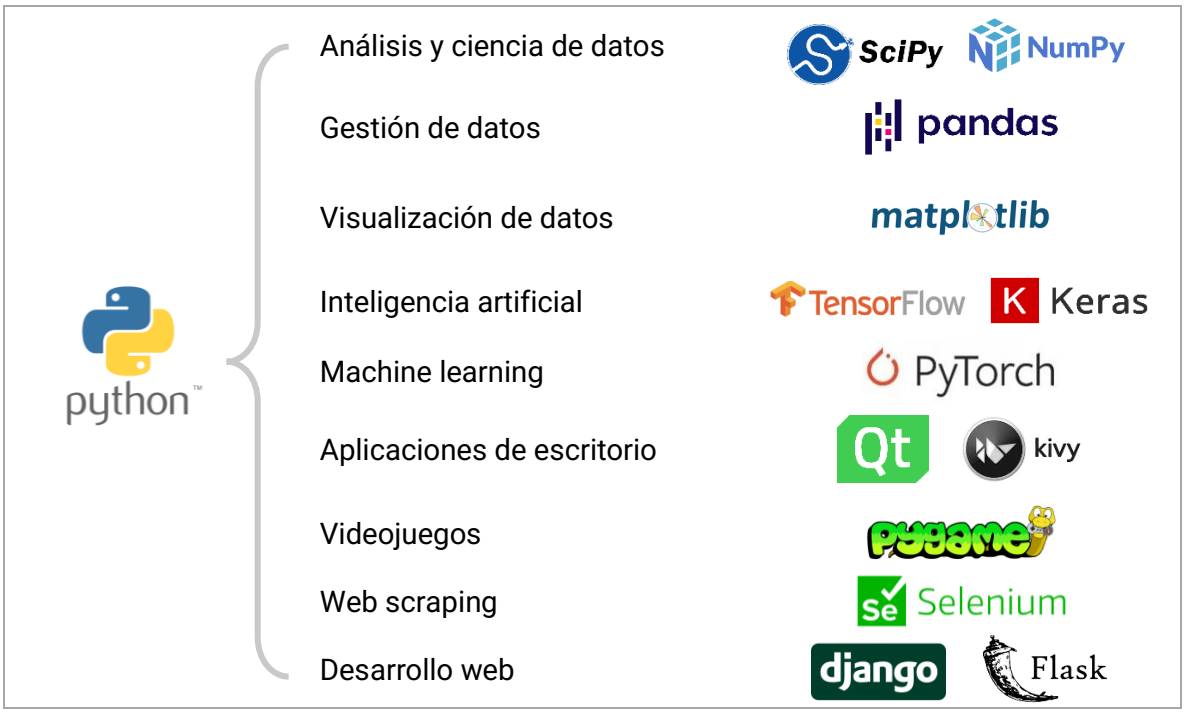


Figura 12. Usos comunes de Python y sus librerías

3.1.2 Entorno de desarrollo

Existen distintos editores de texto e IDEs para programar con Python. Algunos de los más populares son:

- Sublime Text
- Visual Studio Code
- Anaconda, Spyder
- Pycharm

Todos los antes mencionados son muy recomendados, sin embargo, para la realización de la herramienta MallaPy_2D, se utilizará Visual Studio Code como entorno de desarrollo. Su enlace se encuentra disponible en el sitio **code.visualstudio.com**

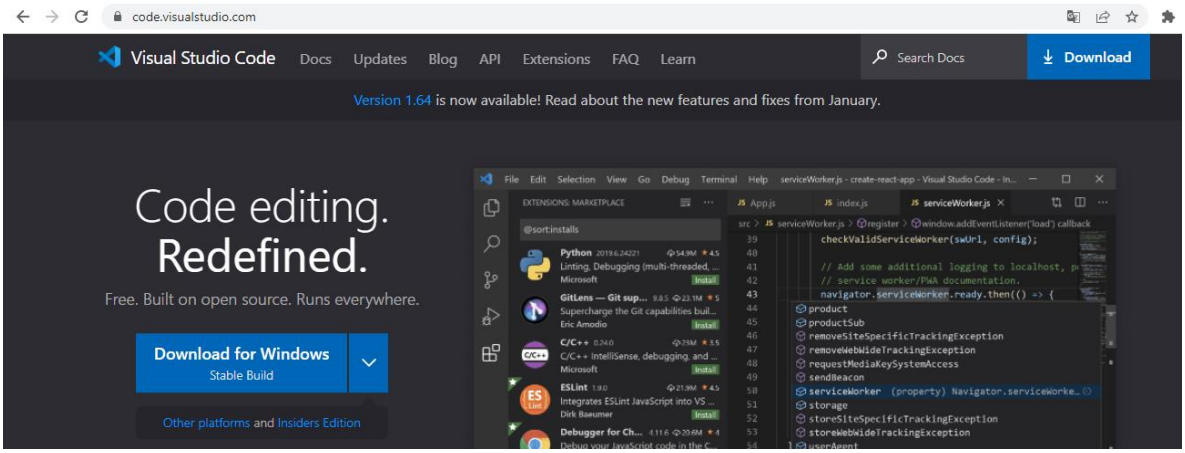


Figura 13. Entorno de desarrollo con VS Code.

Una vez descargado el archivo, ejecuta el programa y da clic en siguiente para completar la instalación. Para optimizar el entorno de desarrollo se recomienda la instalación de extensiones como Pylance y Python.

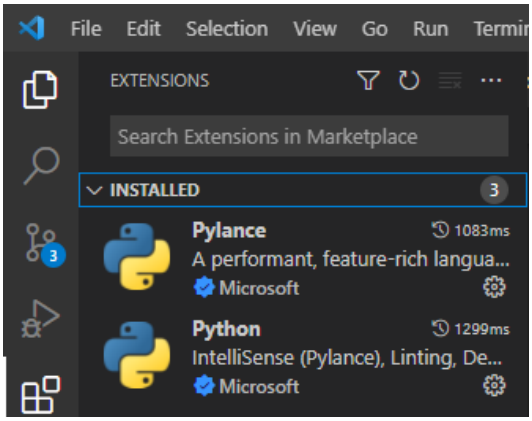


Figura 14. Extensiones de VS Code para Python

3.1.3 Instalación de Python

Para descargar la versión más reciente de Python, nos dirigimos al sitio web oficial **python.org** y damos clic en el botón de descarga.

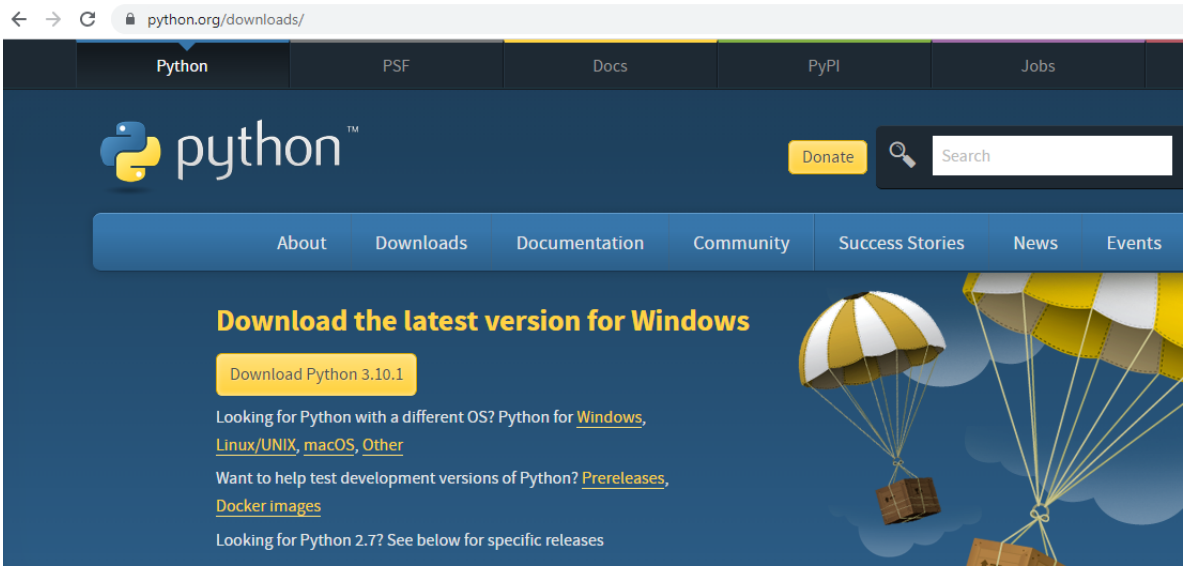


Figura 15. Sitio web oficial para descargar Python

Una vez descargado el archivo, ejecuta el programa, selecciona la opción de opción de **Add Python to PATH** y da clic en instalar.

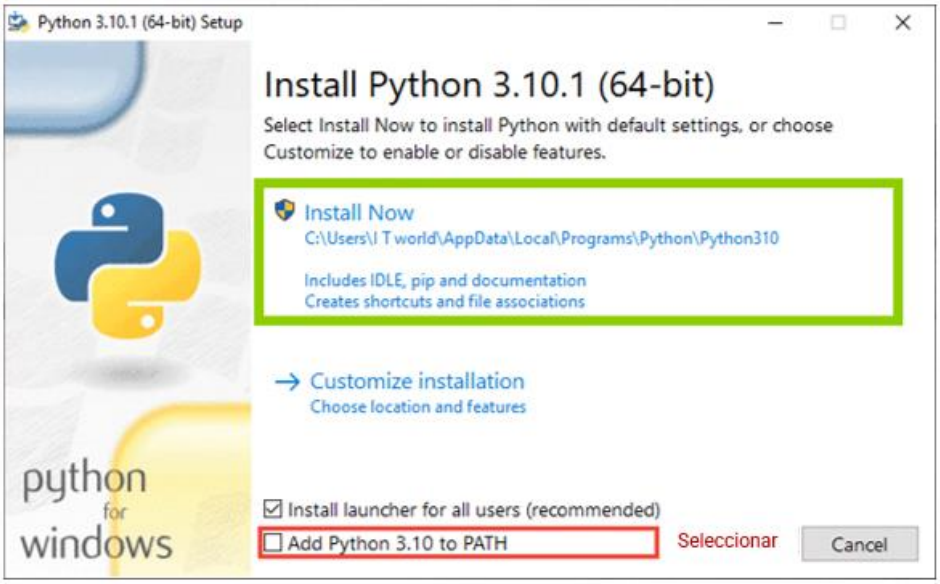


Figura 16. Instalación de Python 3.10.1

3.1.4 Instalación de librerías con pip

El instalador de paquetes de Python (pip) nos permite instalar todas las librerías o frameworks que se requieran. Para ejecutar este comando nos dirigimos a la terminal o símbolo del sistema (cmd) y ejecutamos el siguiente comando:

```
pip install librería_a_instalar
```

Las librerías que se utilizaron para el desarrollo de este proyecto son:

Tabla 2. Librerías implementadas para el desarrollo de software

Manejo de datos	Interfaz grafica	Rendimiento
numpy **pandas** **matplolib**	**PySide6** **PyInstaller**	**numba**

3.2 Algoritmo de MallaPy_2D

A continuación, se describen las instrucciones que permitirán la generación de mallas estructuradas en 2D.

```
1- Definir los puntos en las fronteras del dominio

2- Aplicar Interpolación Transfinita
      Calcular ξ, η
      Calcular los puntos interiores de malla con x = f(ξ), y = f(η)
      Guardar malla inicial

3- Aplicar ecuaciones diferenciales parciales elípticas
      Calcular coeficientes en x
      Resolver mediante algoritmo de Thomas
      Calcular coeficientes en y
      Resolver mediante algoritmo de Thomas
      Exportar malla refinada
```

Lo anterior se puede llevar a cabo con las ecuaciones descritas en el capítulo 2. No obstante, para facilitar la generación de las mallas se plantea el uso de una interfaz gráfica de usuario (GUI). Misma que administrará cada algoritmo y funciones necesarias.

3.3 Generación de interfaces graficas con PySide6

PySide6 es el módulo oficial de Qt para Python, esta librería permite crear una gran variedad de interfaces gráficas y facilitar el uso de algún script de Python. A continuación, se enlistan los widgets más comunes que se pueden implementar.

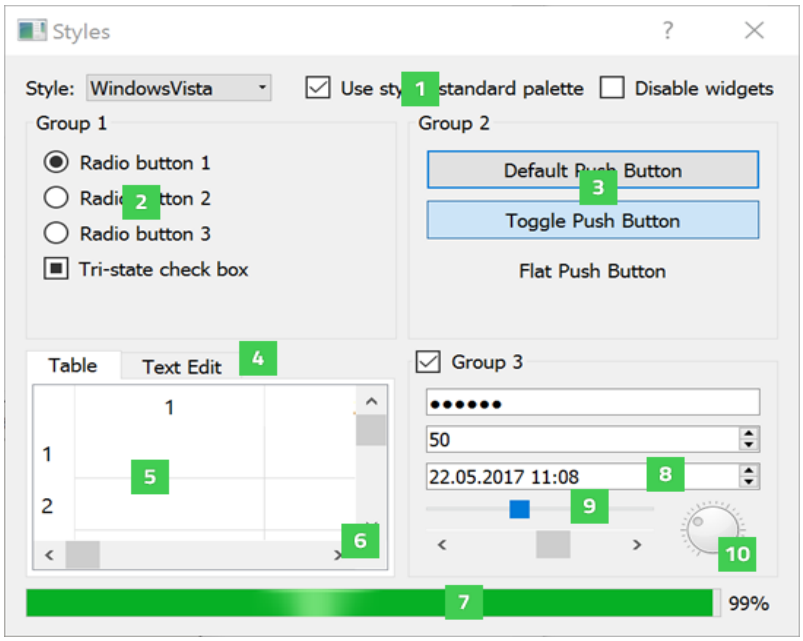


Figura 17. Widgets más utilizados para le creación de interfaces

[1] QCheckBox – casilla de verificación

[2] QRadioButton – botón de opción

[3] QPushButton – botón

[5] QTableWidget – tabla

[6] QScrollBar – barra de desplazamiento

[7] QProgressBar – barra de progreso

Otros widgets muy utilizados son las etiquetas (QLabel), cajas de texto (QLineEdit), layouts para espaciamiento (QGirdLayout, QVBoxLayout, QHBoxLayout), separadores (QSplitter) y frames (QFrame).

3.3.1 Iniciar una interfaz gráfica (GUI) con PySide6

El siguiente código permite iniciar una interfaz gráfica con un botón dentro de ella. Basta con definir una clase principal (*MyWidget*) que representa la interfaz gráfica y mostrarla al ejecutar la función "*__main__*".

