TRABAJO ESPECIAL DE GRADO

IMPLEMENTACIÓN DE UN ALGORITMO PARA LA SIMULACIÓN ESTOCÁSTICA DE MODELADO DE FACIES, BASADO EN EL MÉTODO DE GEOESTADÍSTICA MULTIPUNTO.

Presentado ante la Ilustre

Universidad Central de Venezuela

Por el Br. Corrales R., Matías

Para optar al Título

de Ingeniero Geofísico

Caracas, 2014

TRABAJO ESPECIAL DE GRADO

IMPLEMENTACIÓN DE UN ALGORITMO PARA LA SIMULACIÓN ESTOCÁSTICA DE MODELADO DE FACIES, BASADO EN EL MÉTODO DE GEOESTADÍSTICA MULTIPUNTO.

TUTOR ACADÉMICO: Prof. Ricardo Ambrosio.

Presentado ante la Ilustre

Universidad Central de Venezuela

Por el Br. Corrales R., Matías

Para optar al Título

de Ingeniero Geofísico

Caracas, 2014

Caracas, Febrero 2014

Los abajo firmantes, miembros del Jurado designado por el Consejo de la Escuela de Ingeniería Geológica, Minas y Geofísica, para evaluar el Trabajo Especial degrado presentado por el Bachiller Matías Corrales R. titulado:

"Implementación de un algoritmo para la simulación estocástica de modelado de facies, basado en el método de geoestadística multipunto"

Consideran que el mismo cumple con los requisitos exigidos por el plan de estudios conducente el Título de Ingeniero Geofísico, y sin que ello signifique que se hacen solidarios con las ideas expuestas por el autor, lo declaran APROBADO.

Prof. Miguel Bosch
Jurado

Prof. Ronny Meza
Jurado

Prof. Ricardo Ambrosio
Tutor Académico

Agradecimientos

A la Universidad Central de Venezuela por contribuir junto con su cuerpo docente a la formación personal y profesional.

A mis padres Mabel Romano y Luis Corrales por haberme apoyado durante toda la carrera.

A Ricardo Ambrosio por haberme ofrecido este trabajo al igual que su ayuda y tiempo.

A todos ellos, muchas gracias.

Corrales R., Matías

IMPLEMENTACIÓN DE UN ALGORITMO PARA LA SIMULACIÓN ESTOCÁSTICA DE MODELADO DE FACIES, BASADO EN EL MÉTODO DE GEOESTADÍSTICA MULTIPUNTO.

Tutor Académico: Prof. Ricardo Ambrosio. Trabajo Especial de Grado. Caracas, U.C.V. Facultad de Ingeniería. Escuela de Geología, Minas y Geofísica. Año 2014, 100 pág.

Palabras clave: Geoestadística, Simulación, Multipunto, Algoritmo.

Resumen

En los últimos años se han utilizado diferentes métodos para predecir los valores en localizaciones no observadas de variables descriptivas del subsuelo, tales como propiedades petrofísicas e indicadores de facies. Entre estos métodos encontramos algoritmos de interpolación, superficies de ajuste y métodos de simulación. Específicamente para el modelado de facies se han utilizado principalmente el kriging de variables indicadoras y el modelado por objetos. Estos métodos, a pesar de reproducir correctamente ciertos tipos de escenarios, modelando la correlación espacial entre pares de puntos o las formas de los objetos geológicos, se han encontrado con limitaciones que en la actualidad se pretende superar mediante la aplicación de la geoestadística multipunto la cual permite crear modelos uniendo diferentes fuentes de información, como son datos de facies observadas, por ejemplo de pozos o afloramientos, e información subjetiva que describe el modelo geológico conceptual. Este trabajo se propone implementar y evaluar el desempeño de un algoritmo de simulación multipunto para dos facies en dos dimensiones con la intención de desarrollar y divulgar una mejor comprensión del método sobre el cual hay poca documentación disponible por estar aún en fases de investigación y mejora tanto en la academia como en la industria.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPITULO I

INTRODUCCIÓN

Planteamiento del Problema

Los algoritmos de simulación estocástica de facies han sido utilizados en las últimas décadas con el propósito de cuantificar la incertidumbre asociada a la estimación de la variable de interés en puntos del dominio en estudio donde se desconoce su valor (Obregon & Fragala, 2002). Estos autores, refieren que si bien se han hecho importantes avances con respecto a las técnicas de estimación de los indicadores de facies en cuanto a la representación de las características del yacimiento y cuantificación de la incertidumbre asociada a ella, tales técnicas de simulación espacial son deficientes en la reproducción de aspectos reales del sistema a modelar como por ejemplo, la conectividad de valores extremos y rasgos curvilíneos.

Strebelle (2002) muestra que la correcta reproducción de canales fluviales en modelados de yacimientos tiene un efecto significativo para la predicción del modelo. Tal necesidad es particularmente evidente en el modelado geoestadístico de yacimientos, en el cual, más que la estimación de la variable, importa su conectividad, es decir, el patrón espacial descrito por las facies presentes en la zona de estudio. Esta limitación se debe al hecho de que las herramientas que generalmente se utilizan, están basadas en un atributo estadístico, el variograma, que considera solo estadística entre dos puntos (Obregon & Fragala, 2002).

Aun cuando esta técnica es ampliamente usada para la caracterización de dichos patrones se necesita más que una correlación de dos puntos como lo hace el variograma tradicional, ya que este por sí solo no permite el modelado de los rasgos propios de los yacimientos. Con el propósito de realizar con mayor certeza una simulación estratigráfica, en los últimos años se ha venido desarrollando y afinando, especialmente en sectores científicos relacionados con la industria petrolera, la técnica geoestadística multipunto. Ésta técnica permite capturar y cuantificar patrones espaciales a partir de una o varias “imágenes de entrenamiento”, la cual se obtiene mediante el análisis geológico previo de la zona de estudio, y posteriormente se diseña un bosquejo de la configuración espacial estimada del yacimiento, permitiendo así al modelador considerar los patrones más importantes de estas imágenes y reproducirlos a través de una simulación estocástica condicionada a los datos de pozos y datos de prospección sísmica (Caers & Srinivansan, 2001). Dicho método pudiese emplearse para mejorar las estimaciones que se realizan actualmente en la industria, disminuyendo así la incertidumbre asociada al modelado por kriging en yacimientos.

En vista a lo referido anteriormente, este trabajo se propone realizar mediante la simulación multipunto, la implementación de un algoritmo capaz de modelar diferentes configuraciones bidimensionales de facies, que coincidan con datos sintéticos generados, generando escenarios en donde se intente obtener situaciones del mundo real.

Objetivos

Objetivo General

Validar un algoritmo de simulación de patrones de formaciones geológicas, que cumplan con condiciones introducidas por el usuario a través de una imagen de entrenamiento condicionada en datos sintéticos de facies observada.

Objetivos Específicos

a) Proponer imagen de entrenamiento y configuración de datos de entrada.

b) Extraer patrones de imagen de entrenamiento.

c) Condicionar patrones generados con datos sintéticos de facies observada.

d) Simular modelos de facies basados en los escenarios definidos por la imagen de entrenamiento y características de los datos de entrada sintéticos.

e) Proponer criterios de evaluación de escenarios generados con la imagen de entrenamiento.

f) Evaluar resultados obtenidos y comparar diferentes escenarios con la imagen de entrenamiento.

Justificación

El siguiente trabajo parte de la necesidad de implementar un algoritmo capaz de modelar de manera verosímil, una configuración de facies bidimensionalmente distribuida, en la que otros métodos de interpolación no son capaces de representar.

El propósito de dicho modelo, es el de disminuir la incertidumbre para la posterior toma de decisiones en la propuesta de posibles pozos productores de hidrocarburo, implicando un gran avance en el método que se utiliza actualmente en la industria, ya que mejora la calidad de la información, y así aprovechar más eficientemente los recursos disponibles.

Así mismo, se producirá un aporte a la comunidad geocientífica, con fines académicos y colaborará con el desarrollo de una nueva línea de investigación de esta importante herramienta, la cual no está extensamente encontrada en la literatura.

Alcances

Este trabajo tiene como meta generar modelos que representen dos facies, bidimensionalmente distribuidas, mediante el cual se tendrá como condición que la imagen de entrenamiento esté representada con arreglos de solo dos colores, indicando cada color una respectiva facies a modelar.

CAPÍTULO II

MARCO TEÓRICO

El propósito fundamental al aplicar la geoestadística es el de predecir los valores de la variable en los sitios no muestreados y posteriormente simular estas variables regionalizadas e identificar su comportamiento espacial.

La etapa de poblamiento o predicción geoestadística se divide en dos actividades: la primera es la de poblar con indicadores facies el área a estudiar y la segunda es la de poblar las celdas del modelo asociadas a estas facies con sus respectivas propiedades petrofísicas como son la porosidad, permeabilidad, entre otras. Al hacer esto logramos tener valores con mayor certidumbre y por ende permitirá tomar decisiones más certeras a la hora de estimar la geometría y comportamiento de un yacimiento. Es necesario realizar estas actividades en el orden mencionado, ya que al poblar las facies con las propiedades petrofísicas, estas son condicionales en el tipo de indicador de facies anteriormente simulado.

En este trabajo nos centraremos en la primera actividad, el poblamiento de facies, mediante la simulación de estas, pero ¿Qué es la simulación? Y ¿Por qué es importante?, la simulación es un intento de modelar las situaciones del mundo real por medio de un programa computacional, el cual tiene como finalidad estudiar dichas situaciones y con ello lograr entender el comportamiento y la variabilidad de los efectos aleatorios del sistema. Tradicionalmente, el modelado formal de sistemas ha sido a través de un modelo matemático, que intenta encontrar soluciones analíticas a problemas que permiten la predicción del comportamiento de un sistema de un conjunto de parámetros y condiciones iniciales.

Método de Montecarlo.

El método de Montecarlo es un método numérico que permite resolver problemas matemáticos mediante la simulación de variables aleatorias (Sobol, 1983).

La base teórica de este método es conocida hace mucho tiempo, es más algunos problemas de la estadística se resolvían a veces empleando las muestras aleatorias, o sea, aplicando de hecho el método de Montecarlo. Sin embargo, hasta la aparición de las maquinas calculadoras eléctricas (MCE), este método no encontraba aplicaciones suficientemente amplias ya que la simulación a mano de variables aleatorias constituye un proceso muy laborioso. En otras palabras, la aparición del método de Montecarlo se hizo posible solo gracias a la creación de MCE.

Para lograr tener una idea más clara de la base de este método, consideremos el siguiente ejemplo. Supongamos que debemos calcular el área de la figura plana S (Figura 1), y supongamos que toda la figura está comprendida dentro de un cuadrado de dimensión uno.

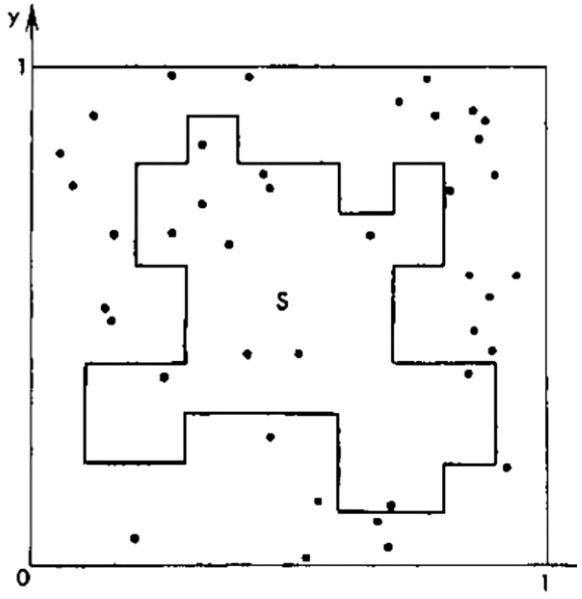


Figura 1. Ejemplo ilustrativo método de Montecarlo

Tomemos en el cuadrado un numero N de puntos aleatorios, sea N' el número de puntos que aparecen dentro de S. Es obvio por razones geométricas, que el área de S es igual aproximadamente a la razón N'/N. cuanto mayor sea N mayor será la exactitud de esta estimación.

En este ejemplo se ha escogido un total de N = 40 puntos. De estos, N' = 12 puntos aparecen dentro de S. Tenemos entonces N'/N = 12/40 = 0.30, mientras que el valor exacto del área de S es 0.35 (Sobol, 1983).

Existen dos peculiaridades de este método, la primera consiste en que su algoritmo tiene una estructura muy sencilla, como regla, se elabora primero un programa para la realización de una prueba aleatoria (en el ejemplo anterior se debe escoger un punto aleatorio y comparar en cada caso si este pertenece o no a S), después, esta prueba se repite N veces de modo que cada experimento sea independiente de los restantes y se toma la media de los resultados de todos los experimentos. Por esto el método de Montecarlo se denomina un método de pruebas estadísticas.

La segunda peculiaridad consiste en que el error es, como regla, proporcional a la magnitud $\sqrt{D/N}$, donde *D* es una constante y *N* es el número de pruebas. Ésta fórmula permite ver que para disminuir el error en 10 veces (en otras palabras, para obtener en el resultado otra cifra decimal exacta) es preciso aumentar N unas 100 veces.

Se pone en manifiesto la imposibilidad de alcanzar en este camino una elevada exactitud. Por eso suele decirse que para el método de Montecarlo resulta, especialmente eficaz, en la solución de problemas en los cuales se necesita conocer el resultado con poca exactitud (del 5 al 10 por ciento). Sin embargo, un mismo problema puede ser resuelto aplicando distintas variables del método de Montecarlo, en numerosos problemas se logra

elevar considerablemente la exactitud escogiendo un procedimiento de cálculo al que se le corresponde un valor mucho menor de D.

Al hablar del tipo de problemas que logra resolver este método, básicamente permite simular cualquier proceso cuya marcha depende de factores aleatorios. (Sobol, 1983).