1_mi_primera_gui.py

```
import sys

from PySide6.QtWidgets import QWidget, QApplication, QPushButton

class MyWidget(QWidget):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 100, 400, 200)
        self.setWindowTitle("Mi primera GUI")
        self.boton = QPushButton("Botón",self)


if __name__ == "__main__":
    app = QApplication(sys.argv)
    widget = MyWidget()
    widget.show()
    sys.exit(app.exec())
```

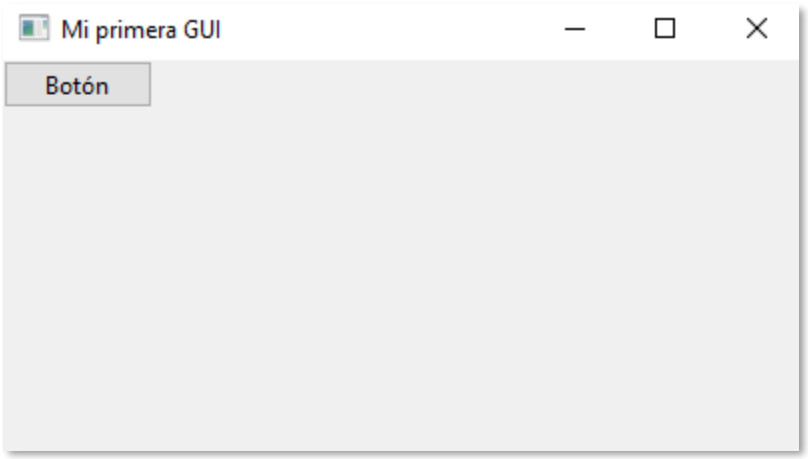


Figura 18. Interfaz sencilla con botón

Para colocar más de un widget dentro de la aplicación, es necesario implementar un *layout*, éste permitirá el espaciamiento uniforme entre cada componente de la interfaz.

3.3.2 GUI con layout

El layout puede alinear widgets en dirección vertical (QVBoxLayout), horizontal (QHBoxLayout) o ambas (QGridLayout).

2_ gui_con_layout.py

```
import sys
from PySide6.QtWidgets import (QProgressBar, QWidget, QApplication,
QVBoxLayout, QPushButton, QLineEdit)

class MyApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 100, 400, 200)
        self.setWindowTitle("GUI con layout")
        #    Definir el layout
        self.layout = QVBoxLayout()
        self.setLayout(self.layout)
        #    Definir widgets
        self.boton = QPushButton("Botón", self)
        self.caja_texto = QLineEdit(self)
        self.barra_progreso = QProgressBar(self)
        #    Agregar widgets al layout
        self.layout.addWidget(self.boton)
        self.layout.addWidget(self.caja_texto)
        self.layout.addWidget(self.barra_progreso)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    myApp = MyApp()
    myApp.show()
    sys.exit(app.exec())
```

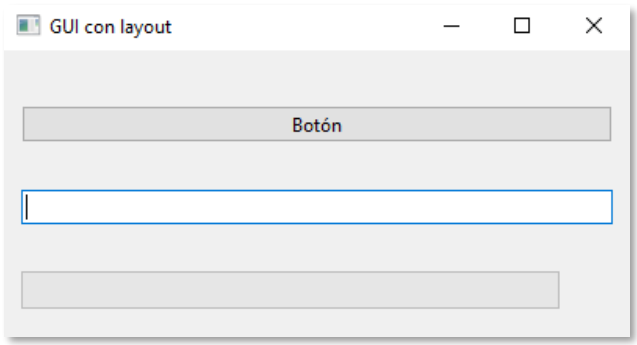


Figura 19. Uso de QVBoxLayout para espaciamiento de widgets

3.3.3 Llamar a una función al hacer clic en botón

Para evitar escribir cada uno de los widgets que se desean importar, es suficiente con escribir un *. Por otro lado, para llamar a una función, basta con conectar el evento click del widget con la función de interés.

3_ gui_con_funcion.py

```
import sys
from PySide6.QtWidgets import *

class MyApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("GUI con función")
        self.setGeometry(100, 100, 300, 150)
        self.layout = QGridLayout()
        self.setLayout(self.layout)
        self.etiqueta_titulo = QLabel("Metros a pies", self)
        self.caja_texto = QLineEdit(self)
        self.boton = QPushButton("Convertir", self)
        self.etiqueta_resultado = QLabel("Resultado", self)

        self.layout.addWidget(self.etiqueta_titulo, 0, 0, 1, 2)
        self.layout.addWidget(self.caja_texto, 1, 0, 1, 1)
        self.layout.addWidget(self.boton, 1, 1, 1, 1)
        self.layout.addWidget(self.etiqueta_resultado, 2, 0, 1, 2)
        #    En caso de click llamamos a la funcion
        self.boton.clicked.connect(self.calcular_metros_a_pies)

    def calcular_metros_a_pies(self):
        metros  = self.caja_texto.text()
        if (metros):
            pies = float(metros)*3.28084
            self.etiqueta_resultado.setText(str(pies) + " pies")

if __name__ == '__main__':
    app = QApplication(sys.argv)
    myApp = MyApp()
    myApp.show()
    sys.exit(app.exec())
```

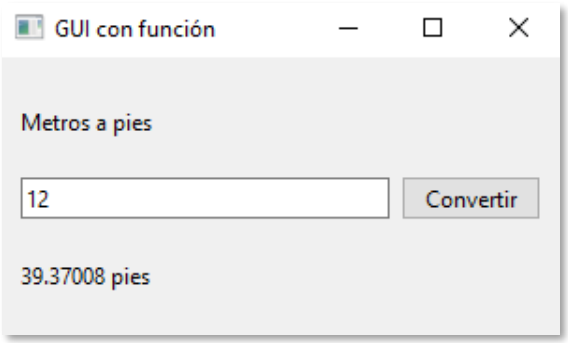


Figura 20. GUI con función interactiva

3.3.4 Canvas con matplotlib

Para colocar un gráfico de matplotlib en una interfaz gráfica generada con Qt, es necesario definir un canvas. Este widget tiene la capacidad de administrar las figuras que se generen al correr una determinada función. Si se desea graficar en múltiples ocasiones, es necesario borrar la figura dentro canvas con el comando. clf*()* y mostrarlo con *.draw()*

4_canvas_con_matplotlib.py

```
import sys
from PySide6.QtWidgets import *
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg
import matplotlib.pyplot as plt
import numpy as np

class MyApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 100, 450, 300)
        self.setWindowTitle("GUI con matplotlib")
        self.layout = QGridLayout()
        self.setLayout(self.layout)
        self.distancia_x = QLineEdit(self)
        self.distancia_y = QLineEdit(self)
        self.nodos_x = QLineEdit(self)
        self.nodos_y = QLineEdit(self)
        self.boton = QPushButton("Generar malla", self)
        self.figura = plt.figure()
        self.canvas = FigureCanvasQTAgg(self.figura)
        self.distancia_x.setPlaceholderText("distancia en x")
        self.distancia_y.setPlaceholderText("distancia en y")
```

```
        self.nodos_x.setPlaceholderText("nodos en x")
        self.nodos_y.setPlaceholderText("nodos en y")
        self.layout.addWidget(self.distancia_x, 0, 0, 1, 1)
        self.layout.addWidget(self.distancia_y, 0, 1, 1, 1)
        self.layout.addWidget(self.nodos_x, 1, 0, 1, 1)
        self.layout.addWidget(self.nodos_y, 1, 1, 1, 1)
        self.layout.addWidget(self.boton, 2, 0, 1, 2)
        self.layout.addWidget(self.canvas, 3, 0, 1, 4)
        self.boton.clicked.connect(self.generar_malla_rectangular)

    def generar_malla_rectangular(self):
        nx, ny = self.nodos_x.text(), self.nodos_y.text()
        lx, ly = self.distancia_x.text(), self.distancia_y.text()
        if (nx != '' and ny != '' and lx != '' and ly != ''):
            malla_x, malla_y = np.meshgrid(
                np.linspace(0,float(lx), int(nx)),
                np.linspace(0,float(ly), int(ny)))

            self.figura.clf()
            ax = self.figura.add_subplot(111)
            ax.plot(malla_x, malla_y, 'g-',
                malla_x.transpose(), malla_y.transpose(), 'g-')
            ax.axis('equal')
            self.canvas.draw()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    myApp = MyApp()
    myApp.show()
    sys.exit(app.exec())
```

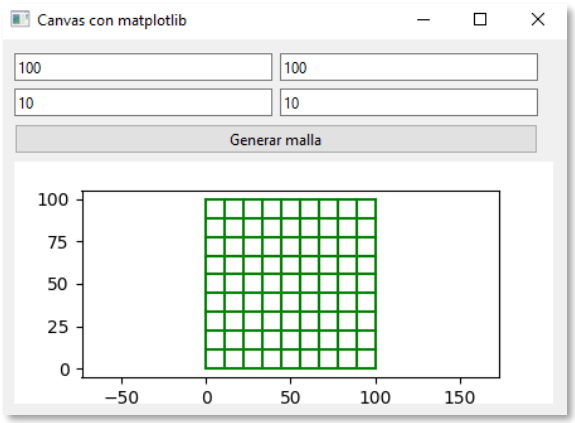


Figura 21. Uso de matplotlib dentro de una interfaz

3.3.5 Eventos en una interfaz gráfica

Un evento se puede generar a partir de la interacción del usuario con el ratón o presionar teclas específicas. A continuación, se muestra un ejemplo donde el usuario puede dibujar una polilínea dentro de la figura/canvas y ésta se puede eliminar al presionar Ctrl + X.

5_gui_con_eventos.py

```
import sys
from PySide6.QtWidgets import *
from PySide6.QtCore import Qt
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg
from matplotlib.backend_bases import MouseButton
import matplotlib.pyplot as plt

class MyApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 100, 400, 250)
        self.setWindowTitle("mouseEvent & keyPressEvent")
        self.layout = QVBoxLayout()
        self.setLayout(self.layout)
        self.etiqueta = QLabel(self)
        self.figura = plt.figure()
        self.canvas = FigureCanvasQTAgg(self.figura)
        self.layout.addWidget(self.etiqueta)
        self.layout.addWidget(self.canvas)
        self.establecer_grafico()

    def establecer_grafico(self):
        self.figura.clf()
        self.ax = self.figura.add_subplot(111)
        self.ax.set_xlim(0, 100)
        self.ax.set_ylim(0, 100)
        self.puntos_x, self.puntos_y  = [], []
        self.polilinea, = self.ax.plot(self.puntos_x, self.puntos_y, 'co-
')
        plt.connect('motion_notify_event', self.on_move)
        plt.connect('button_press_event', self.on_click)
        self.canvas.draw()
```

```
    def on_move(self, event):
        if event.inaxes:
            x, y = event.xdata, event.ydata
            self.etiqueta.setText(
                "Coordenadas: "
                + str(round(x)) + ","
                + str(round(y))
            )

    def on_click(self, event):
        if event.button is MouseButton.LEFT:
            x, y = event.xdata, event.ydata
            self.puntos_x.append(x)
            self.puntos_y.append(y)
            self.polilinea.set_xdata(self.puntos_x)
            self.polilinea.set_ydata(self.puntos_y)
            self.canvas.draw()

    def keyPressEvent(self, event):
        if event.key() == (Qt.Key.Key_Control and Qt.Key.Key_X) :
            self.etiqueta.setText("Se han borrado los puntos")
            self.establecer_grafico()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    myApp = MyApp()
    myApp.show()
    sys.exit(app.exec())
```

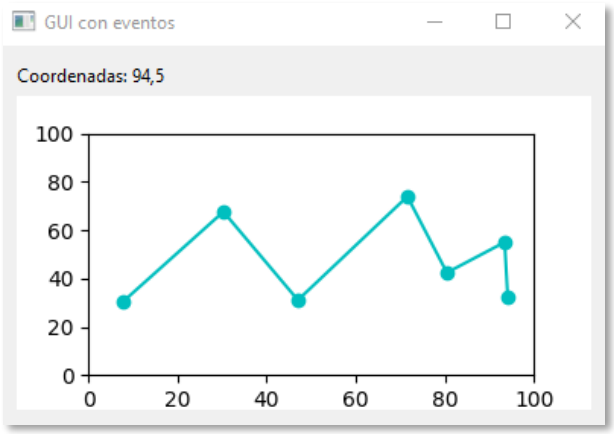


Figura 22. Grafica interactiva con eventos al clic

3.3.6 Signals para conectar dos interfaces

Una señal puede ser de utilidad si se desea enviar un parámetro de un formulario a otro. Para lograrlo basta con definir una interfaz principal, importar la GUI secundaria y en caso de que una señal se active, referenciar a una función.

6_gui_con_signals.py

```
import sys
from PySide6.QtWidgets import *
from  gui_secundaria import GuiSecundaria

class MyApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 100, 250, 200)
        self.setWindowTitle("GUI Principal")
        self.layout = QGridLayout()
        self.setLayout(self.layout)
        self.boton = QPushButton("Abrir GUI Secundaria",self)
        self.etiqueta = QLabel(self)
        self.layout.addWidget(self.boton, 0, 0, 1, 1)
        self.layout.addWidget(self.etiqueta, 1, 0, 1, 1)
        self.boton.clicked.connect(self.abrir_gui_secundaria)

        #    Definimos a la GUI secundaria
        self.nueva_ventana = GuiSecundaria()
        self.nueva_ventana.signal.connect(self.recibir_valores)

    def abrir_gui_secundaria(self):
        self.nueva_ventana.show()

    def recibir_valores(self,valor):
        texto = ( "Has recibido los valores: "+ valor[0] + ", " +
valor[1] )
        self.etiqueta.setText(texto)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    myApp = MyApp()
    myApp.show()
    sys.exit(app.exec())
```

gui_secundaria.py

```
from PySide6.QtWidgets import*
from PySide6.QtCore import Signal

class GuiSecundaria(QWidget):
    signal = Signal(tuple)
    def __init__(self):
        super().__init__()
        self.setGeometry(400, 100, 250, 100)
        self.setWindowTitle("GUI Secundaria")
        self.layout = QGridLayout()
        self.setLayout(self.layout)
        self.caja_1 = QLineEdit(self)
        self.caja_2 = QLineEdit(self)
        self.boton = QPushButton("Enviar", self)
        self.layout.addWidget(self.caja_1, 0, 0, 1, 1)
        self.layout.addWidget(self.caja_2, 0, 1, 1, 1)
        self.layout.addWidget(self.boton, 1, 0, 2, 1)
        self.boton.clicked.connect(self.enviar_parametros)

    def enviar_parametros(self):
        valor_1, valor_2 = self.caja_1.text(), self.caja_2.text()
        if (valor_1 and valor_2):
            valores = valor_1, valor_2
            #    Mandamos la señal a la GUI principal
            self.signal.emit(valores)
```

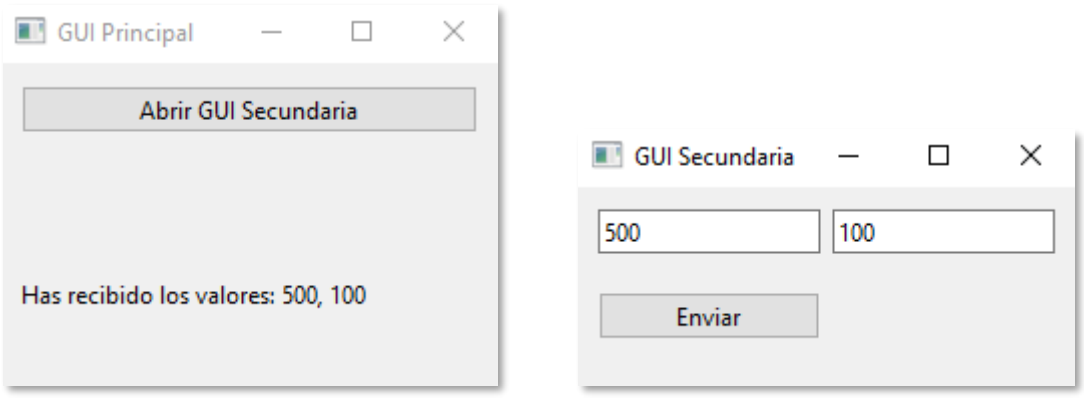


Figura 23. Uso de señales en interfaces graficas

En la expresión *signal = Signal(tuple),* este último puede ser sustituido por otro tipo de valor como enteros, flotantes, arreglos, etc.