Motivación para el modelado de yacimientos

Hasta principios de los años ochenta, la mayoría de los modelos de yacimientos eran simplemente modelos de capas. En estos modelos, capas homogéneas o bloques de fallas contenían permeabilidades constantes lo cual no reflejaba a pequeña escala el verdadero comportamiento de la trayectoria de un fluido (alta permeabilidad), incluso las posibles barreras que esta puede encontrar (baja permeabilidad), con este tipo de modelo de capas.

Posteriormente, métodos de estimación llenaban estas capas, variando continuamente las propiedades petrofísicas mediante la interpolación de valores encontrados en los pozos. Uno de estos métodos de interpolación fue el kriging, el cual utilizaba como entrada un modelo de correlación espacial, llamado el modelo por variograma. Sin embargo, las limitaciones del kriging al ser aplicadas a un modelado de yacimiento son ahora bien conocidas. La distribución espacial de los valores estimados del kriging tienden a suavizar mucho, adicionalmente este suavizado es no estacionario, lo que produce artefactos en las localizaciones cercanas a los pozos. Como resultado de este suavizado artificial, el kriging tiende a sobreestimar las propiedades petrofísicas bajas y a subestimar las altas. Esta propiedad del kriging puede ser un verdadero problema a la hora de utilizar un modelo para calcular la respuesta de los fluidos en un yacimiento (Arbat, 2005).

Variables regionalizadas

En un gran número de actividades humanas, principalmente en las ciencias de la tierra, interesa estudiar la variación espacial de ciertas magnitudes, que llamaremos de manera general, variables regionalizadas. Estas variables tienen ciertas características que las diferencias de otras, las cuales son: localización, continuidad, anisotropía y fenómeno de transición, este último está relacionado con los cambios buscos en la continuidad, como ejemplo simple, se puede citar, para las formaciones sedimentarias, las estratificaciones y las repeticiones lenticulares de los estratos. Puede suceder que la variable la cual es constante o casi constante al interior de los estratos, tenga cambios bruscos al pasar de un estrato a otro. (Matheron, 2008)

Variograma

En la teoría de variables regionalizadas se utiliza el concepto de semivarianza para expresar la relación entre diferentes puntos de la superficie, la cual es definida como:

$$\gamma(h) = \left[{}^{1}/_{2}\, N(h)\right] \sum \left[(z_{xi}) - (z_{xi} + h)\right]^{2}$$

Donde:

$\gamma(h) = semivarianza$

$h = \ lag\ (separación\ entre\ puntos)$

$z_{xi} \ = valor\ de\ muestra\ en\ el\ punto\ x_i$

$z_{xi} + h = \ valor\ de\ muestra\ en\ el\ punto\ x_i \ + h$

$N(h) = \ número\ total\ de\ muestras$

La semivarianza es usada para describir la tasa de cambio de una variable regionalizada como función de la distancia. Sabemos intuitivamente que no debería haber diferencia en valores (semivarianza=0) entre puntos localizados a una distancia h = 0, ya que no hay diferencia entre puntos comprados con ellos mismos, sin embargo cuando comparamos puntos que están separados cierta distancia observamos un incremento en la semivarianza (mientras más alto el valor de la semivarianza, más diferentes los valores de la propiedad entre los puntos). A medida que la distancia aumenta, la semivarianza eventualmente se vuelve aproximadamente igual a la varianza de la misma superficie. Esta distancia es la máxima distancia en donde la medida en un punto en la superficie está relacionada con este valor en otro punto.

La semivarianza se calcula mediante *γ (h)* para cada par de puntos, y asignando a cada par a un intervalo *h*. Ahora bien, si creamos un gráfico, en donde el eje y sea la semivarianza y el eje x la distancia entre puntos (lag), obtenemos un variograma (también conocido como semivariograma) como se observa en la figura 2.

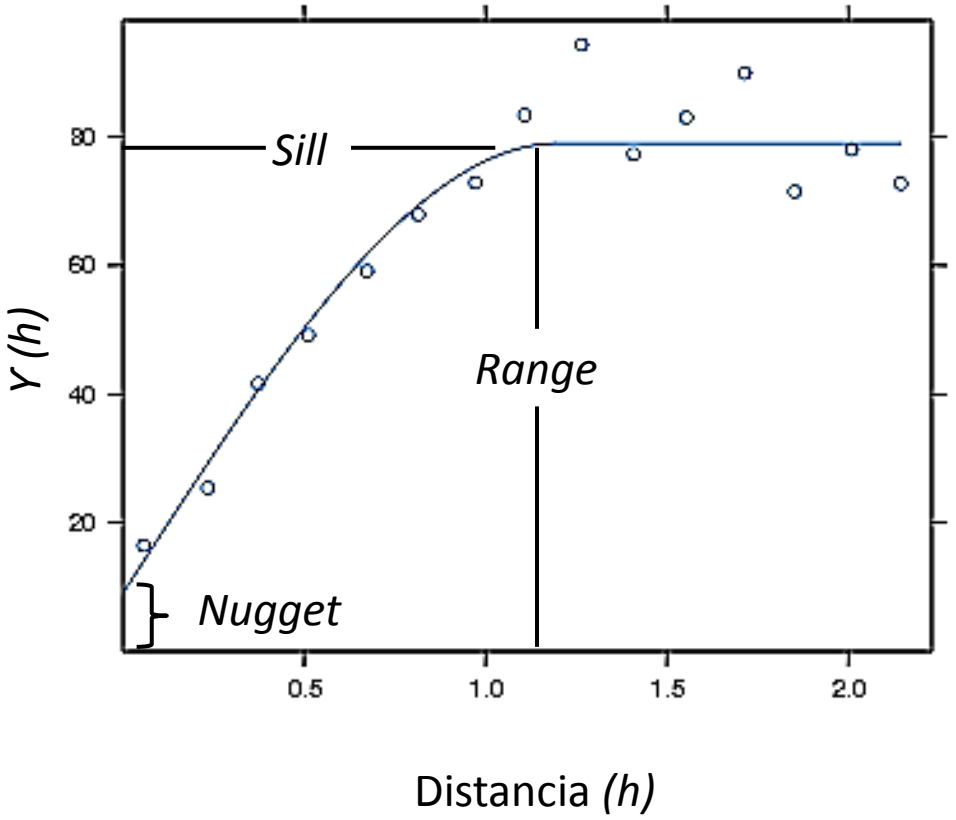


Figura 2. Ejemplo de variograma.

Características del variograma:

Rango (*Range*): Distancia a la cual el variograma se estabiliza (mantiene la pendiente)

Meseta (*Sill*): Valor constante que toma el variograma para distancias iguales o mayores al rango.

Efecto pepita (*Nugget*): Representa la variabilidad en las distancias más pequeñas que el espaciado de muestra típica, incluyendo el error de medición.

El variograma mide disimilitudes, o incremento de la varianza entre puntos (correlación decreciente) como una función de la distancia (Jimba, 2008).

Técnicas frecuentemente empleadas de interpolación de variables indicadoras de facies.

Ponderación por inverso de la distancia

La ponderación por inverso de la distancia está basada en la asunción de que valores cercanos contribuyen más a los valores interpolados que observaciones distantes. En otras palabras, para este método la influencia de un dato conocido esta inversamente relacionado con la distancia de la localización de un dato desconocido que está siendo estimado. La ventaja de este método es que se intuitivo y eficiente. Esta interpolación trabaja mejor con valores distribuidos uniformemente (Azpurua & Dos Ramos, 2010).

La forma más simple de este método es conocida como el método de Shepard, y utiliza la siguiente función de cálculo de pesos w_i:

$$w_i = \frac{h_i^{-p}}{\sum_{j=0}^{n} h_j^{-p}}$$

En donde p es un valor real y positivo arbitrario, llamado el parámetro ponderador, y h_j son las distancias entre los puntos conocidos y los puntos a interpolar (Azpurua & Dos Ramos, 2010).

Kriging de indicadores

Desde que comenzó a aplicarse la computación como herramienta versátil dentro del área de la geoestadística, particularmente en el sector de la interpolación, han tenido auge diversas metodologías de estimación de propiedades espacialmente referenciadas entre las cuales destaca el kriging (Deutsch, 2002).

El kriging es una técnica geoestadística que estima el valor de un atributo punto a punto, sobre un área o en un volumen. Es usualmente utilizado para interpolar nodos de un mallado en aplicaciones de mapeo o contorneo. En teoría, ningún otro proceso de interpolación puede ofrecer mejores estimaciones insesgadas con una mínima varianza de los errores; aunque la efectividad de la técnica realmente depende de modelar correctamente el variograma. La precisión de estimación del kriging es dada por el uso de los modelos de variograma para expresar relaciones de autocorrelación entre los puntos de control en el conjunto de datos. El kriging incluso produce un estimado de la varianza para los valores interpolados. (Jimba, 2008)

El kriging es un método para calcular estimaciones de una variable regionalizada usando una combinación lineal de pesos obtenida a partir de un modelo espacial de correlación. El mismo, asigna pesos a las muestras para minimizar la estimación de la varianza. Sin embargo, para Jimba el principal problema del kriging es que tiende a suavizar los resultados, en algunos casos de manera indeseable, subestimando los valores mínimos y máximos, igualmente refiere que es preferible utilizar este tipo de interpolación cuando se tiene soporte de gran cantidad de datos.

El kriging utiliza como insumo el variograma el cual es una medida de dependencia espacial de una variable regionalizada sobre cierta distancia; un gráfico de similitud entre puntos como una función de distancia entre los mismos. Este puede ser calculado con preferencia direccional. En otras

palabras, es una medida geoestadística usada para caracterizar la variabilidad espacial de un atributo (Jimba, 2008).

El kriging de indicadores es un método basado en la trasformación de valores continuos obtenidos del kriging tradicional en valores binarios o en valores categóricos. Es uno de los métodos no paramétricos de interpolación, que requiere los valores obtenidos del kriging tradicional como valores de entrada como probabilidad de facies, en donde posteriormente simula valores en las celdas no muestreadas para obtener valores categóricos.

Modelado de facies por objetos

Los modelos de facies basados por objetos describen la naturaleza depositacional del yacimiento mediante una serie de realizaciones de facies, simulando de forma ideal las geometrías interpretadas en afloramientos y análogos modernos, lo que lo hace visualmente atractivo e importante como descriptivo del proceso de formación del yacimiento. Estas facies corresponden a objetos geológicos bien definidos con una continuidad no rectilínea realista como se muestra en la figura 3, la cual no puede ser modelada con métodos tradicionales basados en celdas. (Deutsch, 2002)

Este método constituye una familia de técnicas que crea modelos de yacimientos basados en objetos de la misma significancia genética, en lugar de ser construido a partir de un nodo primario o un pixel a la vez. Para utilizar este método, se necesita seleccionar una forma básica para cada litofacies que describa su geometría. Por ejemplo, uno podría querer modelar canales de arena que parecieran medias elipses en la sección trasversal, o deltas como cuñas triangulares en una vista de mapa. También se debe establecer la proporción de las formas en el modelo final y escoger una distribución para los parámetros que describen estas formas. Existen algoritmos que describen como los geocuerpos están posicionados relativos entre ellos (esto es, si

están sobrepuestos y en qué forma lo están, o si debe existir una mínima distancia entre ellos) (Jimba, 2008)

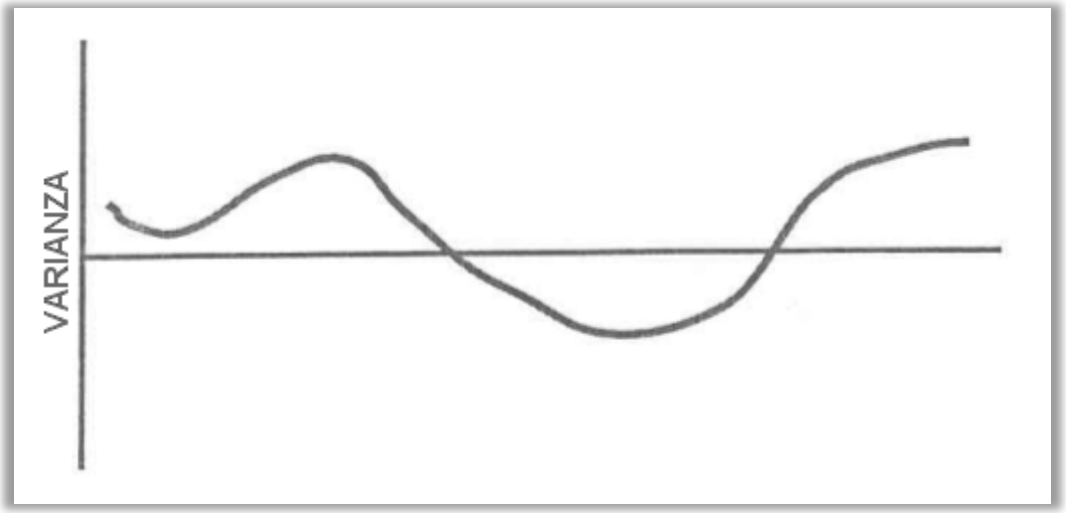


Figura 3. Modelado de un canal por objetos en donde se muestra la varianza calculada analíticamente o por simulación geoestadística (Deutsch, 2002). (El eje horizontal representa el largo del canal)

Jimba establece un algoritmo básico para la realización de un modelado por objetos, que es el siguiente:

Luego de que la distribución de parámetros y posiciones están establecidas, siguen los siguientes pasos del procedimiento:

1. Llenar el fondo del modelo con alguna litofacies (por ejemplo arcilla)
2. Aleatoriamente escoger el punto de partida en el modelo
3. Aleatoriamente seleccionar la forma de la litofacies y trazar un tamaño apropiado, anisotropía y orientación.
4. Chequear si la forma tiene conflictos con alguna de las condiciones de los datos (por ejemplo datos de pozo) (Figura 4) o con otras formas

previamente simuladas. Si no es así, mantener la forma, de otra manera rechazarla y volver al paso anterior.

5. Chequear si las proporciones globales son correctas, si no volver al paso 2

6. Simular propiedades petrofísicas en los geocuerpos usando el método geoestadístico más clásico. Si los datos de control deben ser respetados, este paso es típicamente completado al inicio y luego se simulan las regiones entre los pozos.

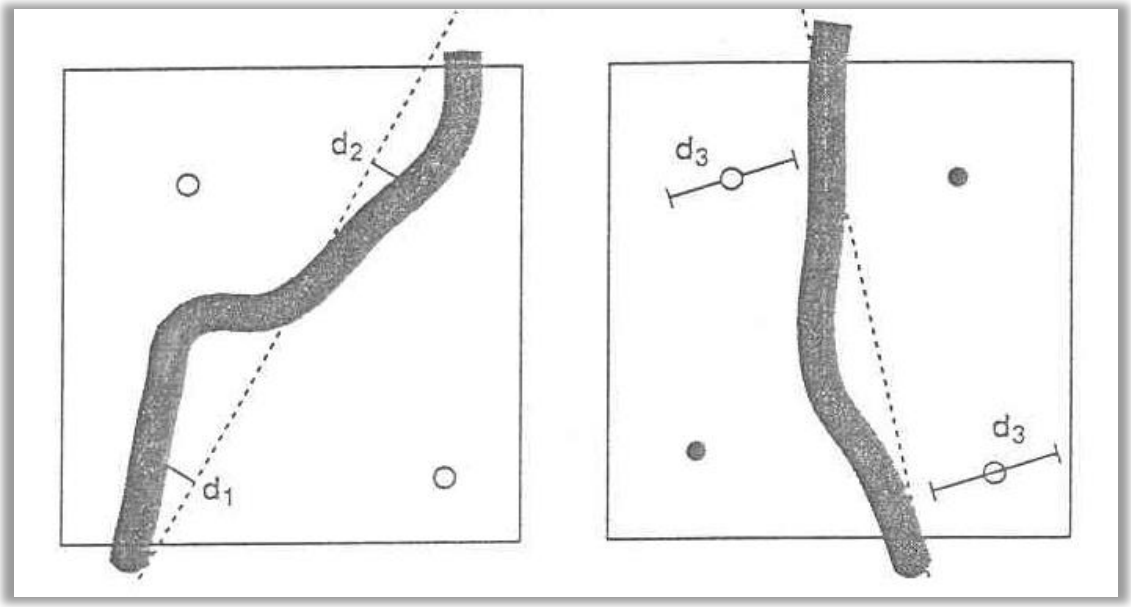


Figura 4. Dos ejemplos de la dirección condicional del modelado de canales por objetos, en donde se respetan los puntos de facies observadas. La línea punteada representa la dirección principal del canal (Deutsch, 2002).

Simulación Multipunto

La necesidad de ir más allá del modelado por objetos y de la estadística de dos puntos provista por el variograma, motiva a la búsqueda de métodos que utilicen estadística de múltiples puntos. Estos métodos podrán corregir en múltiples direcciones al mismo tiempo. Por lo tanto, en teoría, estos podrían ser capaces de reproducir la conectividad de diferentes localizaciones y así modelar complejas y curvilíneas estructuras geológicas (Arbat, 2005).

Los modelos multipunto requieren como entrada típicamente una imagen de entrenamiento. Esta imagen no debe ser condicionada en ninguna localización, lo único que necesita es reflejar la principal característica de la zona geológica (Arbat, 2005).

Imagen de entrenamiento (IE)

Una imagen de entrenamiento (IE) con patrones se define como una configuración de múltiples pixeles la cual identifica una configuración geológica significativa que se cree que existe en el yacimiento. En teoría la IE puede provenir de diferentes fuentes, como fotografías de afloramientos, o simplemente un diagrama propuesto por el geólogo, apropiadamente digitalizado como en la figura 5. En la práctica generalmente se utilizan modelos basados en objetos. (Arbat, 2005).

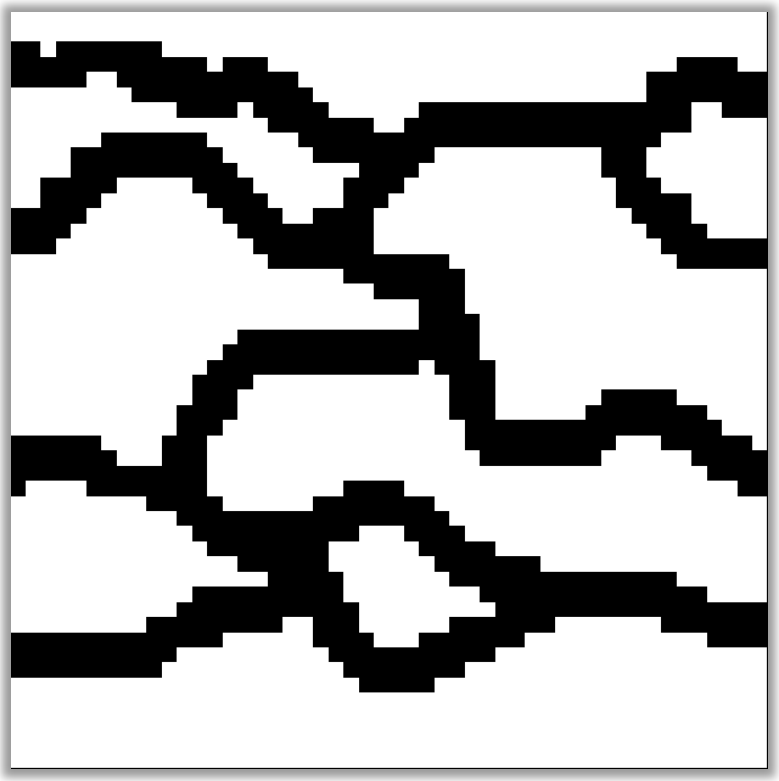


Figura 5. Ejemplo de Imagen de Entrenamiento (IE)

Se dice que es una IE con patrones ya que esta se puede descomponer en diferentes patrones que luego serán los que reconstruyan el modelo. En otras palabras en la IE se encuentran las “piezas del rompecabezas” que posteriormente al colocarlas unas con otras en el orden gobernado por el algoritmo multipunto darán como resultado el modelo final.

Estos patrones pueden ser de diferentes tamaños, siempre y cuando sean menores a la dimensión de la IE. Supongamos que tenemos una IE de dimensiones 50x50 px, esta se puede descomponer en patrones 2x2, 3x3, 4x4, hasta patrones de dimensión 49x49 px. En el ejemplo de la figura 6, observamos el primer patrón de la izquierda y arriba, obtenido de la IE de la figura 5, de dimensiones 3x3.

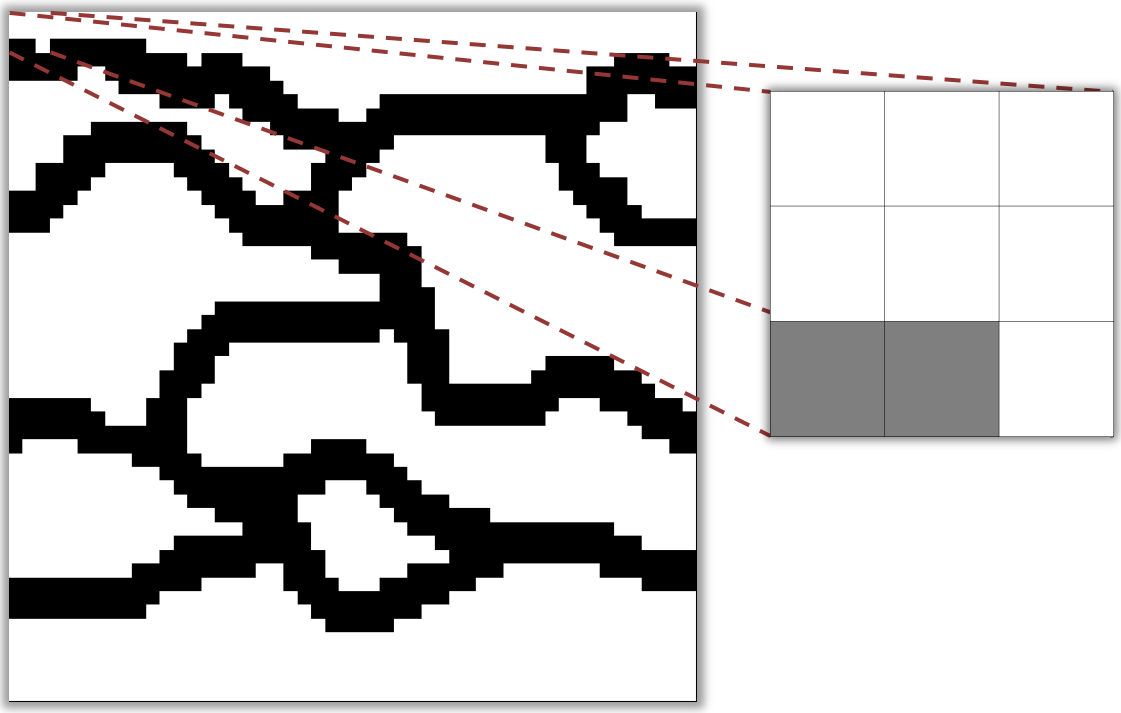


Figura 6. Ejemplo de patrón, obtenido de la IE de la figura 5 de dimensiones 3x3

Otra forma de entender la simulación multipunto es la de redefinir el problema y verlo como un método de reconstrucción de imagen, la idea no es la de reproducir explícitamente lo que muestra la IE, sino la de reproducir directamente a múltiples escalas los patrones obtenidos de la IE de una manera estocástica, (lo cual implícitamente es reproducir un modelo estadístico multipunto). Una de las principales ventajas de utilizar la aproximación de reconstrucción de imagen es la habilidad de capturar la relación entre patrones de la IE, lo que quiere decir que se utiliza una correlación entre diferentes puntos y no solo entre dos puntos como los algoritmos basados en variograma. El método de simulación multipunto puede igualmente encontrar relación entre patrón y punto, por ejemplo al encontrar un evento de dato observado (facies observada), este patrón estaría condicionado a este tipo de evento (Arbat, 2005)

Facies observadas e imagen de probabilidad (IP)

Las facies observadas son los valores en donde se simula un valor de facies. Este valor una vez establecido, mantiene su valor a lo largo de la simulación y nunca es reemplazado. En este trabajo existen dos formas de generar las facies observadas, la primera es generarlas aleatoriamente de manera uniforme en el área a simular, y la segunda forma es mediante la imagen de probabilidad.

Una imagen de probabilidad, es un esquema al igual que la IE, en donde al simular las facies de interés, estas tendrán mayor probabilidad de ocurrencia cerca de las zonas mostradas en la imagen de probabilidad, y así generar las facies observadas con cierto criterio y no con una distribución aleatoria uniforme.

Conceptos básicos.

Para entender posteriormente algunas de las metodologías empleadas, es necesario conocer ciertas terminologías básicas de la estadística, entre ellas están:

Moda.

La moda de una serie de números es aquel valor que se presenta con la mayor frecuencia, es decir, es el valor más común. La moda puede no existir, incluso si existe puede no ser única (Spiegel, 1969) Por ejemplo la moda del siguiente conjunto de datos 2,3,3,4,4,4,5,5 sería el número 4, ya que es el que más se repite. Pudiera existir el caso en que un conjunto de datos sea multimodal, esto quiere decir que se tienen dos o más valores de moda, por ejemplo en este conjunto de datos 2, 3, 3, 4 ,4 ,5 la moda seria 3 y 4, dado de cada uno de ellos se repite dos veces. Otro caso posible es que todos los valores se repitan la misma cantidad de veces, en este caso se dice que no existe moda, por ejemplo 2, 2, 3, 3, 4, 4, 5, 5, todos los valores se repiten dos

veces, lo que no da ningún tipo de sesgo para poder determinar cuál es el valor de la moda.

Probabilidad.

La probabilidad es una medida de posibilidad de ocurrencia de un evento o una medida de cuan relativamente frecuente es un evento. La medida de probabilidad esta escalada en valores de 0 a 1. Donde:

- 0 representa que no hay chance de ocurrencia del evento.
- 1 representa que hay certeza de que el evento ocurra.

La probabilidad no es más que otra herramienta que permite a los estadistas usar información de diferentes muestras para hacer inferencias o para describir la población de donde se obtuvieron estas muestras (Jimba, 2008).

Distribución binomial.

La distribución binomial es una distribución de probabilidad discreta que mide el número de éxitos en una secuencia de n ensayos de Bernoulli independientes entre sí, (este ensayo es un experimento aleatorio en el que solo se pueden obtener dos resultados, habitualmente etiquetados como éxito y fracaso), con una probabilidad p de ocurrencia de éxito entre los ensayos, y una probabilidad de 1-p de fracaso. En la distribución binomial el anterior experimento se repite n veces, de forma independiente, y se trata de calcular la probabilidad de un determinado número de éxitos. Para n=1, la binomial se convierte en una distribución Bernoulli.

Varianza.

La varianza de una variable aleatoria es una medida de la dispersión definida como la esperanza del cuadrado de la desviación de dicha variable respecto a la media. Por otro lado la desviación estándar es una medida de la dispersión de los datos con respecto a la media aritmética de los mismos, la

cual se define como la raíz cuadrada de la varianza Para variables binomiales, que serán de interés para este trabajo, la varianza está definida como p*(1-p), que es el valor de la probabilidad de ocurrencia del evento de interés, por la probabilidad de ocurrencia del evento de no-interés. La varianza tiene como valor mínimo cero, lo que indica que no hay incertidumbre y como valor máximo 0.25, indicando la mayor incertidumbre posible.

CAPITULO III

METODOLOGIA

Planteamiento de la IE.

La IE debe contener las características geológicas más representativas de la zona que se desea modelar, para esto se realiza un bosquejo digital bidimensional en donde se debe tomar en cuenta la orientación, grosor, curvatura, distribución, entre otras, de la principal morfología de las facies a modelar. En otras palabras, es un dibujo en blanco y negro como se muestra en la figura 7, que representa a groso modo el patrón principal de la zona geológica que se desea modelar.