3.3.7 QThread para clases externas

Uno de los inconvenientes al ejecutar una tarea larga con una interfaz gráfica es el congelamiento, es decir, la GUI se bloquea y aparenta un desempeño deficiente. Para disminuir este problema se puede implementar un QThread, el cual corre una clase externa a la par de la GUI y la mantiene actualizada.

7_gui_con_qthread.py

```
import sys
from PySide6.QtWidgets import *
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg
import matplotlib.pyplot as plt
import external_qthread as thread

class MyApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 100, 450, 300)
        self.setWindowTitle("GUI con QThread")
        self.layout = QVBoxLayout()
        self.setLayout(self.layout)
        self.boton = QPushButton("Ejecutar función", self)
        self.barra = QProgressBar(self)
        self.fig = plt.figure()
        self.canvas = FigureCanvasQTAgg(self.fig)
        self.layout.addWidget(self.boton)
        self.layout.addWidget(self.barra)
        self.layout.addWidget(self.canvas)
        self.establecer_grafico()
        self.boton.clicked.connect(self.correr_clase_externa)

    def establecer_grafico(self):
        ax = self.fig.add_subplot(111)
        ax.axis('equal')
        self.ln, = plt.plot([], [], 'r-')
        ax.set_xlim(-500, 500)
        ax.set_ylim(-500, 500)
        self.canvas.draw()

    def correr_clase_externa(self):
        self.funcion_externa = thread.External()
        self.funcion_externa.parametros.connect(self.actualizar_widgets)
        self.funcion_externa.start()
```

```
    def actualizar_widgets(self, x, y, avance):
        self.ln.set_data(x, y)
        self.barra.setValue(avance)
        self.canvas.draw()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    myApp = MyApp()
    myApp.show()
    sys.exit(app.exec())
```

external_qthread.py

```
from PySide6.QtCore import QThread, Signal
import numpy as np

class External(QThread):
    parametros = Signal(np.ndarray, np.ndarray, int)

    def run(self):
        for i in range(0, 360):
            theta = np.radians(np.linspace(0, i*3, 300))
            r = theta**2
            x = r*np.cos(theta)
            y = r*np.sin(theta)

            avance =  int(((i+1)/360)*100)
            self.parametros.emit( x, y, avance )

    def actualizar_widgets(self, x, y, avance):
        self.ln.set_data(x, y)
        self.barra.setValue(avance)
        self.canvas.draw()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    myApp = MyApp()
    myApp.show()
    sys.exit(app.exec())
```

Uno de los mayores inconvenientes de QThread es no poder detener la ejecución de la tarea/función una vez iniciada. De este modo, el usuario deberá esperar hasta que se termine la ejecución y así volver a realizar una corrida más. Por otro lado, a pesar de implementar un Thread, es posible presenciar un efecto de congelamiento. Esto dependerá de los recursos y memoria de cada ordenador.

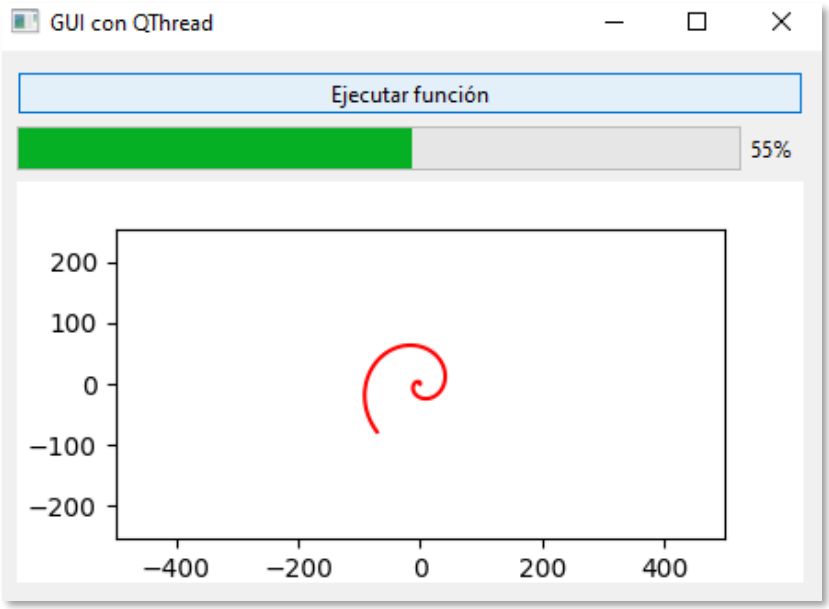


Figura 24. Uso de QThread para funciones en segundo plano

3.3.8 Hoja de estilos

Una hoja de estilos permite estilizar los elementos de una interfaz gráfica. En el caso de PySide6 se puede importar un archivo con terminación .py y finalmente aplicar el estilo a la aplicación principal.

8_gui_con_estilos.py

```
import sys
from PySide6.QtWidgets import *
from hoja_estilos import style

class MyApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 100, 300, 150)
        self.setWindowTitle("GUI con estilos")
```

```
        self.layout = QVBoxLayout()
        self.setLayout(self.layout)
        self.etiqueta = QLabel("GUI con hoja de estilos", self)
        self.boton = QPushButton("Enviar", self)
        self.caja_texto = QLineEdit(self)
        self.layout.addWidget(self.caja_texto)
        self.layout.addWidget(self.etiqueta)
        self.layout.addWidget(self.boton)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    app.setStyleSheet(style)
    myApp = MyApp()
    myApp.show()
    sys.exit(app.exec())
```

hoja_estilos.py

```
style = '''
QWidget {
    font-size: 15px;
    font-weight: bold;
}

QLineEdit {
    color: gray;
    padding: 5px;
    border: 1px solid gray;
}

QPushButton {
    border: 0;
    color: white;
    background-color: #F14505;
    padding: 5px;
}
QPushButton::hover {
    background-color: #B73708 ;
}

'''
```

Otra alternativa para modificar el estilo de los widgets es mediante la propiedad *setObjectName("my-widget")*, misma que actúa como un índice y puede ser identificada en la hoja de estilos al agregar el símbolo "#". Siguiendo el ejemplo, quedaría: *#my-widget { estilos }*

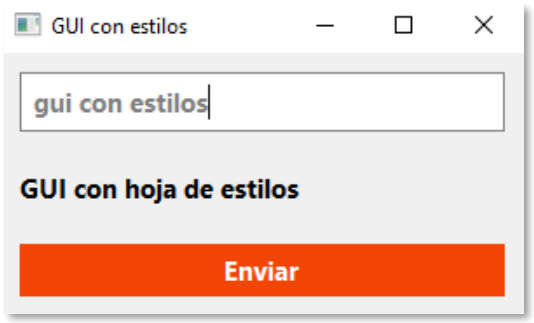


Figura 25. GUI con hoja de estilos

3.4 Interfaz gráfica de MallaPy_2D

La interfaz gráfica resultante adapta los conceptos teóricos y prácticos explicados con anterioridad. Para la generación del ejecutable se implementó la librería PyInstaller.

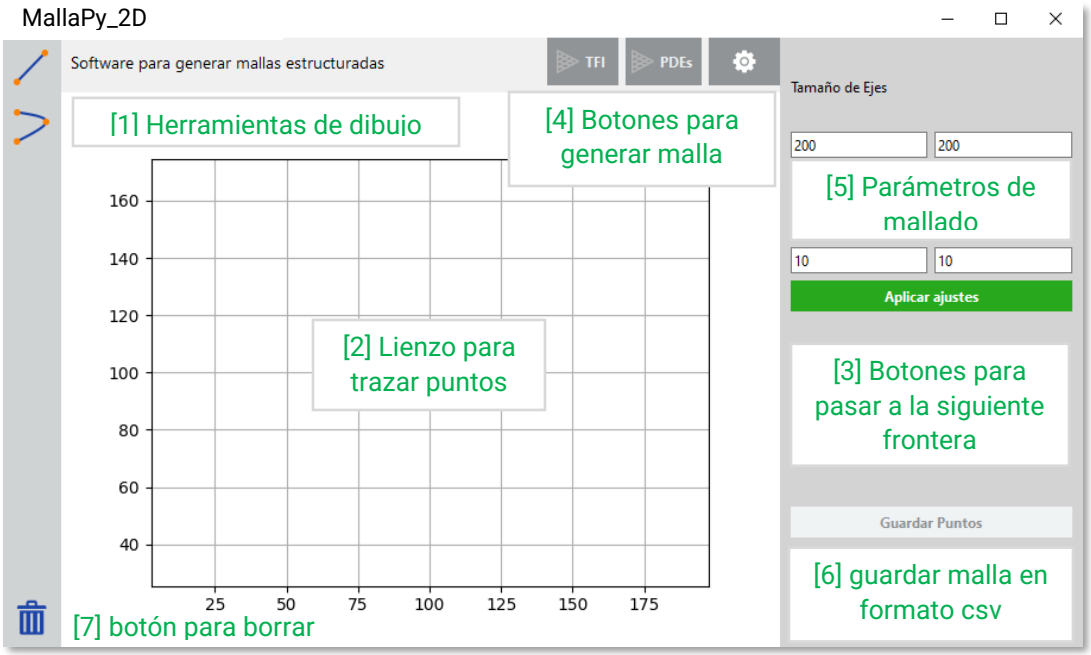


Figura 26. Interfaz gráfica de MallaPy_2D

4. Utilización de la herramienta MallaPy_2D

En este capítulo se muestra el uso básico del software, algunos ejemplos de las mallas no ortogonales que el usuario puede generar y la aplicación de estas en un problema de simulación matemática de yacimientos.

4.1 Manual de usuario de MallaPy_2D

A continuación, se mencionan las instrucciones de uso del software.

1. Seleccionar una de las herramientas de dibujo: polilínea o curva
2. Trazar los puntos de control de la frontera Sur
3. Para pasar a la frontera Este, dar **Ctrl + X** o clic en el botón de fronteras
4. Repetir el paso 2 y 3 hasta llegar a la frontera Oeste
5. Unir los puntos del contorno de la malla con **Ctrl + X** o clic en el botón de fronteras
6. Generar la malla con el botón **TFI**
7. Refinarla con el botón **PDEs**
8. Finalmente, modificar los parámetros de mallado, guardar lo puntos de la malla o eliminarla

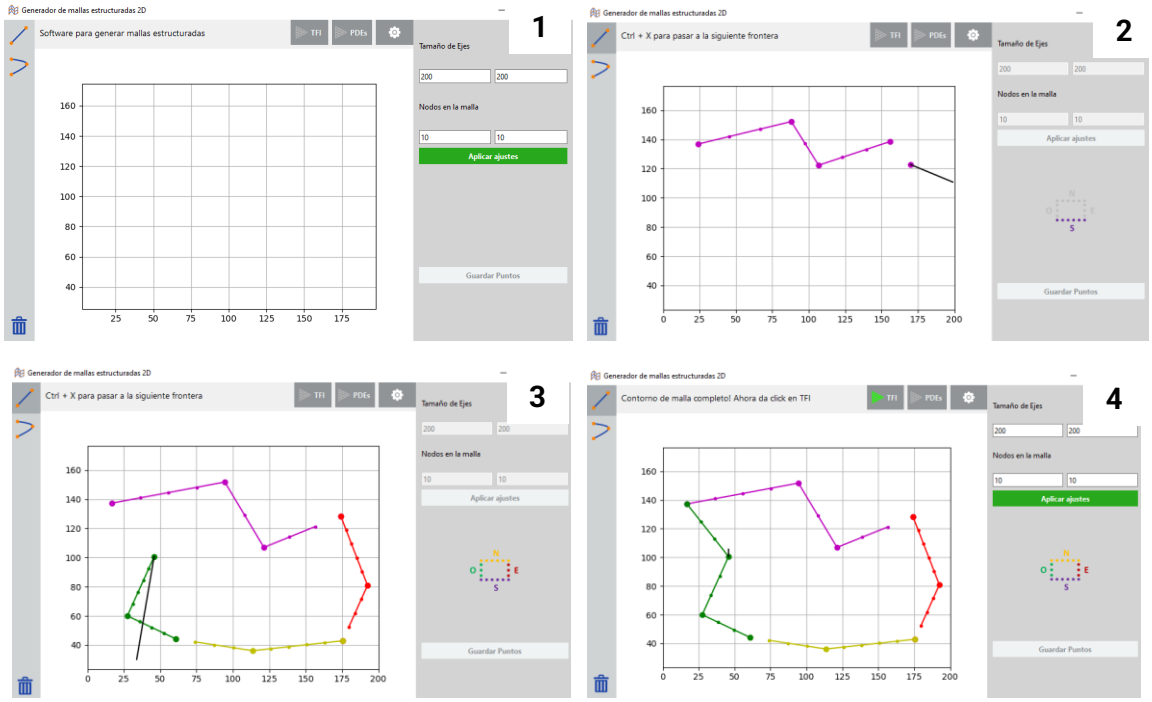


Figura 27. Proceso para generar una malla estructurada no ortogonal con MayaPy_2D

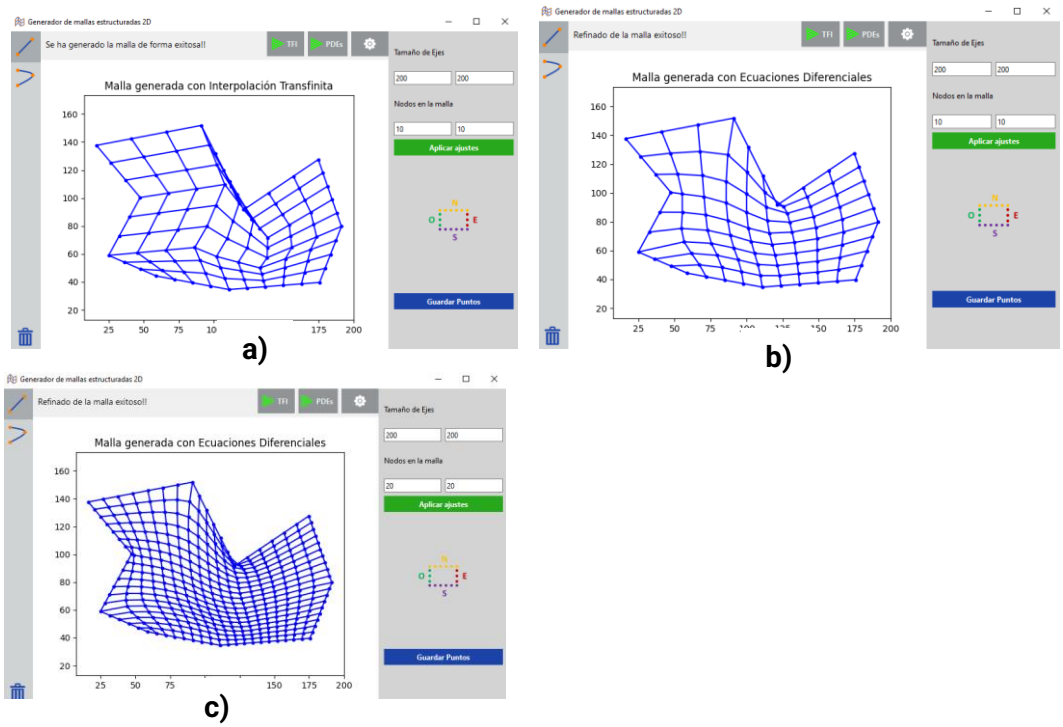


Figura 28. Mallas resultantes generadas con MallaPy_2D. a) Malla con interpolación transfinita, b) Malla refinada con EDPs, c) Malla refinada con más nodos

4.2 Mallas 2D con vista frontal de estructuras geológicas

Las estructuras geológicas que permiten la acumulación de hidrocarburos son conocidas como trampas. Éstas pueden estructurales o estratigráficas. A continuación, se muestran imágenes reales de estas estructuras y las mallas generadas con MallaPy_2D.