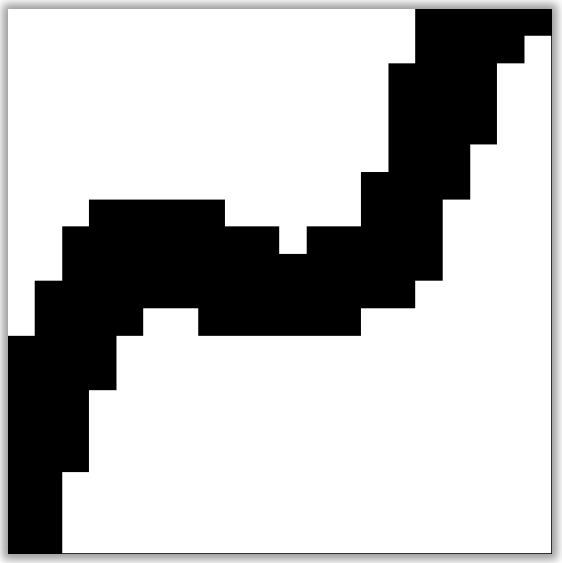


Figura 7. IE de dimensiones 20 x 20 px, representado dos facies, en donde el color negro es la facies de interés, por ejemplo arena, y el color blanco no-arena.

Tamaño del patrón.

Para descomponer la IE en patrones en necesario establecer el tamaño de estos, la cual puede hacerse de dos formas, la primera y trivial es que el usuario la establezca y la segunda es realizar un algoritmo que calcule el tamaño del patrón más óptimo, con el fin de no tener mucha cantidad de patrones repetidos y al mismo tiempo no tener patrones que contengan patrones dentro de ellos mismos. Esta idea esta esquematizada a continuación:

Algoritmo: Establecer Tamaño de Patrón Automáticamente

1. Descomponer la IE en patrones iniciando por el más pequeño 2x2 px.
2. Cuantificar la cantidad de patrones repetidos.
3. Aumentar el tamaño del patrón.
4. Volver al paso 2. hasta que existan solo patrones únicos.
5. Una vez encontrado el tamaño del patrón para que no solo existan patrones únicos, restar uno a este tamaño de patrón.

Al realizar este algoritmo nos aseguramos de que el tamaño de patrón escogido sea el más pequeño para que se tomen todos los patrones existentes en la IE, y al mismo tiempo no tener patrones tan grandes que contengan a su vez patrones dentro de ellos.

Para descomponer la IE en sus respectivos patrones, se utiliza una plantilla de dimensiones iguales al tamaño del patrón, que recorre todos los px de la IE como se muestrea en la figura 8.

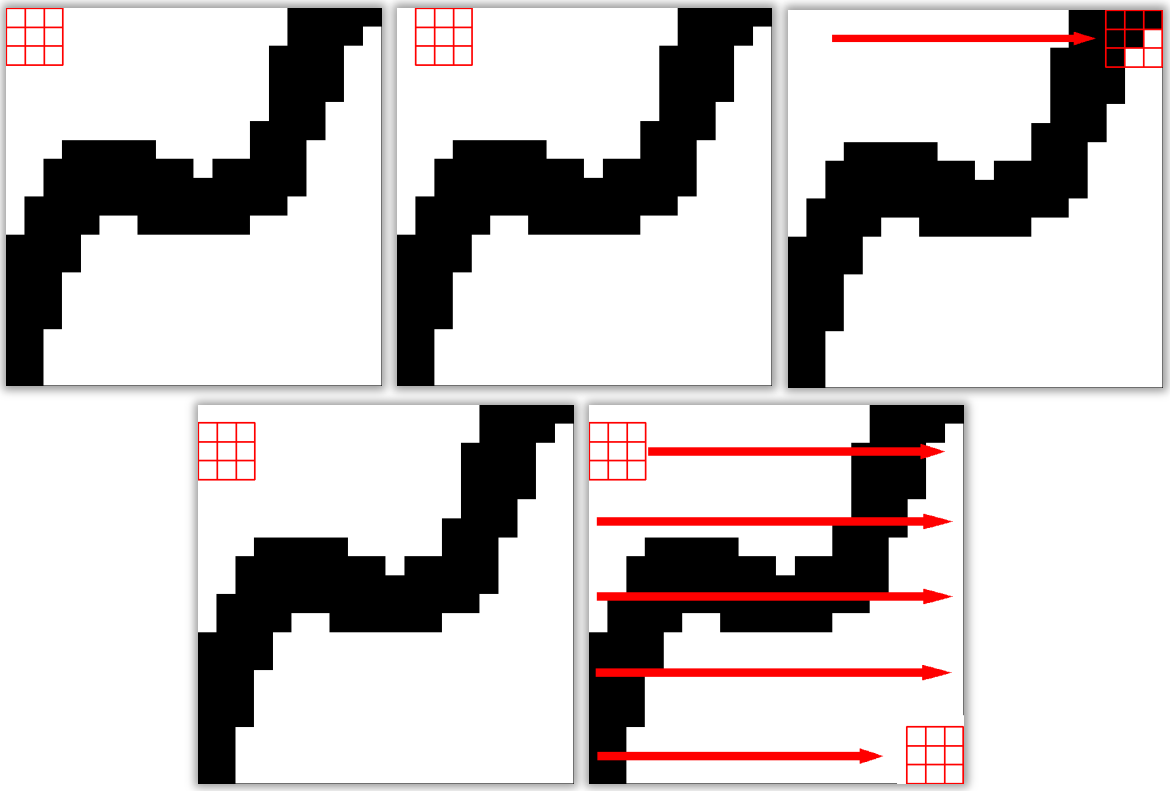


Figura 8. Recorrido de la plantilla (color rojo), de dimensiones 3x3 px, para la descomposición en patrones de la IE (Figura 7).

Representando gráficamente lo antes expuesto, al descomponer la IE (Figura 7) con un tamaño de patrón tres, obtenemos 324 patrones como se ve en la figura 9, cada uno de dimensiones 3x3 px.

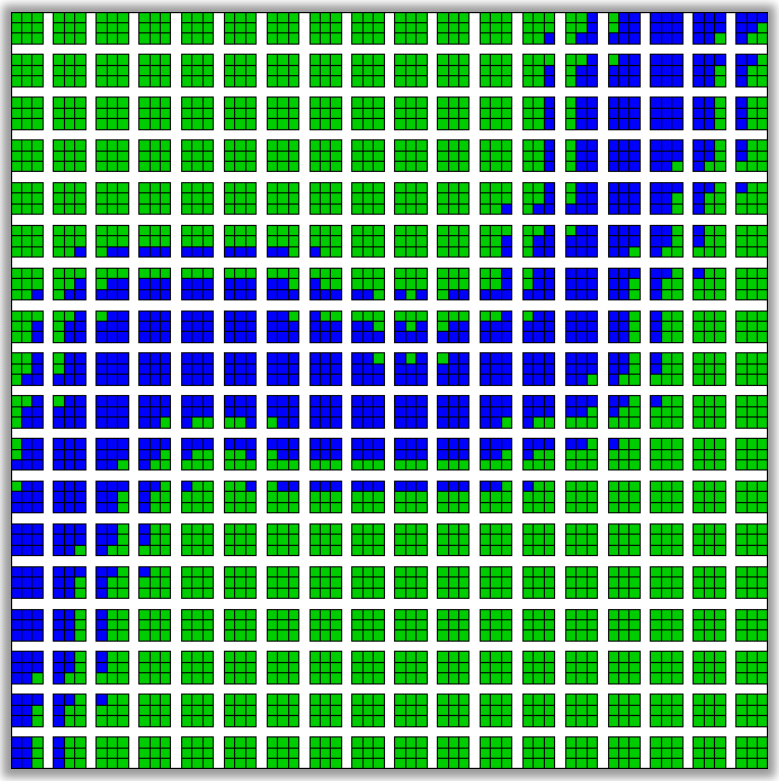


Figura 9. Patrones obtenidos a partir de la descomposición de la IE (Figura 7) con tamaño de patrón tres, en donde el color azul representa la facies de interés, por ejemplo arena, y el color verde no-arena.

El principal objetivo de variar el tamaño del patrón es el de lograr reproducir posteriormente lo que estos patrones subjetivamente aportan, al establecer un tamaño de patrón pequeño, estaríamos obteniendo patrones que representen a pequeña escala lo que muestra la IE, en otras palabras estaríamos representando de manera lo que muestra la IE de forma residual, por otro lado, si establecemos un patrón grande obtendríamos un resultado más regional de lo que representa la IE. Sin embargo es importante resaltar que los patrones deben representar lo que la IE representa y no la forma de la IE.

Establecer probabilidad de patrones.

Obtenidos todos los patrones, el siguiente paso es determinar cuántas veces se repiten cada uno de ellos.

Para facilitar la búsqueda de patrones y repeticiones se establece la siguiente nomenclatura, un vector el cual solo está compuesto por unos (1) y ceros (0), en donde los unos representan la facies de interés y los ceros lo opuesto, por ejemplo en la figura 10.a tenemos el patrón de dimensión 3x3 px, que sería representado por el vector [100000000] y el patrón de la figura 10.b por el vector [000001111].

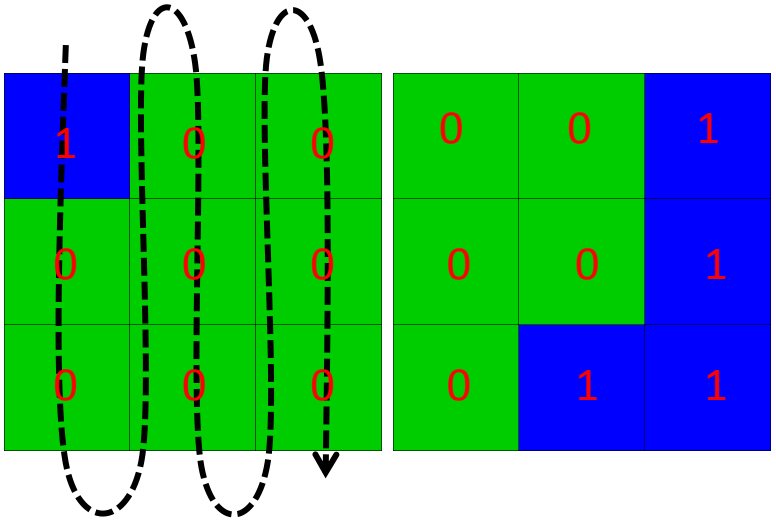


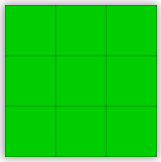
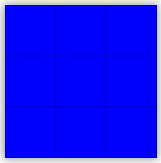
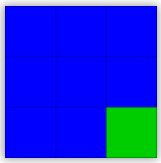
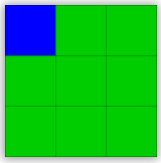
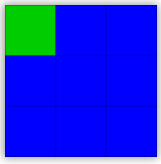
Figura 10. De izquierda a derecha: a) Patrón [100000000], b) Patrón [000001111]

Una vez entendido esto, se puede obtener con mayor eficiencia computacional, la cantidad de repeticiones por patrón, y con ellos calcular en base al total y a la cantidad de veces que se repite cada uno, la probabilidad de ocurrencia de cada patrón mediante la siguiente forma:

$$\text{Probabilidad} = \frac{\text{Nº de Repeticiones del patrón}}{\text{Total de Patrones}}$$

Al aplicar esta ecuación a cada uno de los patrones se obtiene una tabla en donde se puede observar el nombre del patrón, la cantidad de veces que se repite y su probabilidad.

Tabla 1. Ejemplo de Probabilidad de Patrones

Nombre	Patrón	Repeticiones	Probabilidad
000000000		168	0.519
111111111		29	0.090
111111110		9	0.028
100000000		8	0.025
011111111		7	0.022

Área a modelar y distribución de patrones.

Una vez obtenidos los patrones y su probabilidad, es necesaria un área en donde simular, esta área es cuadrada y las dimensiones son establecidas por el usuario, por ejemplo un área de 60x60 px, luego las posiciones de cada patrón se generan con una distribución aleatoria uniforme (Método de Montecarlo), esta posición obtenida representa la esquina superior izquierda del patrón donde posteriormente será posicionado.

Al ser la posición obtenida la esquina superior izquierda, esto genera problemas en el borde inferior y derecho del área, para evitar este problema, se expande el área en su perímetro la misma cantidad de px del tamaño del patrón elegido. Luego de simular, esta expansión es suprimida, y con esto evitamos los posibles problemas de borde.

Representado los primeros pasos de una simulación obtendríamos lo siguiente (Figura 11):

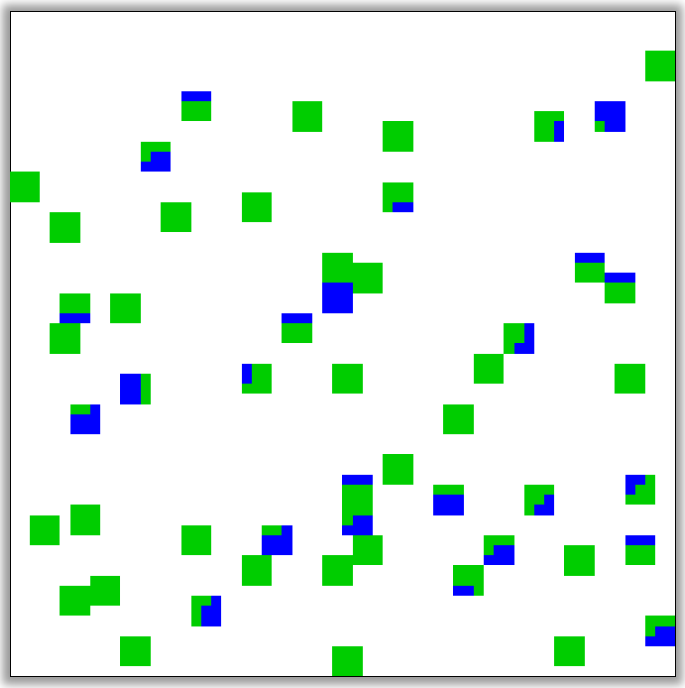


Figura 11. Primeras distribuciones de patrones.

Intersección de patrones.