4.2.1 Anticlinal

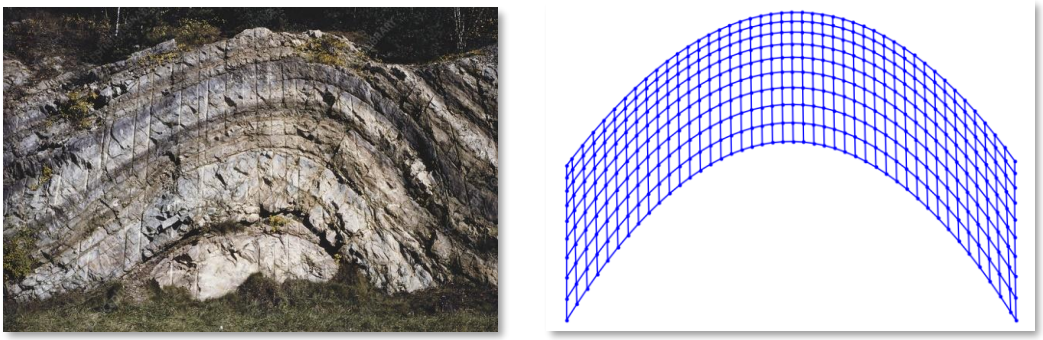


Figura 29. Malla de anticlinal

4.2.2 Fallas

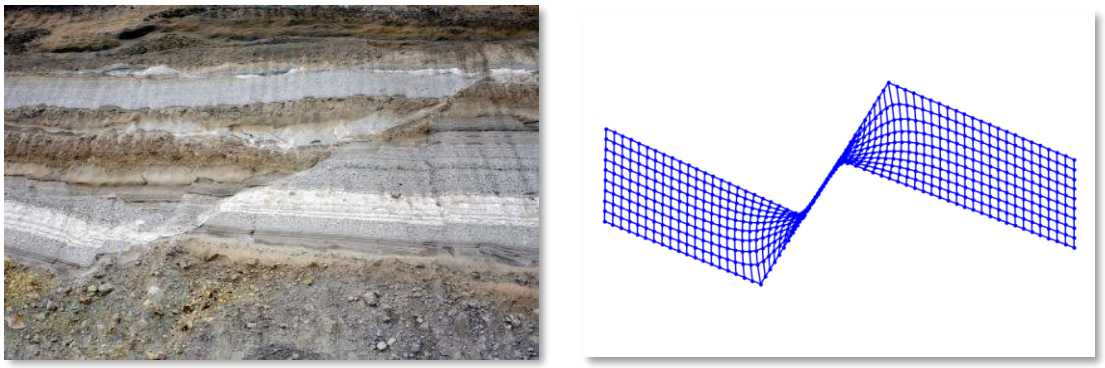


Figura 30. Malla de falla geológica

4.2.3 Trampa tipo cuña

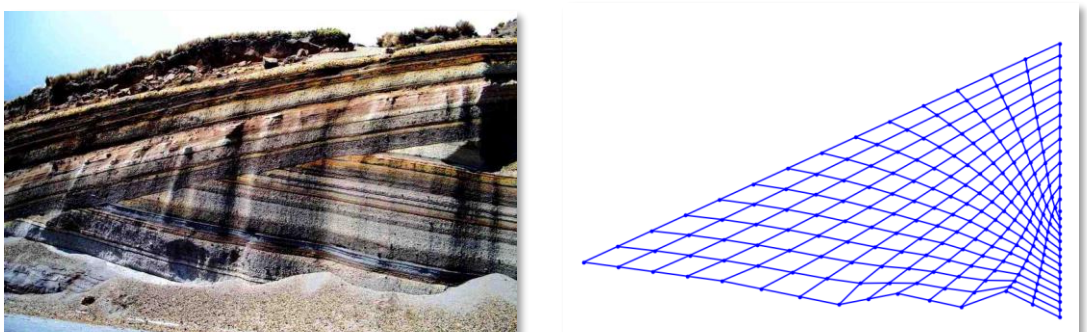


Figura 31. Malla de estructura tipo cuña

4.2.4 Discordancias

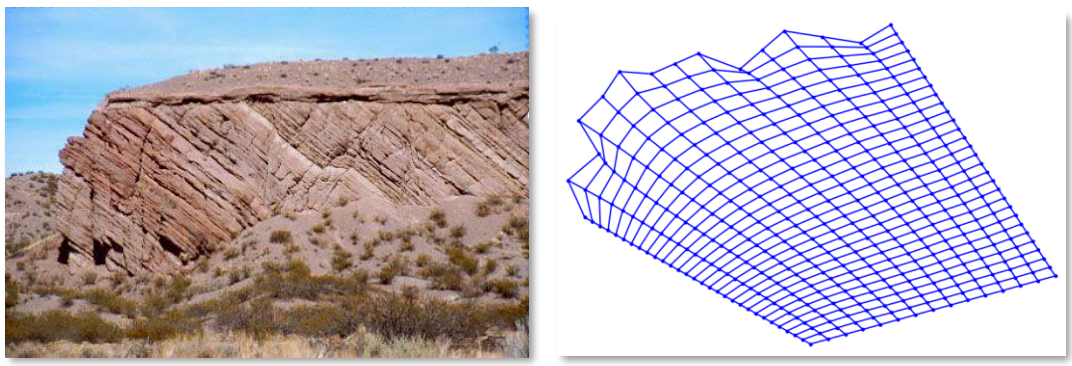


Figura 32. Malla de una discordancia

4.3 Mallas 2D con vista superior de ambientes sedimentarios

A continuación, se muestran algunos ambientes sedimentarios y su representación en malla numérica que pueden ser de interés para un análisis de simulación matemática de yacimientos.

4.3.1 Abanico

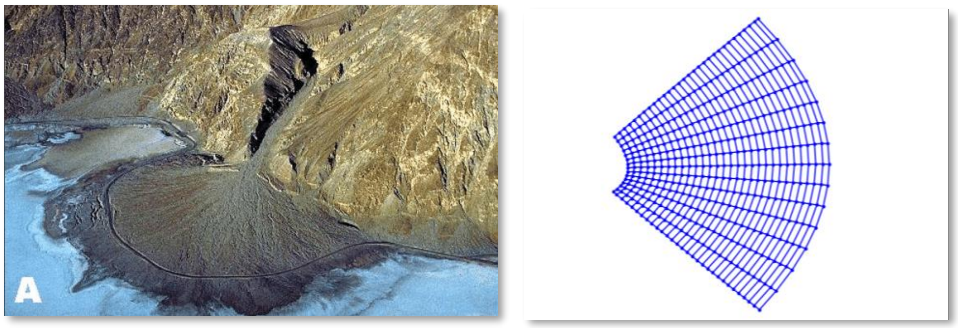


Figura 33. Malla de un abanico

4.3.2 Rio

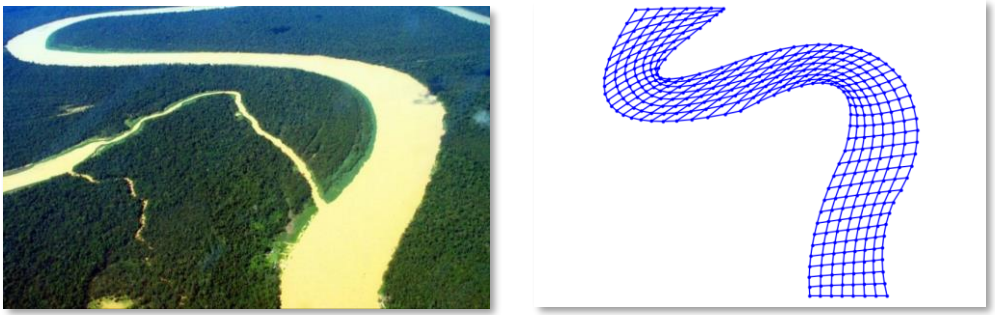


Figura 34. Malla de rio

4.3.3 Lago

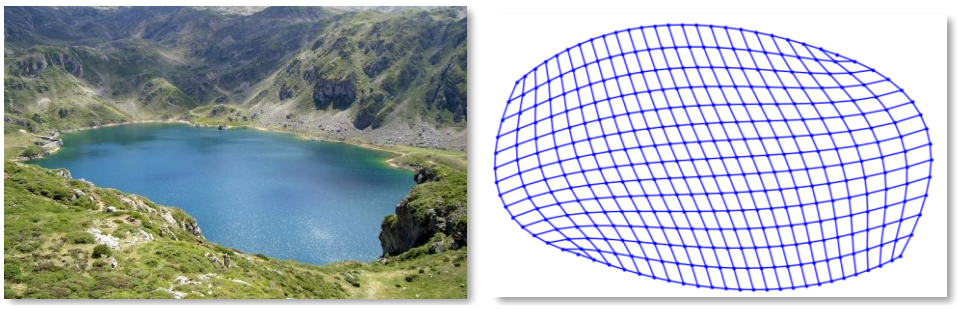


Figura 35. Malla de lago

5. Acople con un simulador de medios porosos

En este capítulo se mencionan las premisas y ecuaciones que ayudan a simular un proceso de inyección de agua en un yacimiento. De igual manera, se retoman los conceptos de MallaPy_2D para la creación de SMYPy_UNAM. Un software con la capacidad de correr un algoritmo de simulación y desplegar los resultados en pantalla.

5.1 Modelo físico conceptual

Como se mencionó en el capítulo 1, en el modelo físico se definen las consideraciones necesarias para abordar un problema de ingeniería, en este caso, la inyección de agua. A continuación, se asumen dichas consideraciones:

- Flujo bifásico
- Medio bidimensional
- Fluidos incompresibles e inmiscibles
- Medio poroso y homogéneo
- Se desprecian los efectos de gravedad
- No hay transferencia de masa entre las fases

Antes de continuar con el modelo matemático es necesario mencionar las ecuaciones fundamentales que gobiernan el flujo de fluidos en medios porosos.

5.1.1 Conservación de la cantidad de movimiento

Se trata de la Ley de Darcy, misma que se puede expresar como:

$$u_\alpha = -\frac{1}{\mu_\alpha} k_\alpha \left(\nabla p_\alpha - \rho_\alpha \delta \nabla \mathrm{z}\right) \tag{38}$$

Donde:

$\alpha = fase \quad w\text{: } agua, \quad o\text{: } aceite$

$u_\alpha = velocidad\ de\ la\ fase$

$k_\alpha = permeabilidad\ efectiva$

$\mu_\alpha = viscosidad$

$p_\alpha = presion$

5.1.2 Ecuación de conservación de masa

$$\frac{\partial(\phi\rho_\alpha S_\alpha)}{\partial t} = -\nabla \cdot (u_\alpha\, \rho_\alpha) + q_\alpha \tag{39}$$

Donde:

$\phi = porosidad$

$\rho_\alpha = densidad\ de\ la\ fase$

$s_\alpha = saturacion\ de\ la\ fase$

$\mu_\alpha = viscosidad$

$q_\alpha = gasto$

5.1.3 Ecuación de continuidad

Considerando que el medio poroso está completamente saturado por las fases agua y aceite:

$$S_w + S_o = 1 \tag{40}$$

Donde

$S_w = Saturación\ de\ agua$

$S_o = Saturación\ de\ aceite$

5.2 Modelo matemático

El modelo matemático promete la obtención de expresiones matemáticas que representen las consideraciones del modelo físico. Entre las ecuaciones que se logran obtener son las de saturación y presión. El siguiente desarrollo matemático fue adaptado de la tesis doctoral: Modelado Numérico de Procesos Térmicos de Recuperación Mejorada de Hidrocarburos (Teja, L., 2018).

Si se reescribe la ecuación de balance de masa de medios porosos en términos de la movilidad de cada fase $\bar{\bar{k}}\lambda_\alpha$, se llega a la siguiente expresión:

$$\frac{\partial \phi \rho_\alpha S_\alpha}{\partial t} - \nabla \cdot \left(\rho_\alpha \bar{\bar{k}} \lambda_\alpha (\nabla p_\alpha - \rho_\alpha \delta \nabla z) \right) = q_\alpha \tag{41}$$

Donde $\lambda_\alpha = \frac{k_{r\alpha}}{\mu_\alpha}$

5.2.1 Ecuación de presión

Para la obtención de la ecuación de presión, se toma la ecuación de conservación de masa (41) y se divide por el termino de densidad de la fase.

$$\frac{1}{\rho_\alpha} \left[\frac{\partial \phi \rho_\alpha S_\alpha}{\partial t} + \nabla \cdot (\rho_\alpha \vec{\mu}_\alpha) - q_\alpha \right] = 0 \tag{42}$$

Sumando sobre todas las fases, se obtiene:

$$\sum_\alpha \left\{ \frac{1}{\rho_\alpha} \left[\frac{\partial \phi \rho_\alpha S_\alpha}{\partial t} + \nabla \cdot (\rho_\alpha \vec{\mu}_\alpha) - q_\alpha \right] \right\} = 0 \tag{43}$$

Al desarrollar las derivadas, se llega a:

$$\sum_\alpha \left\{ \frac{1}{\rho_\alpha} \left[\rho_\alpha S_\alpha \frac{\partial \phi}{\partial t} + \phi S_\alpha \frac{\partial \rho_\alpha}{\partial t} + \rho_\alpha \phi \frac{\partial S_\alpha}{\partial t} + \rho_\alpha \nabla \cdot \vec{\mu}_\alpha + \vec{\mu}_\alpha \nabla \cdot \rho_\alpha - q_\alpha \right] \right\} = 0 \tag{44}$$

Aplicando la ecuación constitutiva $\sum_{\alpha=1,n} S_\alpha = 1$ a la ecuación anterior, se llega a la siguiente expresión:

$$\frac{\partial \phi}{\partial t} + \sum_\alpha \nabla \cdot \vec{\mu}_\alpha + \sum_\alpha \frac{1}{\rho_\alpha} \left[\phi S_\alpha \frac{\partial \rho_\alpha}{\partial t} + \vec{\mu}_\alpha \nabla \cdot \rho_\alpha \right] - \sum_\alpha \frac{q_\alpha}{\rho_\alpha} = 0 \tag{45}$$

Definiendo la velocidad total $\vec{\mu}$ como:

$$\vec{\mu} = \sum_\alpha \vec{\mu}_\alpha \tag{46}$$

Se tiene la siguiente relación:

$$\nabla \cdot \vec{\mu} = \nabla \cdot \sum_{\alpha} \vec{\mu}_{\alpha} = \sum_{\alpha} \nabla \cdot \vec{\mu}_{\alpha} \tag{47}$$

Sustituyendo la ecuación de la velocidad total (47) en (45), se obtiene:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot \vec{\mu}_{\alpha} + \sum_{\alpha} \frac{1}{\rho_{\alpha}} \left[\phi S_{\alpha} \frac{\partial \rho_{\alpha}}{\partial t} + \vec{\mu}_{\alpha} \nabla \cdot \rho_{\alpha} \right] - \sum_{\alpha} \frac{q_{\alpha}}{\rho_{\alpha}} = 0 \tag{48}$$

Aplicando la definición de la velocidad de Darcy para flujo multifásico y la relación de movilidad λ_{α}:

$$\vec{\mu} = \sum_{\alpha} \vec{\mu}_{\alpha} = \sum_{\alpha} \left[-\bar{\bar{k}} \lambda_{\alpha} (\nabla p_{\alpha} - \rho_{\alpha} \delta \nabla \mathrm{z}) \right] \tag{49}$$

Definiendo la relación de flujo fraccional como:

$$f_{\alpha} = \frac{\lambda_{\alpha}}{\lambda} \rightarrow \lambda_{\alpha} = f_{\alpha} \lambda \tag{50}$$

Donde $\lambda = \sum_{\alpha} \lambda_{\alpha}$ es la movilidad total y $\sum_{\alpha} f_{\alpha} = 1$

Sustituyendo la ecuación (50) en (49), se tiene:

$$\vec{\mu} = -\bar{\bar{k}} \lambda \left[\sum_{\alpha} \mathrm{f}_{\alpha} \nabla p_{\alpha} - \sum_{\alpha} f_{\alpha} \rho_{\alpha} \delta \nabla \mathrm{z} \right] \tag{51}$$

Sustituyendo la ecuación de la velocidad total (51) en (48), se obtiene la ecuación de presión en función del flujo fraccional:

$$\frac{\partial \phi}{\partial t} - \nabla \cdot \bar{\bar{k}} \lambda \left[\sum_{\alpha} \mathrm{f}_{\alpha} \nabla p_{\alpha} - \sum_{\alpha} f_{\alpha} \rho_{\alpha} \delta \nabla \mathrm{z} \right] + \sum_{\alpha} \frac{1}{\rho_{\alpha}} \left[\phi S_{\alpha} \frac{\partial \rho_{\alpha}}{\partial t} + \vec{\mu}_{\alpha} \nabla \cdot \rho_{\alpha} \right] - \sum_{\alpha} \frac{q_{\alpha}}{\rho_{\alpha}} = 0 \tag{52}$$

5.2.2 Ecuación de saturación

Partiendo de la ecuación (41) para la fase agua (w) y aceite (o), se tiene lo siguiente:

$$\frac{\partial \phi \rho_w S_w}{\partial t} - \nabla \cdot \left(\rho_w \bar{\bar{k}} \lambda_w (\nabla p_w - \rho_w \delta \nabla \mathrm{z}) \right) = q_w \tag{53}$$

$$\frac{\partial \phi \rho_o S_o}{\partial t} - \nabla \cdot \left(\rho_o \bar{\bar{k}} \lambda_o (\nabla p_o - \rho_o \delta \nabla \mathrm{z}) \right) = q_o \tag{54}$$

Donde (53) es la ecuación de balance de masa para la fase agua y (54) es la ecuación de balance de masa para la fase aceite. Para la ecuación de la fase agua, se tiene:

$$\frac{\partial \phi \rho_w S_w}{\partial t} - \nabla \cdot \left(\rho_w \bar{\bar{k}} \lambda_w (\nabla (p_o - \rho_c) - \rho_w \delta \nabla \mathrm{z}) \right) = q_w \tag{55}$$

Al expandir las derivadas del gradiente de la ecuación se llega a la ecuación de conservación de masa de la fase a gua en función de la presión capilar y la presión de aceite:

$$\frac{\partial (\phi \rho_w S_w)}{\partial t} - \nabla \cdot \left(\rho_w \bar{\bar{k}} \lambda_w \left((\nabla p_o - \nabla \rho_c) - \rho_w \delta \nabla \mathrm{z} \right) \right) = q_w \tag{56}$$

Partiendo de las premisas del modelo físico conceptual, se plantea la siguiente expresión:

$$\nabla \mathrm{p_c} = \frac{dp_c}{dS_w} \nabla S_w \tag{57}$$

Donde, la presión capilar es función de la saturación de la fase mojante.

Sustituyendo en la ecuación (57) en la ecuación (56), llega a la siguiente expresión:

$$\phi \frac{\partial S_w}{\partial t} - \nabla \cdot \left[\bar{\bar{k}} \lambda_w \left(\nabla \mathrm{p_o} - \frac{dp_c}{dS_w} \nabla S_w - \rho_w \delta \nabla \mathrm{z} \right) \right] = \frac{q_w}{\rho_w} \tag{58}$$

Aplicando las condiciones del modelo físico conceptual en la ecuación de conservación de masa de la fase aceite, se tiene:

$$-\phi \frac{\partial S_w}{\partial t} - \nabla \cdot \left[\bar{\bar{k}} \lambda_o (\nabla \mathrm{p_o} - \rho_o \delta \nabla \mathrm{z}) \right] = \frac{q_o}{\rho_w} \tag{59}$$

Al partir de la ecuación de presión general y aplicando las consideraciones del modelo físico conceptual, se tiene:

$$\cancel{\frac{\partial \phi}{\partial t}} - \nabla \cdot \bar{\bar{k}}\lambda \left[\sum_{\alpha} \mathrm{f}_{\alpha} \nabla p_{\alpha} - \sum_{\alpha} f_{\alpha} \rho_{\alpha} \delta \nabla \mathrm{z} \right] + \sum_{\alpha} \frac{1}{\rho_{\alpha}} \cancel{\left[\phi S_{\alpha} \frac{\partial \rho_{\alpha}}{\partial t} + \vec{\mu}_{\alpha} \nabla \cdot \rho_{\alpha} \right]} = \sum_{\alpha} \frac{q_{\alpha}}{\rho_{\alpha}}$$

$$-\nabla \cdot \bar{\bar{k}}\lambda [\mathrm{f_w} \nabla p_w + \mathrm{f_o} \nabla p_o - (f_w \rho_w + f_o \rho_o) \delta \nabla \mathrm{z}] = \frac{q_w}{\rho_w} + \frac{q_o}{\rho_o} \tag{60}$$

Se tiene que: $\mathrm{f_w} + \mathrm{f_o} = 1$ y $\rho_\mathrm{w} = \rho_\mathrm{o} - \rho_\mathrm{c}$, el desarrollo de la ecuación (60) queda de la siguiente manera:

$$-\nabla \cdot \bar{\bar{k}}\lambda [\mathrm{f_w} (\nabla p_o - \nabla p_c) + \mathrm{f_o} \nabla p_o - (f_w \rho_w + f_o \rho_o) \delta \nabla \mathrm{z}] = \frac{q_w}{\rho_w} + \frac{q_o}{\rho_o}$$

$$-\nabla \cdot \bar{\bar{k}}\lambda [\nabla p_o - \mathrm{f_w} \nabla p_c - (f_w \rho_w + f_o \rho_o) \delta \nabla \mathrm{z}] = \frac{q_w}{\rho_w} + \frac{q_o}{\rho_o}$$

$$-\nabla \cdot \bar{\bar{k}}\lambda \left[\nabla p_o - \mathrm{f_w} \frac{dp_c}{\mathrm{d}S_w} \nabla S_w - (f_w \rho_w + f_o \rho_o) \delta \nabla \mathrm{z} \right] = \frac{q_w}{\rho_w} + \frac{q_o}{\rho_o}$$

Donde finalmente, se tiene la ecuación de presión correspondiente al modelo físico conceptual:

$$-\nabla \cdot \left[\bar{\bar{k}}\lambda \nabla p_o - \bar{\bar{k}}\lambda_\mathrm{w} \frac{dp_c}{\mathrm{d}S_w} \nabla S_w - \bar{\bar{k}} (\lambda_w \rho_w + \lambda_o \rho_o) \delta \nabla \mathrm{z} \right] = \frac{q_w}{\rho_w} + \frac{q_o}{\rho_o} \tag{61}$$

Posteriormente, se procede a discretizar la ecuación de saturación mediante diferencias finitas y se aplican condiciones de frontera cerrada.

El proceso para obtener las ecuaciones transformadas al plano computacional se desarrolla a detalle en la tesis doctoral: Modelado Numérico de Procesos Térmicos de Recuperación Mejorada de Hidrocarburos (Teja, L., 2018, pp. 44, 47).

5.3 Interfaz gráfica SMYPy_UNAM

Esta interfaz adapta los conceptos teóricos y prácticos de MallaPy_2D, es decir, se pueden generar mallas no ortogonales como las mostradas en el capítulo 4. Además, se adecuó un código de simulación numérica de yacimientos que trabaja con este tipo de mallas y emplea las ecuaciones de presión y saturación que se explicaron anteriormente.

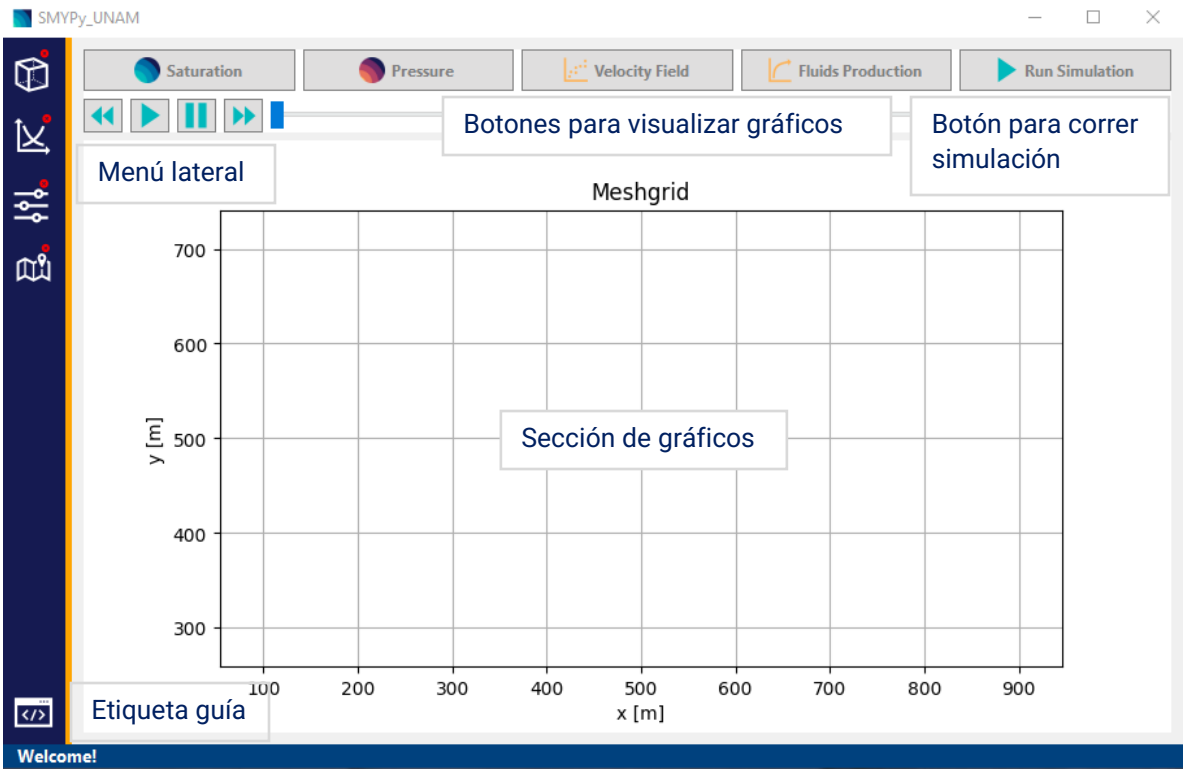


Figura 36. GUI de SMYPy_UNAM

Las opciones del **menú lateral** enlistadas de arriba hacia abajo son:

1. Generación de malla
2. Curvas de permeabilidad relativa
3. Opciones del solver
4. Colocación de pozos productor e inyector

Cada vez que el usuario modifique y guarde los parámetros de simulación, se generará una base de datos con archivos .csv que contiene la información del yacimiento, de igual manera, los botones del menú lateral cambiaran el punto rojo a un punto verde, esto como señal de proceder con la siguiente modificación. Cuando todos los botones laterales tengan el punto verde, es un indicativo de que se puede proceder con la simulación del yacimiento.

5.3.1 Manual de usuario

Generación de la malla

Se realizan los mismos pasos que se explicaron para MallaPy_2D. Es decir, dibujar la malla utilizando el ratón, **Ctrl + X** para pasar a la siguiente frontera y cerrar el contorno. Posteriormente, utilizar el botón de **TFI o PDE** para visualizar la malla y el botón ***Save*** para guardar.

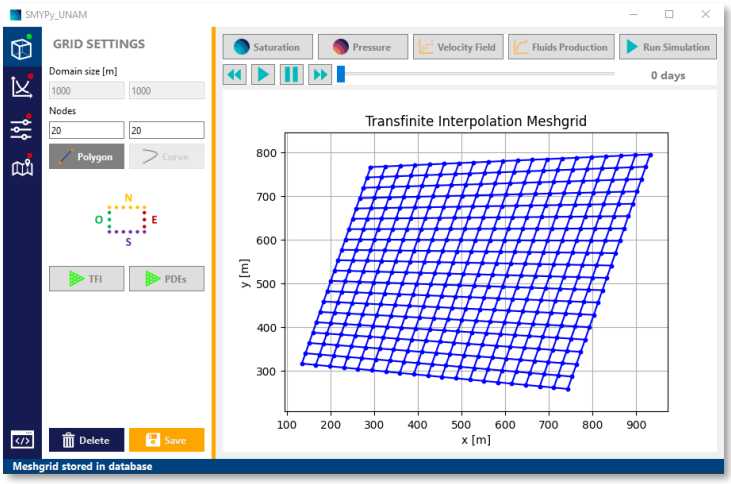


Figura 37. Malla generada con SMYPy_UNAM

Guardar los parámetros de las curvas de permeabilidad relativa

Para calcular las curvas de permeabilidad relativa se utiliza la relación de Brooks-Corey, la cual tiene valores cargados por defecto, pero el usuario puede modificar dichos parámetros y presionar botón ***Save*** para guardar.

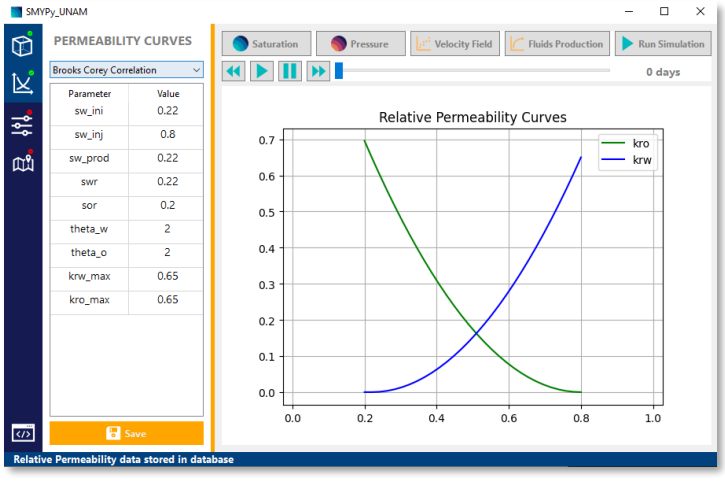


Figura 38. Curvas de permeabilidad relativa

Guardar los parámetros del solver

Modificar los parámetros de la simulación y presionar el botón ***Save*** para continuar.