Al distribuir uniformemente los patrones, existe la posibilidad donde dos o más patrones compartan el mismo espacio, cuando nos encontramos en este caso, se busca un patrón en donde los px de intersección coincidan con los anteriores y en caso de existir varios patrones que cumplan con esta condición, se hace un muestreo dependiente de su probabilidad como se muestra en la figura 12, en donde la plantilla de color rojo representa la nueva posición del nuevo patrón. En otras palabras es igual a armar un rompecabezas en donde se debe buscar una pieza que encaje con sus vecinas.

De no existir un patrón que encaje, entonces se busca el patrón más similar sin reemplazar los valores ya simulados.

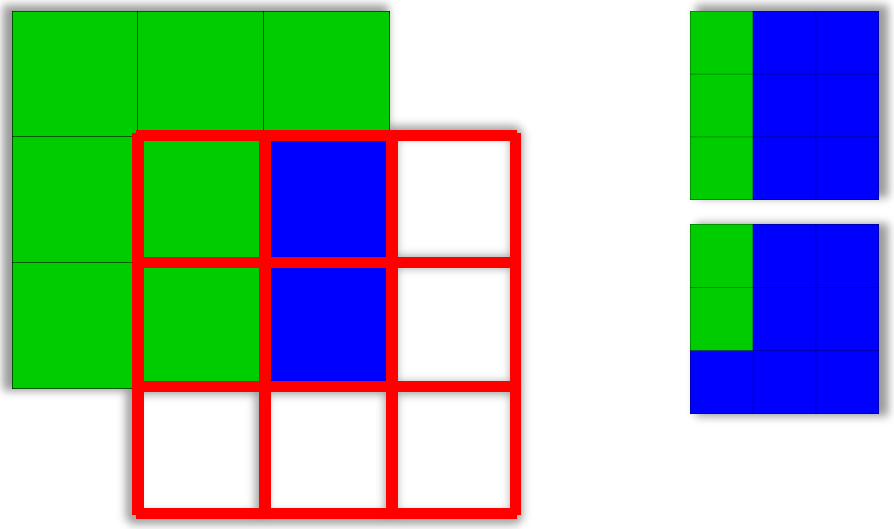


Figura 12. Intersección de patrones, mostrando a la derecha las posibles opciones a reemplazar.

Corrección de modelo.

Los patrones al ser distribuidos aleatoriamente pueden generar incongruencias, esto se debe a que en algunos casos los patrones caen unos

al lado de otros, dando como resultado una imagen sin correspondencia o continuidad, debido a esto debe realizarse un proceso en donde se corrijan este tipo de casos.

Para representar mejor este problema, podemos observar como en la figura 13.a tenemos un modelo sin correcciones y luego en la figura 13.b un modelo corregido. A este tipo de procedimiento de corrección lo llamaremos posteriormente “procedimiento de limpieza”.

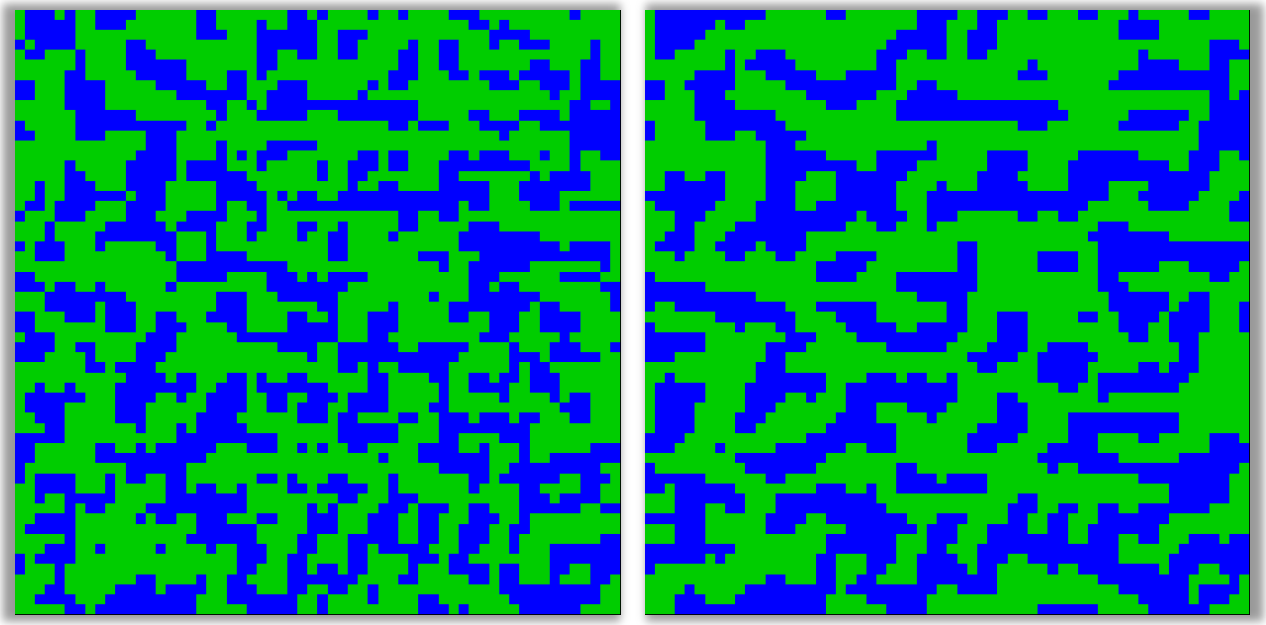


Figura 13 a) Modelo sin corrección, b) Modelo luego de la corrección. Facies de interés representada por color azul.

Estas correcciones consisten en recorrer el modelo realizado, con un tamaño de patrón, igual o menor, al seleccionado anteriormente, y al encontrar patrones que no existen dentro de la lista de patrones de la IE, estos son cambiados por uno que si exista y que encaje correctamente, con esto se solucionan los posibles errores de incongruencia.

CONFIGURACIONES OPCIONALES

Imagen de probabilidad y facies sintéticas observadas.

La imagen de probabilidad es un bosquejo, al igual que la IE, de dimensión igual al área a modelar, la cual representa una idea general de como el usuario estima que están distribuidas las facies en el área a modelar (Figura 14), esta imagen de probabilidad luego pasa por un proceso para distribuir su probabilidad por medio del método de inverso de la distancia (Figura 15), lo que hace este método es distribuir en forma de gradiente las probabilidades de encontrar la facies de interés y así darle mayor libertad al algoritmo para simular y no establecer límites tan estrictos.

Figura 14. Imagen de Probabilidad

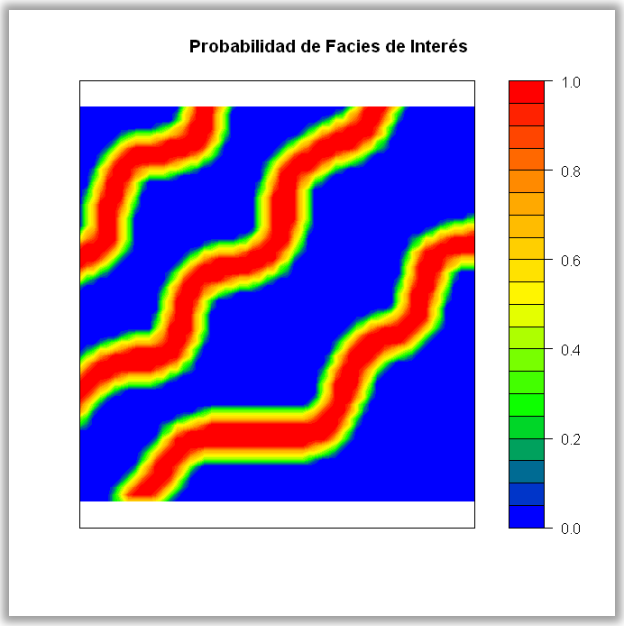


Figura 15. Mapa de Inverso de la Distancia de la IP, con número de vecinos igualo a uno

Una vez obtenido el mapa de distribución de probabilidad de la facies de interés, se toma los números de observaciones que el usuario decida y condicional en la probabilidad estas facies serán de interés o no. En el ejemplo de la figura 16, tenemos tres mapas diferentes para mostrar el comportamiento del muestreo de facies de interés, en donde en la primera tenemos el 5% del área observada, luego el 20% y por último el 50% de facies observadas dependientes de la distribución de probabilidad de la imagen de probabilidad.

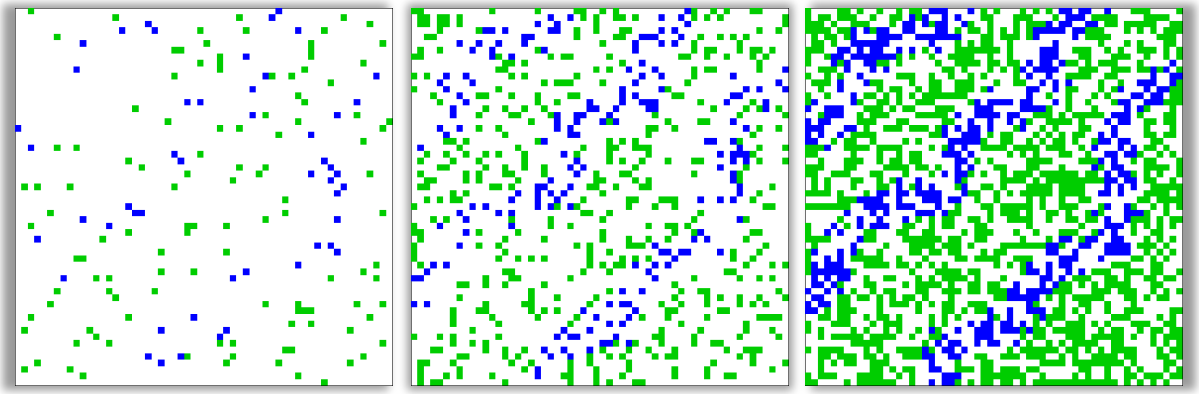


Figura 16. Representación del 5%, 20% y 50 % de facies observadas dependientes de la IP.

Al tener las facies observadas, estas siempre se mantendrán, lo que quiere decir que al simular estos valores siempre serán respetados y nunca cambiaran su valor.

Este mapa de facies observadas sería sustituido por el área del modelo antes de empezar a simular, lo que solo daría libertad al algoritmo de simular en los espacios en blanco condicionales en las facies observadas. Tomando la imagen de probabilidad (Figura 14) y su respectiva distribución de probabilidad, simulando y corrigiendo, con el 20% de facies observadas obtendremos como resultado, por ejemplo, el modelo de la figura 17.

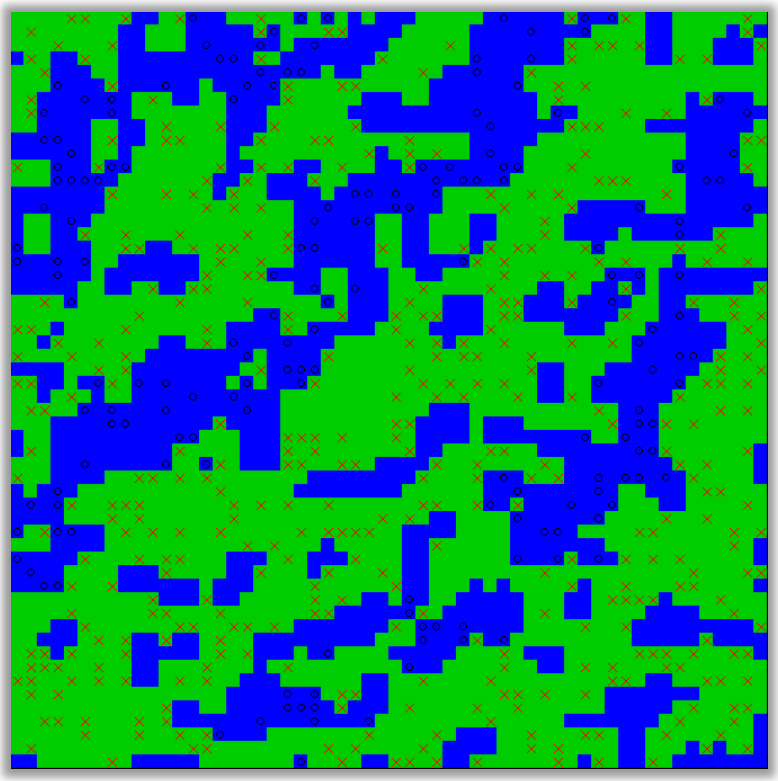


Figura 17. Modelo en donde se observan las facies por círculos y equis, representado facies de interés y facies de no-interés respectivamente.

Plantilla gruesa – Patrones para detectar comportamientos regionales.

Para representar de manera regional lo que la IE muestra, una opción es utilizar una plantilla gruesa, esta es igual a la plantilla antes utilizada con la excepción de que esta tiene espacios vacíos de por medio (Figura 18), los cuales son definidos por el usuario.

Al descomponer de esta forma la IE, se obtienen lo que llamaremos “patrones gruesos” o “patrones regionales”, los cuales luego de simular, aplicando la misma metodología antes expuesta, nos dará como resultado el área del modelo o mallado con espacios vacíos de por medio como se muestra en la figura 19.

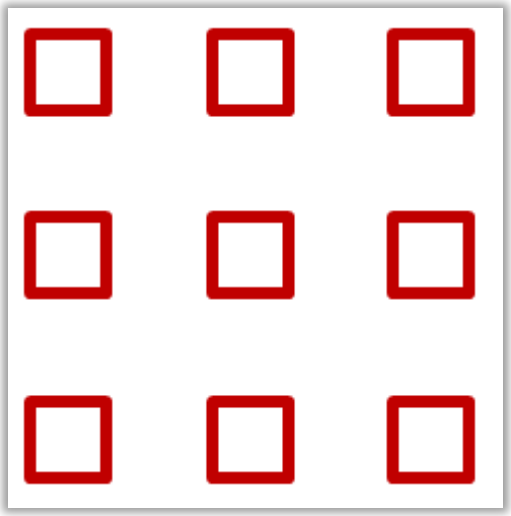


Figura 18. Plantilla gruesa con separación uno

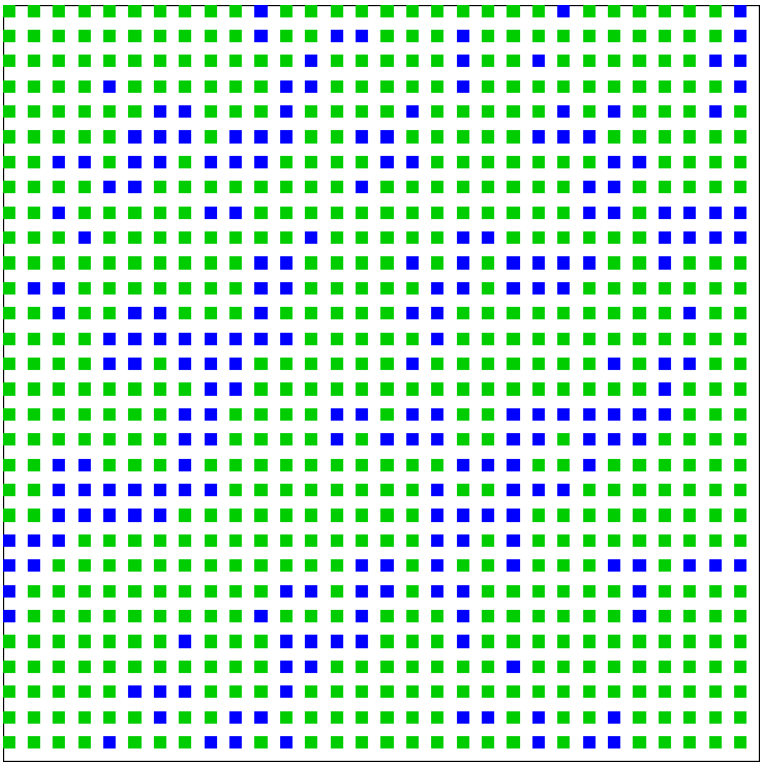


Figura 19. Ejemplo de mallado grueso.

Una vez realizado el mallado grueso, se procede a rellenar los espacios vacíos con patrones finos, los cuales estarán condicionados por los valores

ya existentes provenientes del mallado grueso, con la excepción de que al no encontrar ningún patrón similar que encaje, entonces puede reemplazar los valores del mallado grueso, (siempre y cuando no provengan del mapa de facies observadas) por los patrones del mallado fino, de ser necesario, como ejemplo de un modelo final la figura 20.

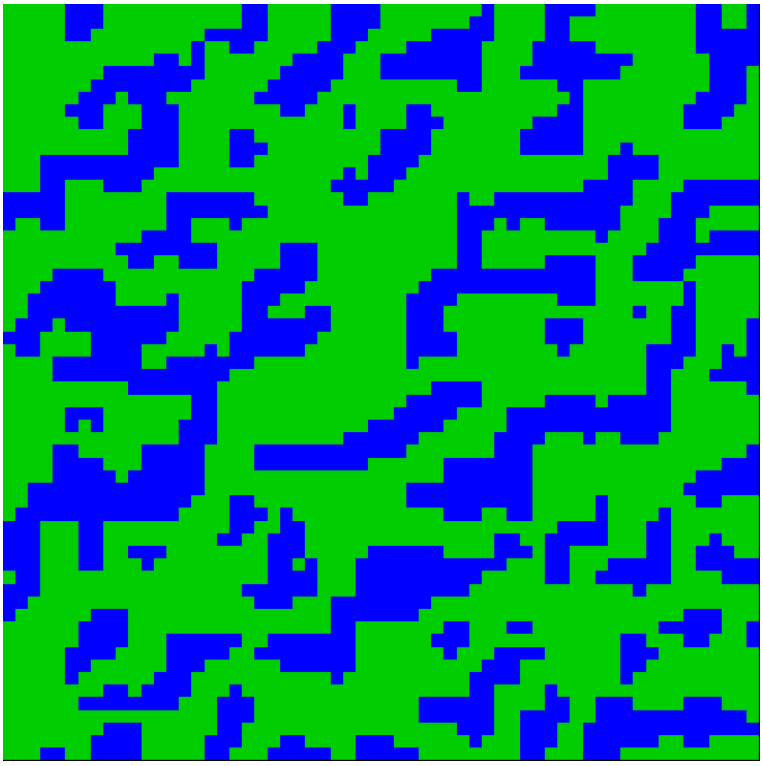


Figura 20. Modelo realizado con plantilla gruesa, con separación uno.

Mapas MPV (Moda Probabilidad y Varianza)

Una vez generado el modelo, es necesario determinar si es geológicamente correcto, hablando técnicamente cualquiera de las realizaciones arroja como resultado una de las posibles distribuciones de las facies del yacimiento, ya que cada uno de los modelos generados es igualmente probable por ser independientes, cada uno de ellos cumple con las condiciones preestablecidas, pero dado a que es estadísticamente probable, esto no quiere decir que sea geológicamente aceptable (Jimba, 2008).

Jimba (2008) aconseja realizar varios modelos y con estos crear mapas que logren generalizar lo que indican cada uno de estos modelos, la forma de hacer esto es mediante los mapas de moda, probabilidad y varianza.

Supongamos que generamos cinco realizaciones de dimensiones 90x90, las cuales nos arrojarían cinco modelos diferentes, en donde cada uno es igualmente probable, ya que cumplen con las condiciones de facies observadas y regidos por la IE.

La metodología para obtener el mapa de moda es simple y se basa en obtener la moda estadística por pixel de todos los modelos como se muestra en la figura19, la forma de hacerlo es ver los modelos como matrices, en este ejemplo de matrices de dimensiones 90x90, en donde las facies de interés serán unos y las facies de no-interés serán ceros, al sacar la moda por cada pixel obtendremos tres opciones de resultados, la primera en donde la moda es la facies de interés (1), la segunda donde la moda es la facies de no-interés (0), y la tercera opción en donde la cantidad de 1 y 0 encontrada es igual, entonces establecemos que al ocurrir este tipo de caso simplemente elija aleatoriamente una de las dos opciones con 50% de probabilidad cada una. En la figura 22 se observa un ejemplo de un mapa de moda obtenido a partir de los cinco modelos de la figura 21.

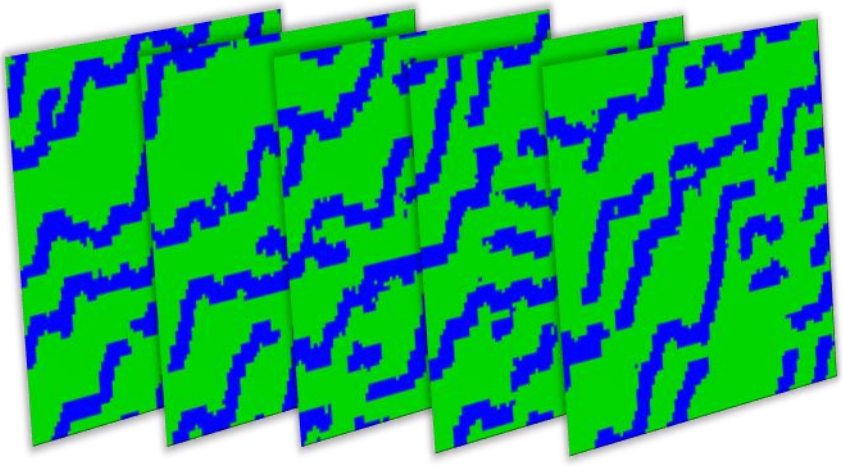


Figura 21. Ejemplo de 5 realizaciones de dimensiones 90x90.

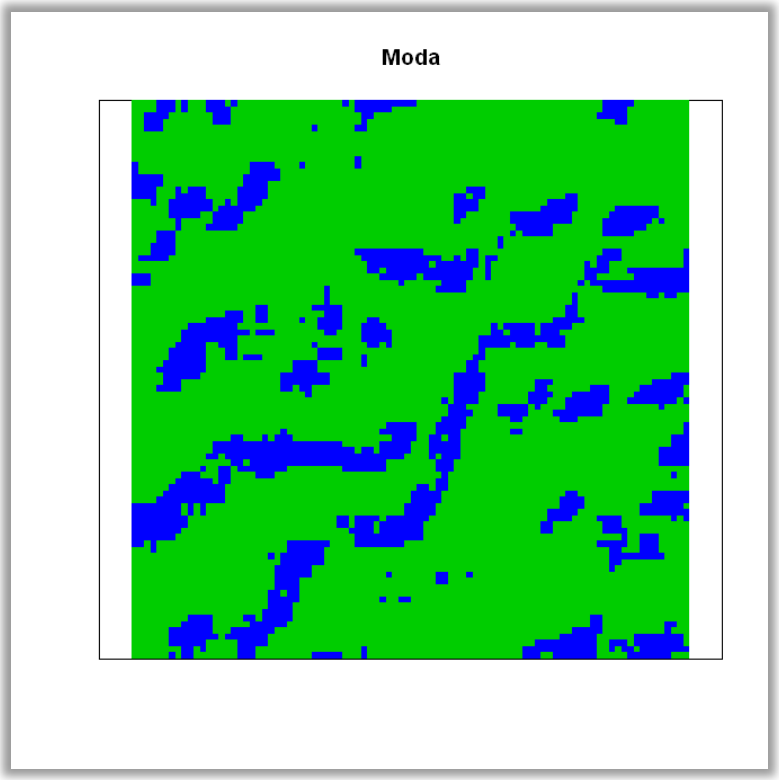


Figura 22. Ejemplo de Mapa de moda.

Para calcular el mapa de probabilidad, se necesita sumar los mapas y normalizar los valores en un rango del cero al uno, que indiquen los valores de probabilidad, pero al ser solo valores de 1 y 0, no es necesario normalizar, solo con un simple promedio se logra obtener el mapa de probabilidad, entones el procedimiento, continuando con el ejemplo anterior, sería, sacar el promedio con cada una de las cinco matrices que representan cada modelo (Figura 21), que sería sumar los valores de cada pixel y dividirlo entre la cantidad de modelos totales. Esto nos daría como resultado un mapa de probabilidad como el de la figura 23.

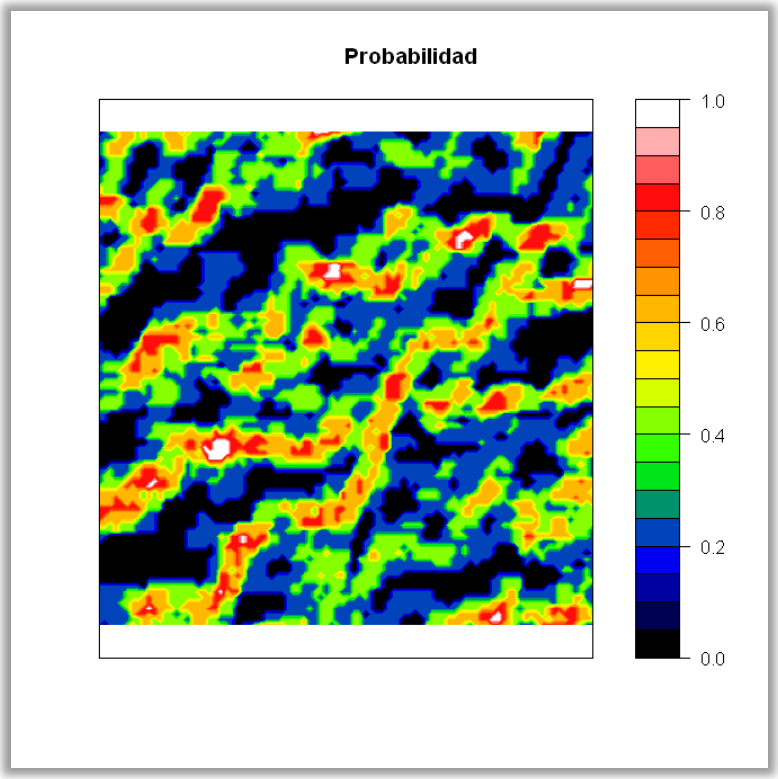


Figura 23. Ejemplo de Mapa de probabilidad.

Por ultimo restaría calcular el mapa de varianza, el mismo se calcula a partir del mapa de probabilidad, ya que lo único necesario para obtenerlo es la probabilidad por pixel (p) y el valor de uno menos esta probabilidad (1-p) (Distribución Bernoulli), al multiplicar estos valores p*(1-p), obtenemos los

valores de varianza asociada a cada uno de los px como se muestra en la figura 24. Posteriormente, para obtener una mejor idea de cuánto es la varianza total del mapa resultante, se obtiene este valor normalizado entre cero y uno, en donde cero representa la existencia total de certidumbre y uno lo opuesto, esto se hace sumando la varianza de todo el mapa y dividiendo entre la cantidad de observaciones o px en el mapa, y luego multiplicando por el valor máximo de varianza máxima (0.25) para lograr normalizar los valores entre el intervalo deseado. Este valor nos permite conocer y poder comparar cuánta incertidumbre existe en el mapa.

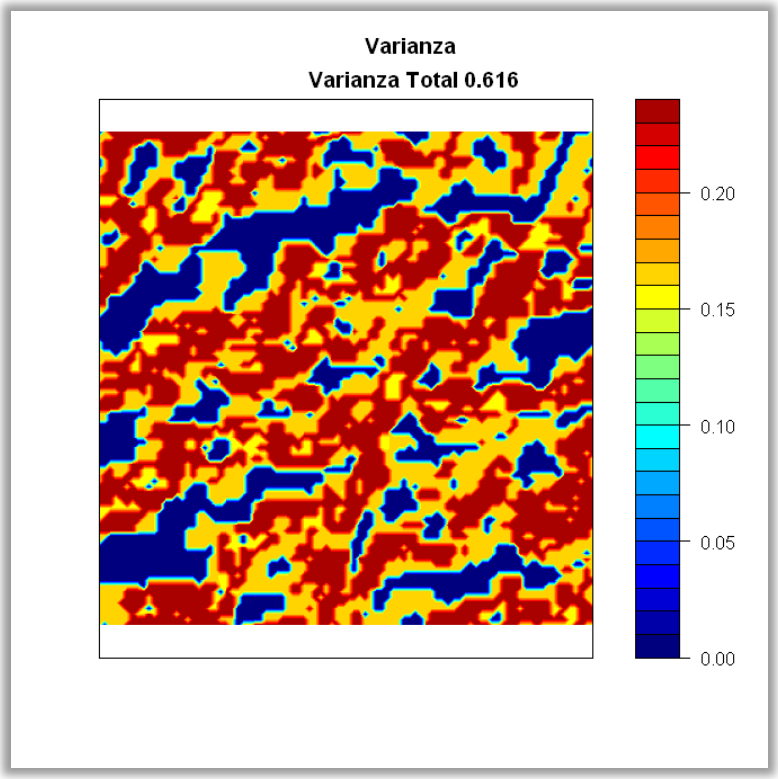


Figura 24. Ejemplo de Mapa de Varianza.