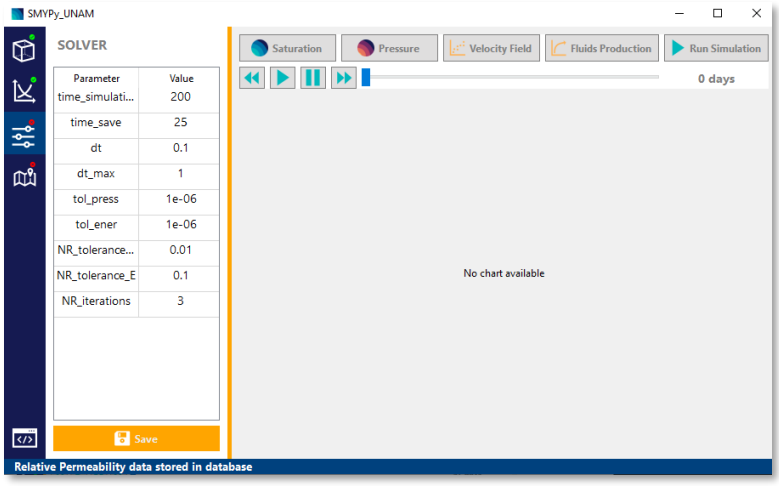


Figura 39. Pantalla del solver de SMYPy_UNAM

Colocación de pozos

Por defecto, el pozo inyector se encuentra en el nodo inicial de la malla, mientras que el pozo productor en el nodo final. No obstante, el usuario puede modificar el posicionamiento de los pozos al interior de la malla.

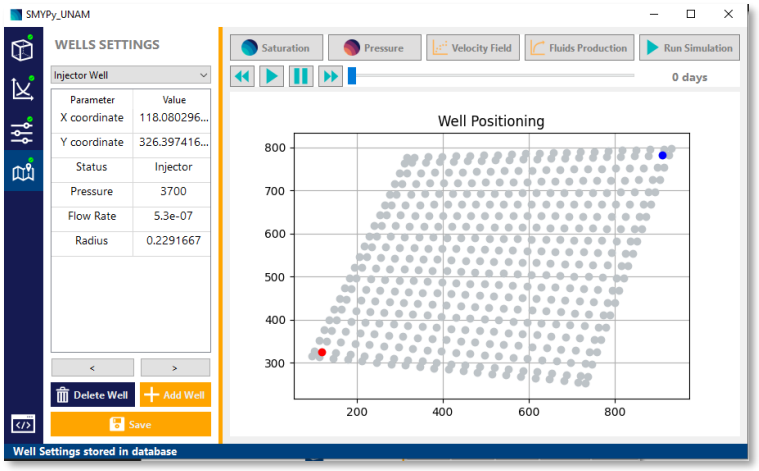


Figura 40. Posicionamiento de los pozos inyector y productor

Correr la simulación

Presionar el botón **Run Simulation,** esperar alrededor de 0.5 a 1 mientras se procesa el código y finalmente, se mostrarán los gráficos de la simulación en pantalla.

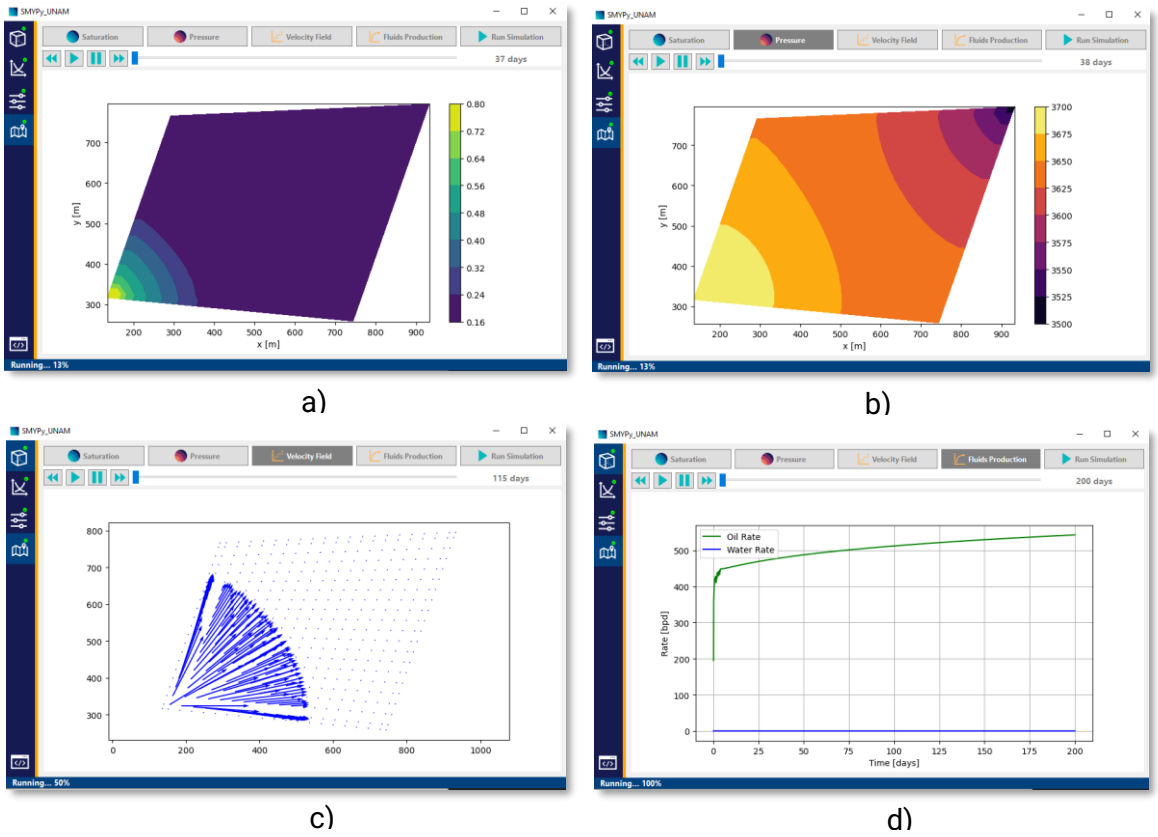


Figura 41. Ejemplo de una corrida de simulación con SMYPy_UNAM

La figura 41-a corresponde al grafico de saturación de agua, 41-b representa la variación de presión dentro del yacimiento. Por otro lado, en la figura 42-c, se ubica el campo de velocidad y finalmente, la figura 41-d, se muestra la gráfica de producción de fluidos, donde el color azul corresponde al agua y el verde al aceite.

Los resultados de la simulación se almacenarán en una carpeta local llamada smy-project. Esta carpeta contendrá los datos a la entrada y a la salida del análisis de simulación.

```
smy-project
|___input-data
|      |___meshgrid-parameters.csv
|      |___permeability-curves.csv
|___ouput-data
      |___simulation_day_25.csv
      |___simulation_day_50.csv
```

5.4 Validación de resultados

Para comprobar el funcionamiento del simulador de yacimientos SMYPy_UNAM, se considera un yacimiento homogéneo con las propiedades descritas en la tabla 3.

Tabla 3. Datos del yacimiento a simular

Propiedad	Valor
Permeabilidad absoluta (kx, ky)	(100, 100) (mD)
Porosidad (ϕ)	0.2
Viscosidad del agua (μ_w)	0.1 (cP)
Viscosidad del aceite (μ_o)	1.14 (cP)
Presión de inyección (p_{inj})	3,700 (psi)
Presión de producción (p_{prod})	3,500 (psi)
Saturación de agua residual (S_{wr})	0.22

5.4.1 Ejemplo 1: simulación con pozos en los extremos de la malla

La malla sobre la cual se realiza la simulación matemática del yacimiento tiene aproximadamente 1000 metros a lo largo y 700 a lo ancho, tiene 50 nodos en ambos ejes (x y) y se compone por curvas en los extremos. Para su realización se aplicó interpolación transfinita y el método iterativo basado en ecuaciones diferenciales.

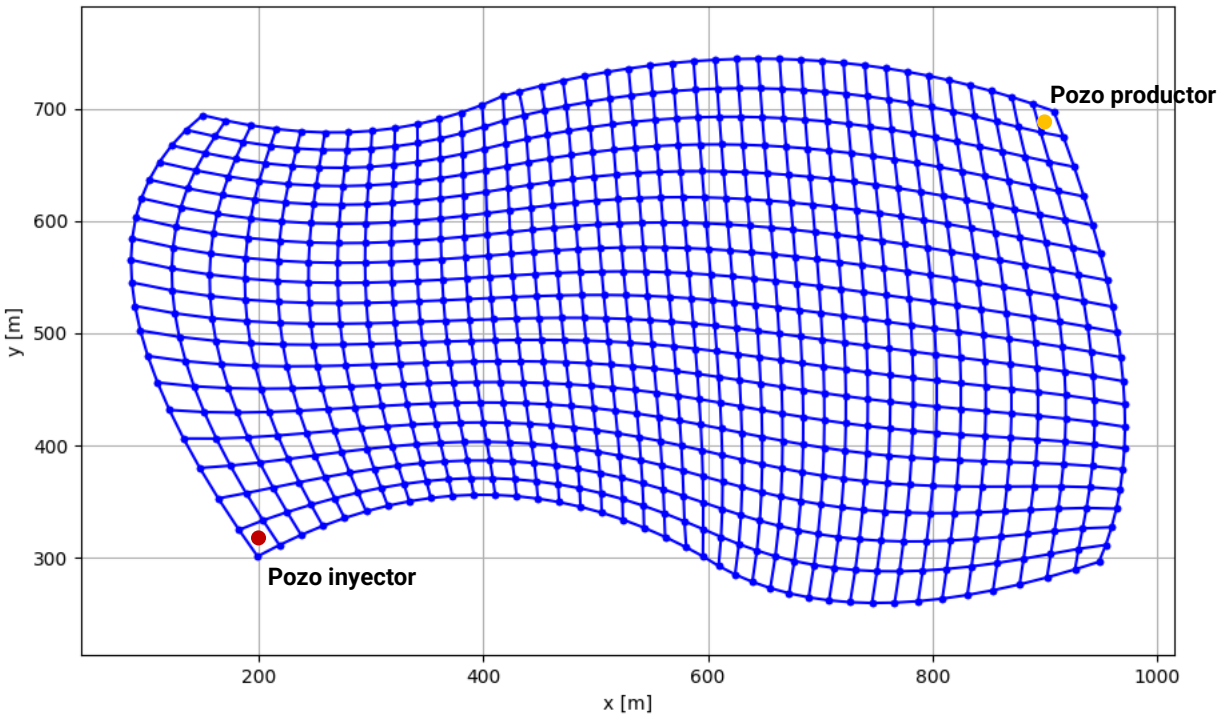


Figura 42. Posicionamiento de pozos inyector y productor del ejemplo 1

Al ejecutar el algoritmo de simulación matemática se obtienen los siguientes gráficos de saturación de agua desde el día 100 al día 400.

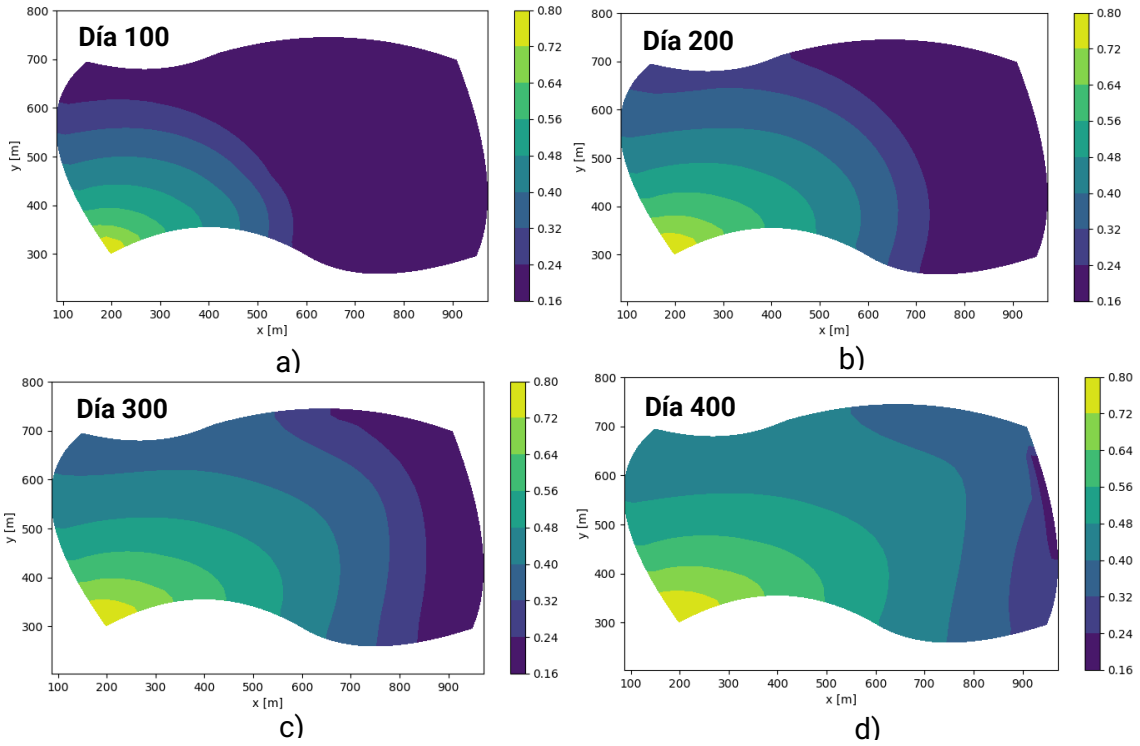


Figura 43. Corte de agua a distintos tiempos de simulación del ejemplo 1

En la figura 43-a y 43-b se logra apreciar el avance que tiene el agua de inyección dentro del yacimiento. Por otro lado, en la figura 43-c y 43-d, el agua de inyección empieza a ser producida por el pozo productor, en consecuencia, la producción de aceite disminuye gradualmente.