CAPITULO IV

RESULTADOS Y ANALISIS

Para entender el comportamiento del simulador, y de los diferentes resultados que este puede arrojar, se realizaron diferentes pruebas, las cuales tienen como finalidad establecer el impacto de cada una de las variables sobre los modelos resultantes.

Las diferentes pruebas realizadas se enumeran a continuación:

1) Comparación entre tamaño del patrón definido automáticamente con patrones menores y mayores.
2) Variación del tamaño del área a modelar.
3) Coherencia en la distribución espacial de las facies observadas y errores asociados.
4) Diferencias en los modelos al utilizar patrones simples y regionales.
5) Relación entre proporción de facies de interés en IE, facies observadas y modelos resultantes.
6) Cantidad optima de modelos a realizar.

Comparación entre tamaño del patrón definido automáticamente con patrones menores y mayores.

En esta sección hablaremos de por qué es eficiente utilizar el método automático para calcular el tamaño de patrón, y para entenderlo más fácilmente lo demostraremos con varios ejemplos que muestran las principales diferencias entre los modelos al simularlos con diferentes tamaños de patrones.

La idea principal de utilizar el cálculo de patrones automáticos es el de obtener óptimamente los patrones más representativos de la IE y así representar en los modelos lo que esta IE realmente aporta.

Al generar los patrones de la IE de la figura 7, de dimensiones 20x20 px, con un tamaño de patrón 2x2 px, luego con un tamaño 4x4 px y así sucesivamente solo con múltiplos de dos, hasta llegar a patrones de 18x18 px, en un área de modelo de 60x60 px. Observando los modelos generados con cada uno de estos patrones sin aplicar la corrección de “limpieza” se obtiene lo que se muestra en la figura 25, en donde el patrón automático es de dimensiones 8x8 px. Ahora bien, si observamos los modelos que están generados con patrones más pequeños que el automático, nos damos cuenta de que estos son en general muy ruidosos y poco consistentes con lo que se espera que represente la IE, sin embargo, a medida que el tamaño de patrones se acerca al del método automático (8x8 px), los modelos tienden a ser más coherentes. Por otro lado, los modelos generados con patrones mayores al automáticos muestran la falsa ilusión de ser modelos que realmente representan lo que muestra la IE, sin embargo, la idea principal no es la de representar la IE como tal, sino la de representar de forma estocástica los patrones que la IE contiene.

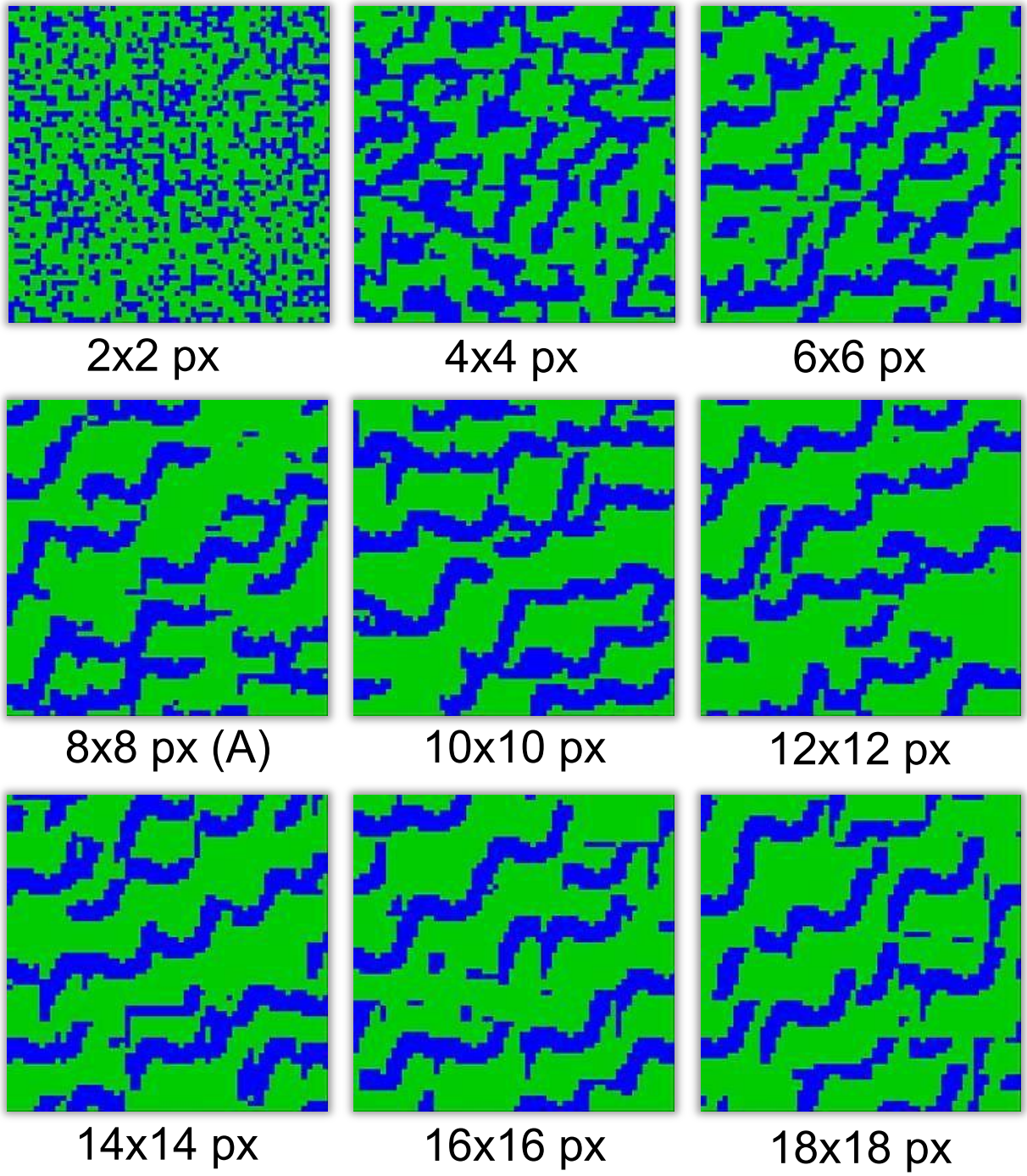


Figura 25. Modelos generados a partir de la IE de entrenamiento de la figura 7, con diferentes tamaños de patrones. (A = Automático).

Al utilizar el tamaño de patrón calculado automáticamente nos aseguramos de obtener los patrones que no sean pequeños como para generar modelos inconsistentes con la IE, ni tan grandes que repliquen exactamente lo que la IE muestra. En otras palabras el tamaño de patrón calculado automáticamente es el tamaño óptimo para descomponer la IE en los patrones que esta puede aportar y con ellos lograr simular y obtener un modelo que represente eficientemente los patrones de la IE.

Variación del tamaño del área a modelar.

Al diseñar un nuevo modelo es necesario establecer las dimensiones del área en la cual se simulara, en teoría esta área puede ser de cualquier dimensión, pero al tener la IE y sus respectivos patrones preestablecidos, resulta obvio entender que esta área debe ser al menos de la misma dimensión que la IE, aunque no necesariamente, esto es debido a que los patrones al no tener el espacio necesario, estos dentro de la aleatoriedad del simulado, no podrán generar un modelo que sea similar a lo que se espera.

Otro punto importante al determinar el área a modelar, es la escala en la que se quiere que se observen las facies, por ejemplo si establecemos que la relación entre la IE y el área a modelar sea dos a uno, esto quiere decir que el área a modelar es dos veces el área de la IE, supongamos que la IE tiene dimensiones 20x20 px, el doble de esta área seria 40x40 px, lo que haría posible que la IE se reproduzca 4 veces dentro del área a modelar como se muestra en la figura 26. La idea de entender este tipo de efecto es que en el modelo resultante se verá escalado en relación a cuantas veces mayor sea el área a modelar, continuando con la idea del ejemplo anterior al ser 2 veces más grande el área que la IE, entonces en el modelo se verán las facies un cuarto del tamaño en relación a lo representado en la IE como se muestra en la figura 27.

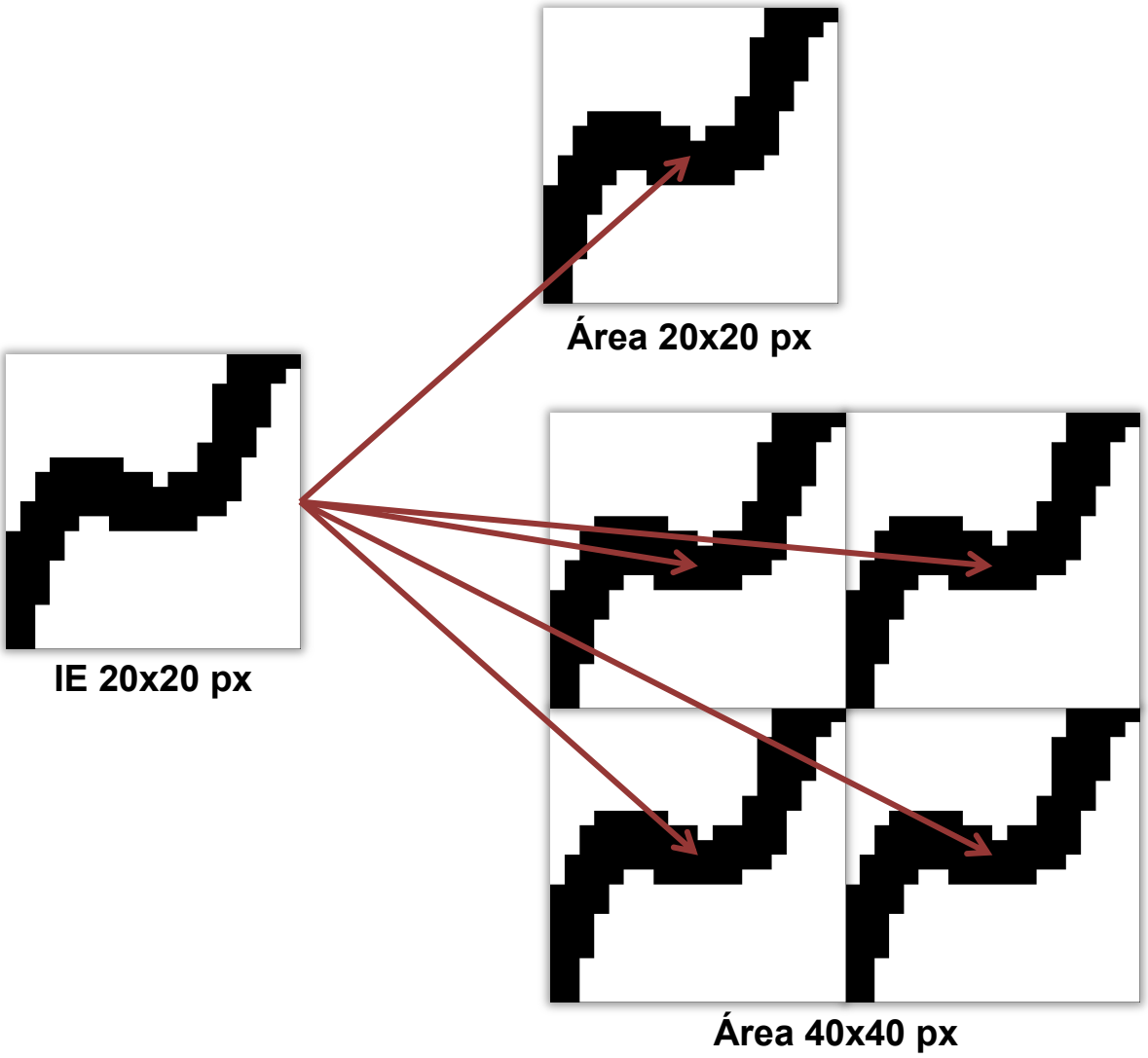


Figura 26. Relación de visualización entre IE y área a modelar.

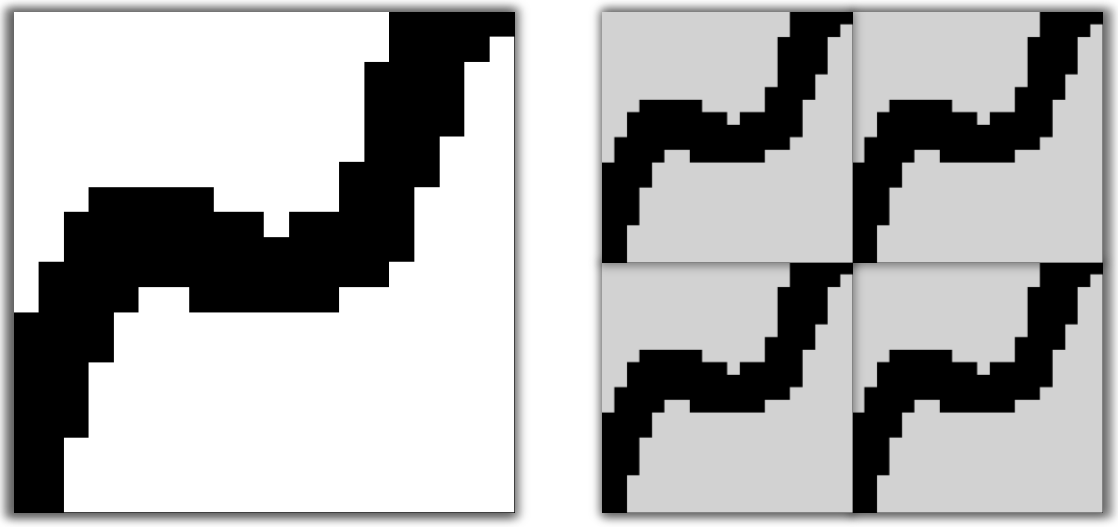


Figura 27. Relación dos a uno entre IE y área a modelar.