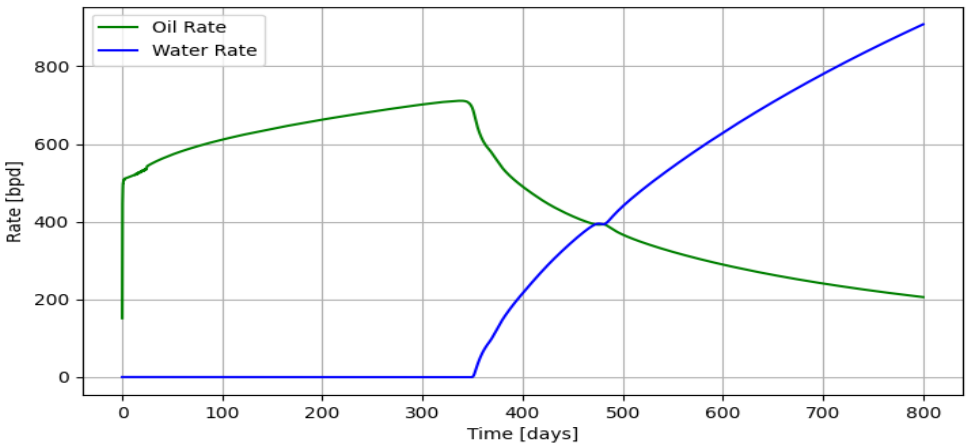
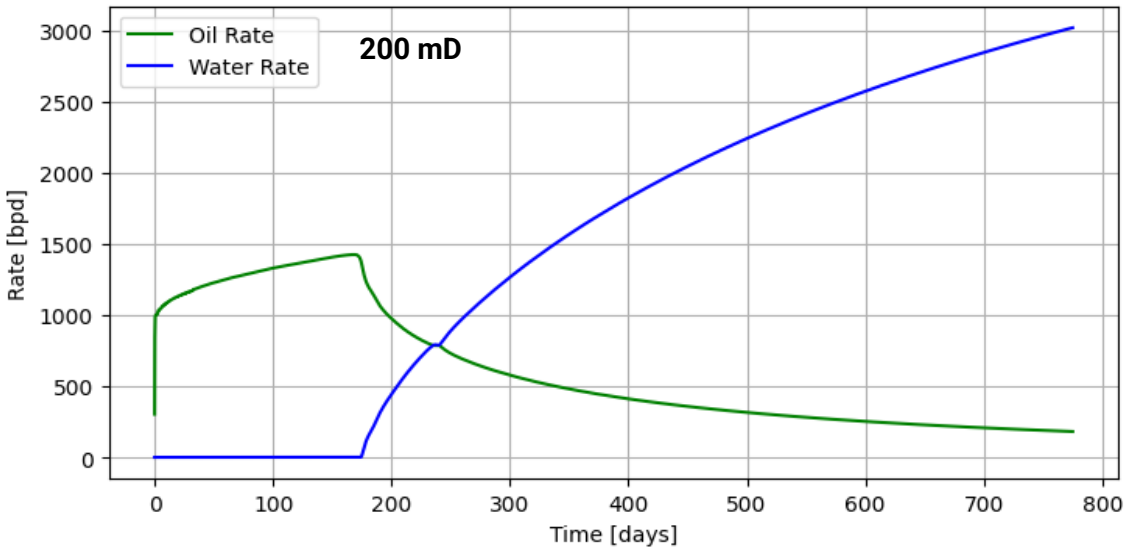
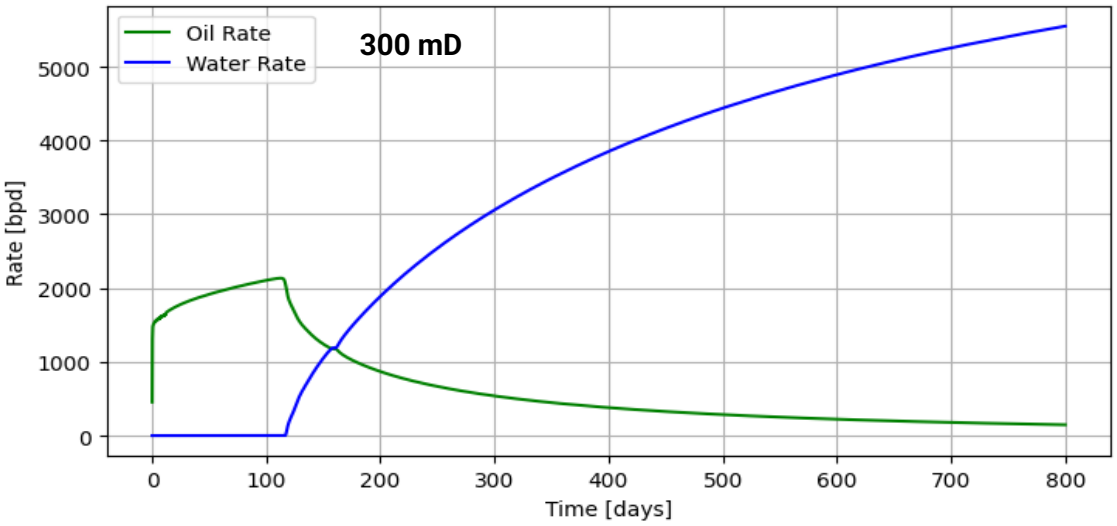


Figura 44. Gráfico de producción de fluidos

Al variar la permeabilidad absoluta en ambos ejes desde 100 mD a 200 mD y 300 mD se puede apreciar que el corte de agua ocurre en los días 180 y 110, respectivamente. Por otro lado, el gasto de aceite supera los 1,000 bpd en ambos casos. En otras palabras, un yacimiento con permeabilidad mayor a los 100 mD iniciales permite que la producción de aceite se realice en menos tiempo.



a)



b)

Figura 45. Producción de fluidos para el ejemplo 1 al variar la permeabilidad absoluta

5.4.2 Ejemplo 2: simulación con pozos al interior de la malla

Considerando una malla en forma de anticlinal con 40 nodos en el eje x y 10 nodos en el eje y, se obtiene la siguiente figura:

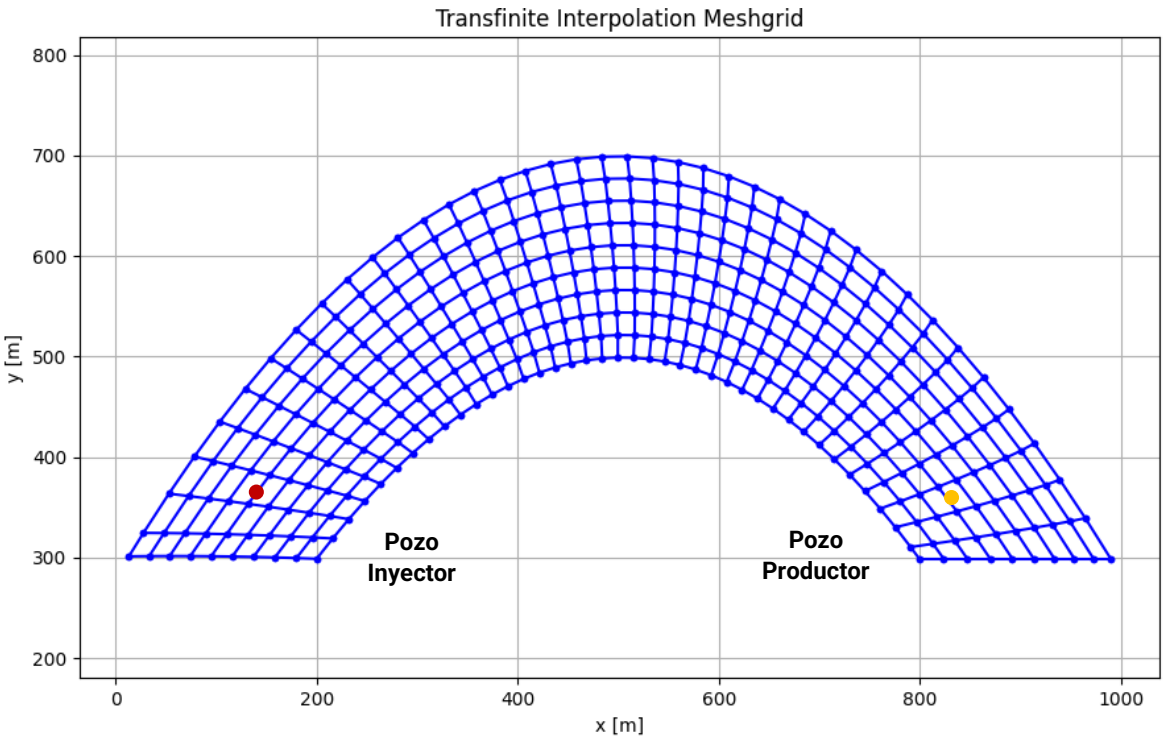


Figura 46. Malla con forma de anticlinal para ejemplo 2

Para este ejemplo, los pozos inyector y productor se encuentran en los nodos interiores de la malla, en vez de los exteriores, como en el ejemplo 1. Los gráficos de saturación de agua para los días 25, 100, 200 y 300 se muestran a continuación.

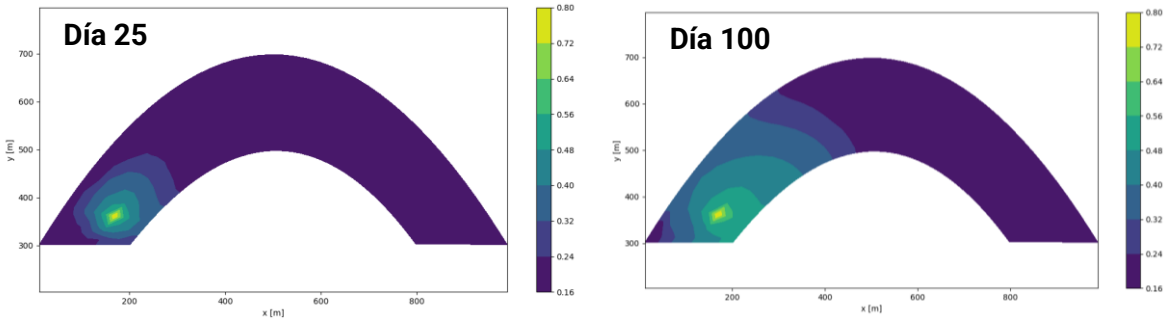
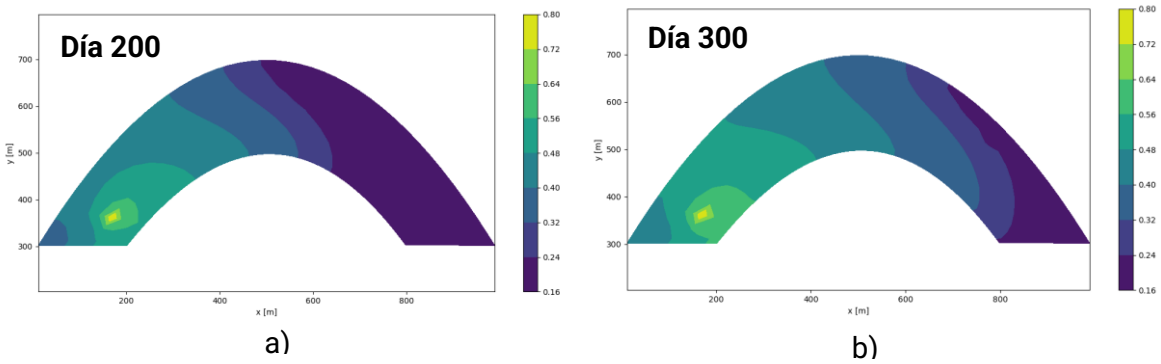


Figura 47. Gráficos de saturación de agua para días 25 y 100



a) b)

Figura 48. Gráficos de saturación de agua para días 200 y 300

Como se puede apreciar en los gráficos de saturación, el avance del agua de inyección tiene un comportamiento similar al ejemplo 1. Por otro lado, en los gráficos de producción de fluidos, a partir del día 300, el agua de inyección aumenta en el pozo productor y el aceite disminuye gradualmente.

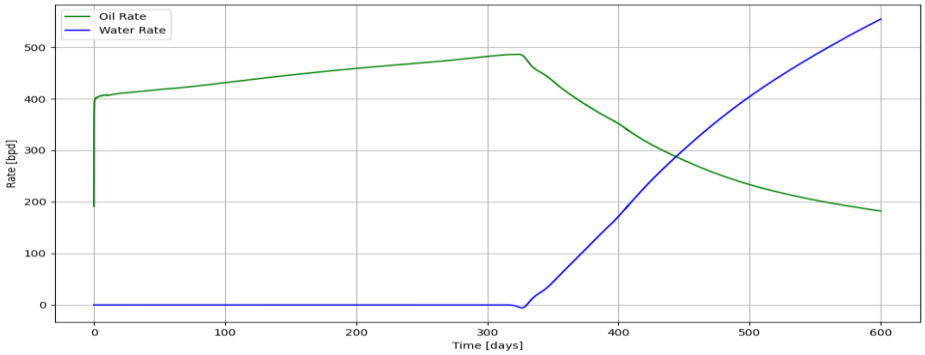


Figura 49. Producción de fluidos para el ejemplo 2

Al modificar los parámetros de las curvas de permeabilidad relativa, en particular θ_w y θ_o, de 2 a 3, se obtienen los siguientes gráficos de saturación:

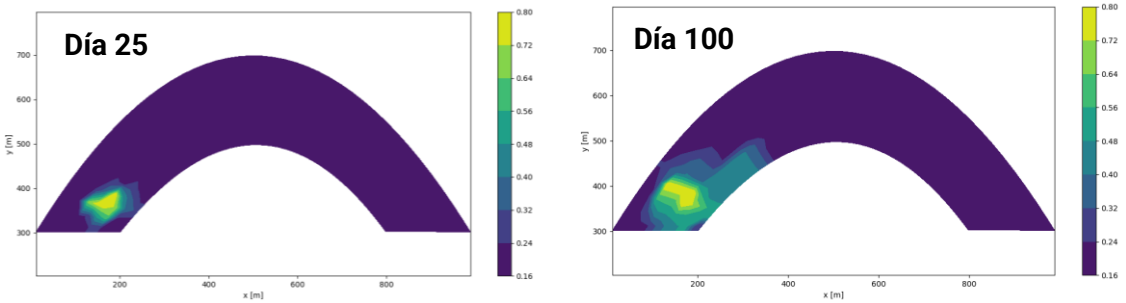


Figura 50. Saturación de agua al modificar los parámetros theta de la correlación Brooks-Corey

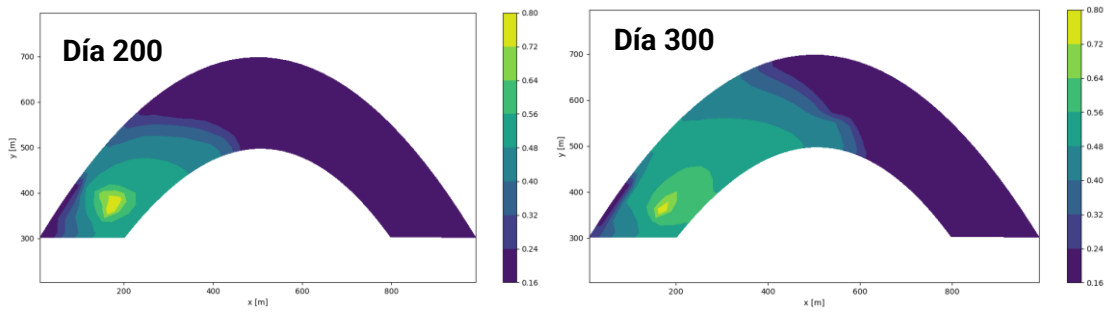


Figura 51. Saturación de agua para los días 200 y 300

Como consecuencia de modificar θ_w y θ_o, el avance del agua de inyección dentro del medio poroso es menos uniforme y esto se puede corroborar con la figura 52, donde se muestra la producción de fluidos.

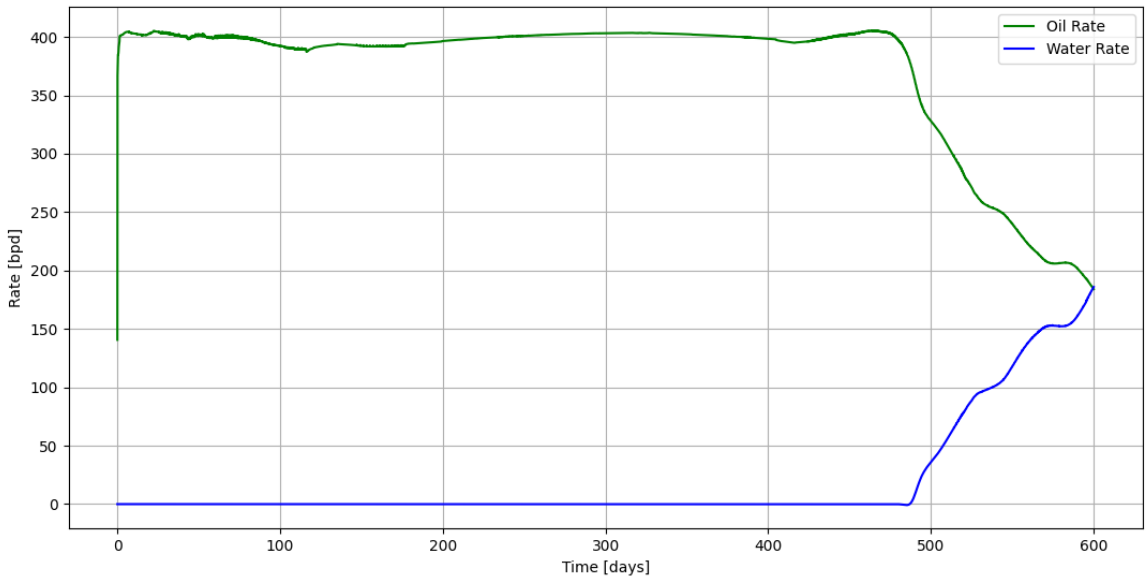


Figura 52. Producción de fluidos al modificar las curvas de permeabilidad relativa

Otra característica es que el agua de inyección se empieza a producir entre los 490 y 500 días de simulación, mismo que contrasta con los 325 días de la simulación anterior.

5.4.3 Ejemplo 3: simulación sobre una formación heterogénea

Una formación se considera heterogénea cuando las propiedades de la roca cambian con la ubicación del yacimiento. En este ejemplo se introduce una función que genera valores aleatorios de permeabilidad absoluta en un rango de 20 a 180 mD, en ambos ejes y la porosidad de 0.05 a 0.35. La malla sobre la cual se ejecutará la simulación se muestra a continuación.

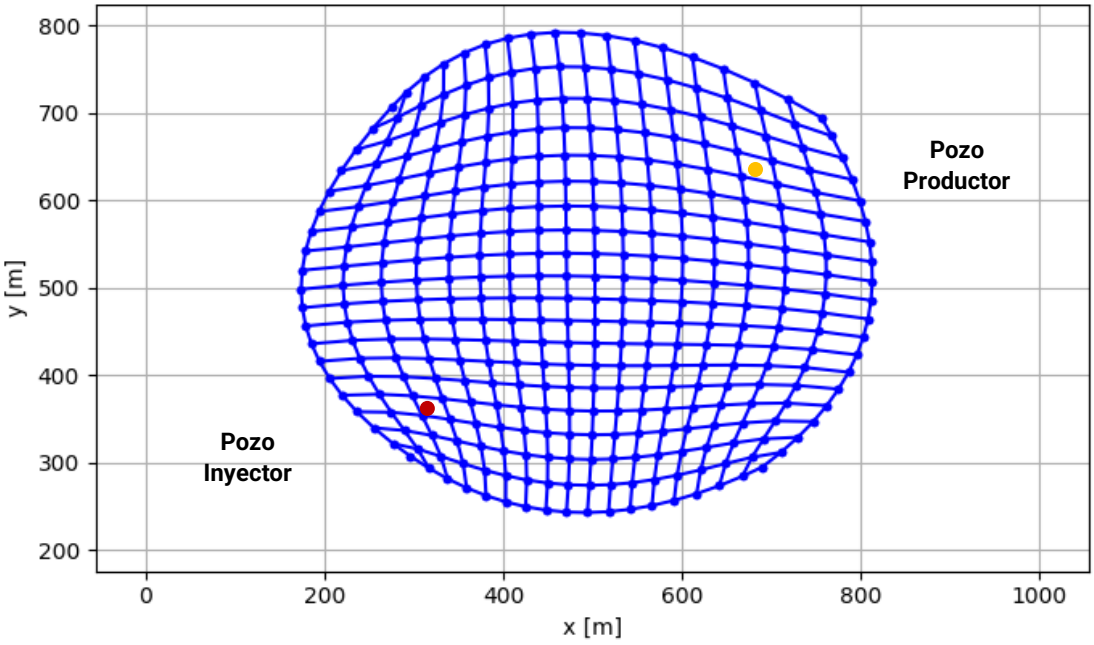


Figura 53. Malla geométrica para ejemplo 3

En los siguientes gráficos se muestra el resultado de aplicar una función que genera números aleatorios dada una distribución normal. Como consecuencia se tiene una mejor representación del yacimiento.

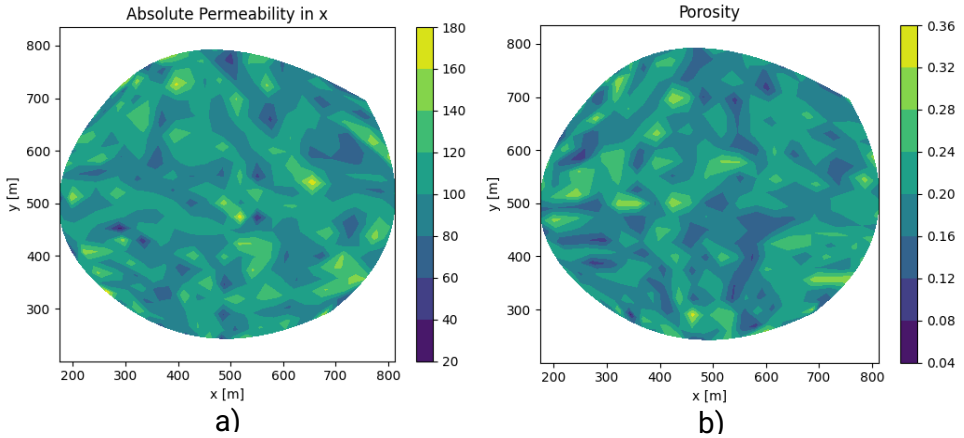


Figura 54. Permeabilidad absoluta (a) y porosidad (b) de formación heterogénea

Los gráficos de saturación de agua denotan un frente de inyección con ligeros picos y variaciones. Esta característica se puede atribuir a que las propiedades del yacimiento, en este caso, permeabilidad y porosidad, varían con respecto a la posición.

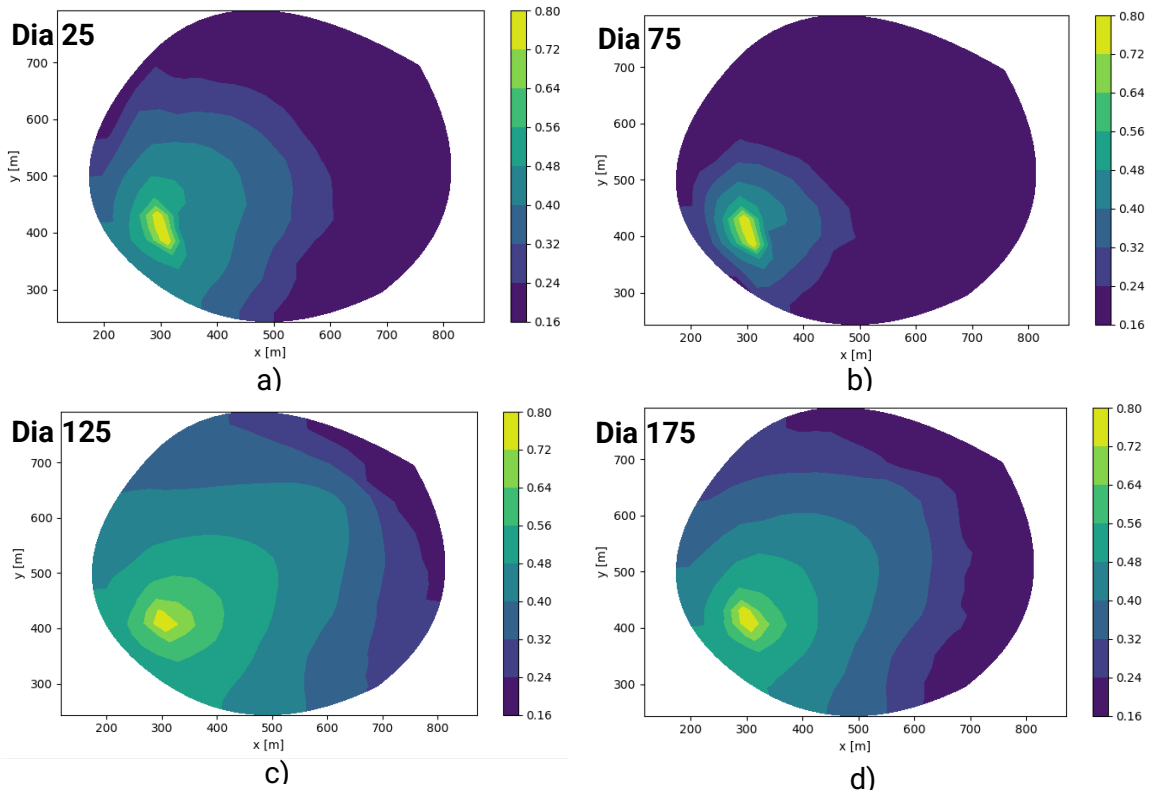


Figura 55. Gráficos de saturación de agua del ejemplo 3

El agua de inyección se produce en un tiempo relativamente menor, comparado con los ejemplos anteriores. Esto puede ocurrir debido a la geometría de la malla y zonas donde la permeabilidad o porosidad es mayor al promedio.

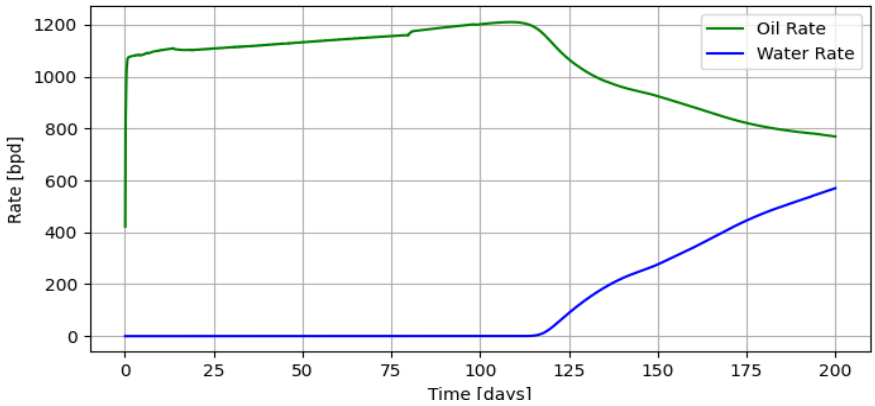


Figura 56. Gráfico de producción de fluidos del ejemplo 3

6. Conclusiones

La realización de las dos interfaces graficas presentadas en este material didáctico representa un complemento para aquellos códigos y algoritmos de simulación matemática de yacimientos (SMY) que se explican en clase, pero no se llegan a desarrollar en una aplicación o software portable.

Por otro lado, una de las ventajas de la aplicación SMYPy_UNAM es que permite realizar un análisis de simulación matemática de yacimientos con mayor facilidad en comparación con los editores de código. Además, se agrega la implementación de mallas no ortogonales con los fundamentos del software MallaPy_2D, lo cual puede significar una mejor representación del yacimiento. Por otro lado, el usuario puede interactuar de forma sencilla y sin muchas complicaciones con los datos a la entrada y a la salida del simulador. Cabe destacar que los softwares desarrollados y presentados en este material didáctico se registraron ante INDAUTOR y sus números de registro son: 03-2022-051314150500-01 para MallaPy_2D y 03-2022-051314125000-01 para SMYPy_UNAM.

Finalmente, los resultados a la salida indican un buen comportamiento de las ecuaciones que gobiernan el flujo de fluidos en el medio poroso, no obstante, se debe continuar con la optimización de los algoritmos de SMY y su representación en código. Esto con la finalidad de que el resultado de la simulación matemática del yacimiento sea lo más aproximado posible al problema real.

7. Recomendaciones y trabajo a futuro

7.1 Recomendaciones para MallaPy_2D

Trabajar con geometrías irregulares sencillas. De tal manera que, si el usuario comete algún error, no se tenga que empezar desde cero el contorno de la malla.

Esperar la primera ejecución del programa por más de medio minuto, esto debido a la optimización del código en las subsecuentes corridas.

7.2 Recomendaciones para SMYPy_UNAM

Verificar que las mallas resultantes no tengan cruces entre líneas o la distancia entre éstas sea mínima, ya que puede afectar los cálculos de diferencias y generar un error al momento de la simulación.

Asegurarse de guardar los datos modificados antes de correr el análisis de simulación.

No se recomienda realizar una nueva corrida de simulación cuando se está ejecutando un proceso.

7.3 Trabajo a futuro para MallaPy_2D

Generar una nueva versión donde se tenga un mayor control sobre las líneas de atracción de la malla.

Introducir una solución para generar mallas estructuradas no ortogonales en 3 dimensiones.

7.4 Trabajo a futuro para SMYPy_UNAM

Disminuir el tiempo de cómputo que tarda el programa en resolver las ecuaciones diferenciales, de tal manera que los cálculos asimilen la velocidad de un leguaje tipado como C o C++.

Mejorar la interfaz gráfica de acuerdo con las necesidades del ingeniero de yacimientos o el encargado de gestionar el análisis de simulación numérica del yacimiento, esto para facilitar el flujo de trabajo.

Implementar el problema en 3 dimensiones. De tal forma que los gráficos resultantes ejemplifiquen de mejor forma lo que ocurre en el yacimiento.

Prevención de bugs y fallos al momento de introducir datos erróneos o mallas no soportadas para análisis de SMY.

8. Referencias

AAPG_Wiki. (20 de Enero de 2022). *Permeability*. Obtenido de https://wiki.aapg.org/Permeability#cite_note-Petersetal_2012-1

AAPG_Wiki. (20 de Enero de 2022). *Porosity*. Obtenido de https://wiki.aapg.org/Porosity

Chen, Z., Hua, G., & Ma, Y. (2006). *Computational Methods for Multiphase Flows in Porous Media.* Dallas,Texas: Society for Industrial and Applied Mathematics.

Colchado, A. (20 de Enero de 2022). *Capítulo 5 - DINÁMICA DE FLUIDOS COMPUTACIONAL*. Obtenido de Google_Sites: https://sites.google.com/site/manifo1dmx/capitulo-5

Hernández, M. A., & Dominguez, G. C. (1984). *Apuntes de simulación matemática de yacimientos.* Ciudad de México: Facultad de Ingeniería UNAM.

PetroWiki. (15 de Enero de 2018). *Reservoir simulation*. Obtenido de https://petrowiki.spe.org/Reservoir_simulation

Ruiz, A., & López, A. (s.f.). *CONACYT*. Obtenido de CONACYT: https://www.conacyt.gov.py/sites/default/files/PVCT16-135_Transferencia.pdf

Schlumberger. (5 de Enero de 2022). Obtenido de Schlumberger: https://glossary.oilfield.slb.com/es/terms/r/relative_permeability

Schlumberger. (2022). *Effective Permeability*. Obtenido de Oilfield Glossary: https://glossary.oilfield.slb.com/en/terms/p/permeability#:~:text=Effective%20permeability%20is%20the%20ability,a%20gas%2Dwater%20reservoir).

Schlumberger. (2022). *Relative Permeability*. Obtenido de Oilfield Glossary: https://glossary.oilfield.slb.com/en/terms/p/permeability#:~:text=Relative%20permeability%20is%20the%20ratio,its%20relative%20permeability%20is%201.0.

Selmin, V. (19 de Enero de 2022). *SlideToDoc*. Obtenido de https://slidetodoc.com/mesh-generation-concept-structured-grids-v-selmin-multidisciplinary/

Teja, L. (2011). *Generación de Mallas Numéricas para Geometrías Irregulares y Complejas.* México: Centro Nacional de Investigación y Desarrollo Tecnológico.

Teja, L. (2018). *Modelado Numérico de Procesos Térmicos de Recuperación Mejorada de Hidrocarburos.* Ciudad De México: Instituto de Geofísica.

Thomson, J. F., Soni, B., & Weatherill, N. (1999). *Handbook of grid generation* . CRC Press.