Coherencia en la distribución espacial de las facies observadas y errores asociados.

Al generar facies observadas sintéticas, estas pueden influir en gran parte la morfología final del modelo, como ya sabemos estas facies deben ser respetadas y nunca cambiar su valor a lo largo de la simulación, debido a esto, si estas facies no tienen relación directa con lo que representa la IE, el algoritmo, al simular, no tendrá otra opción que generar incoherencias en algunas zonas, al no lograr conectar estas facies, dando un modelo en donde las facies observadas quedan aisladas del resto del modelo o en lugares no esperados. Para demostrar este tipo de fenómeno, utilizamos un muestreo de facies con una configuración de tablero de ajedrez para representar una distribución no coherente, como se ve en la figura 28, de esta imagen se tomó aleatoriamente el 2%, 5% y 10% de estos puntos y se generaron tres modelos con la IE de la figura 29.

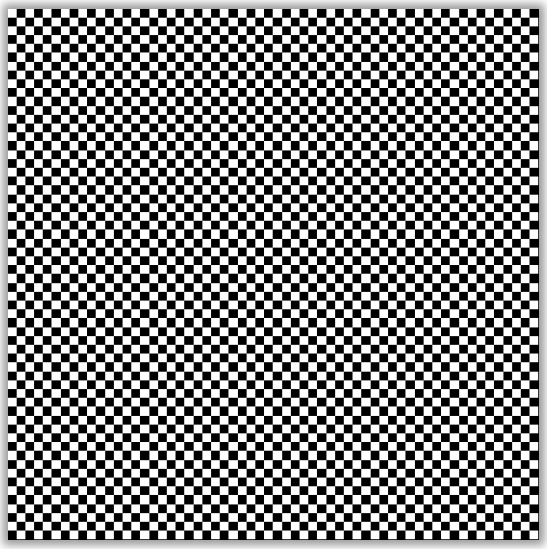


Figura 28. Configuración de distribución de facies, donde el color negro representa facies de interés y el color blanco lo opuesto.

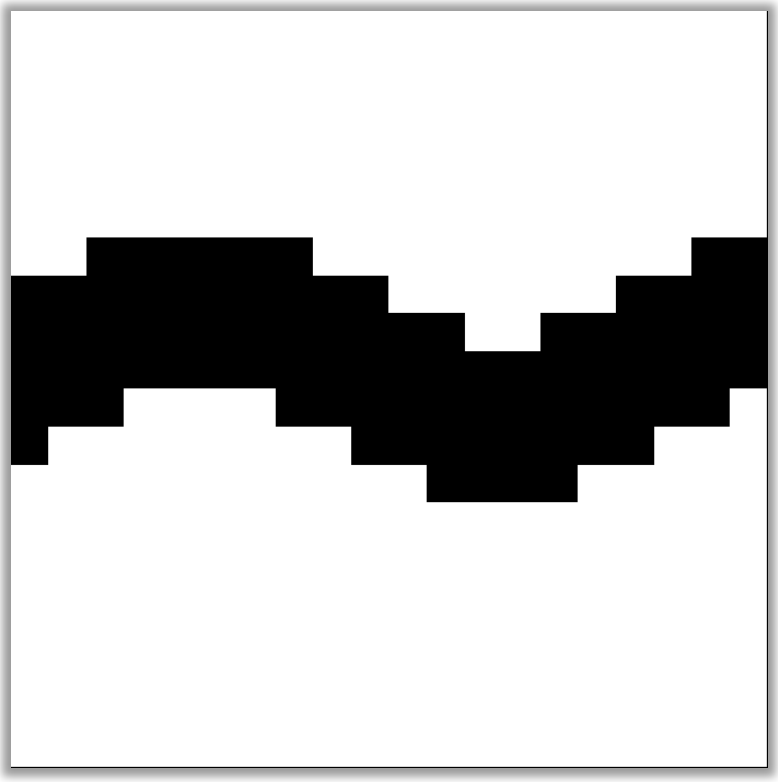


Figura 29.IE en forma de meandro horizontal.

Al generar los modelos se observó que estos presentaban los errores esperados dando facies observadas aisladas como se muestra en la figura 30, también podemos darnos cuenta que al aumentar el número de facies observadas las conexiones son más difíciles de hacer para el algoritmo mostrando aún más incoherencias del mismo tipo.

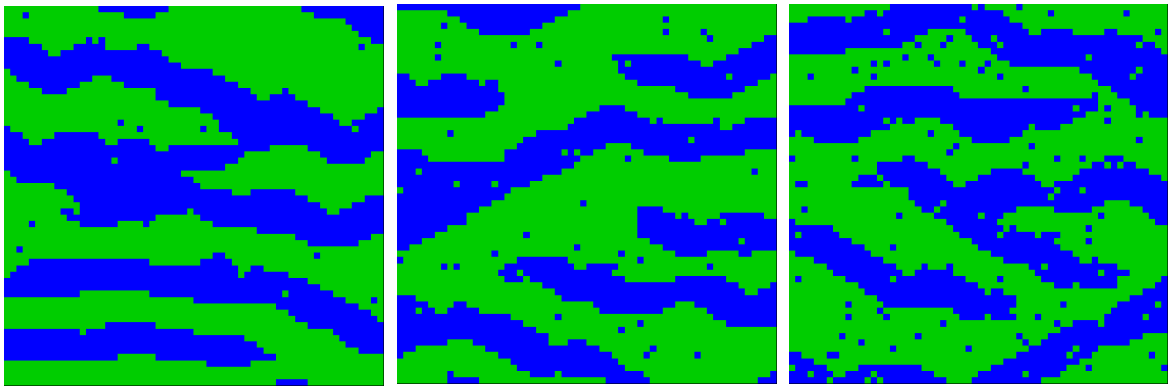


Figura 30. Modelos realizados con IE en forma de meandro horizontal (Figura 29) y de izquierda a derecha 2%, 5% y 10% de facies observadas.

Al utilizar un conjunto de facies observadas que si sean coherentes con la IE, podemos encontrar que este tipo de errores disminuye notablemente, en la figura 31 observamos una distribución de facies con una orientación de facies de interés que va desde abajo a la izquierda hasta arriba a la derecha, rodeada de facies de no-interés, con el fin de generar modelos que no produzcan los errores antes mencionados, generamos cinco modelos con la IE de la figura 7, la cual tiene una orientación similar a las facies observadas, dando como resultado modelos con muy pocos errores de incoherencias y de facies observada aislada como se muestra en la figura 32.

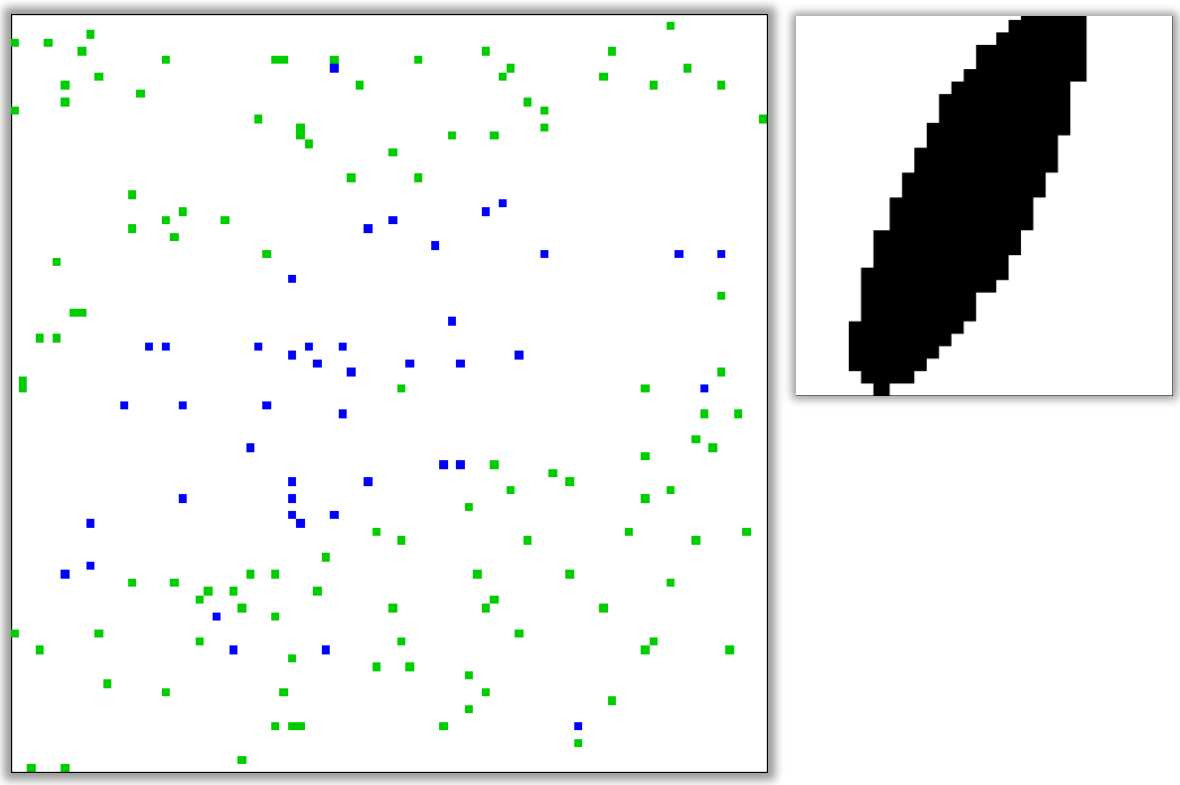


Figura 31. 5% de facies observadas sintéticas con orientación izquierda abajo – derecha arriba tomada de la IP ubicada a la derecha.

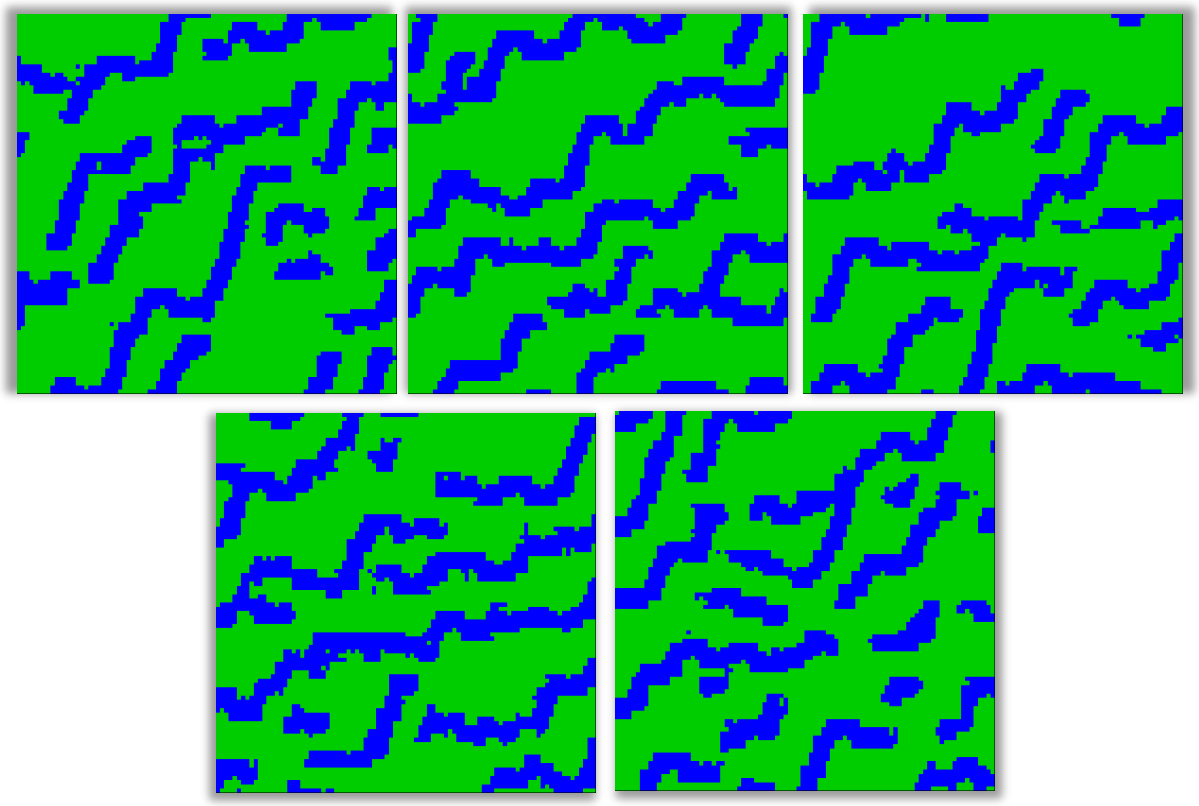


Figura 32. Cinco modelos obtenidos a partir de la IE de la figura 7, con facies observadas de la figura 31.

Diferencias en los modelos al utilizar patrones simples y regionales.

Al generar modelos con patrones simples, obtenemos modelos en donde la IE es representada óptimamente como se ha explicado anteriormente, pero existen casos en donde al descomponer la IE en patrones, estos no son los ideales para generar modelos que representen lo que esta muestra. Supongamos que tenemos las IE de la figura 33, y las descomponemos en patrones simples con el cálculo de tamaño de patrón automático, dando un tamaño de 10x10 px, al hacer esto, los patrones obtenidos serían muy similares entre las dos IE, como se muestrea en la figura 34.a y 34.b, la única diferencia significativa seria que la probabilidad de cada uno de los patrones que contienen la facies de interés serían más probables y no se modelaría tanta cantidad de facies de no-interés en la IE que tiene dos meandros (derecha).

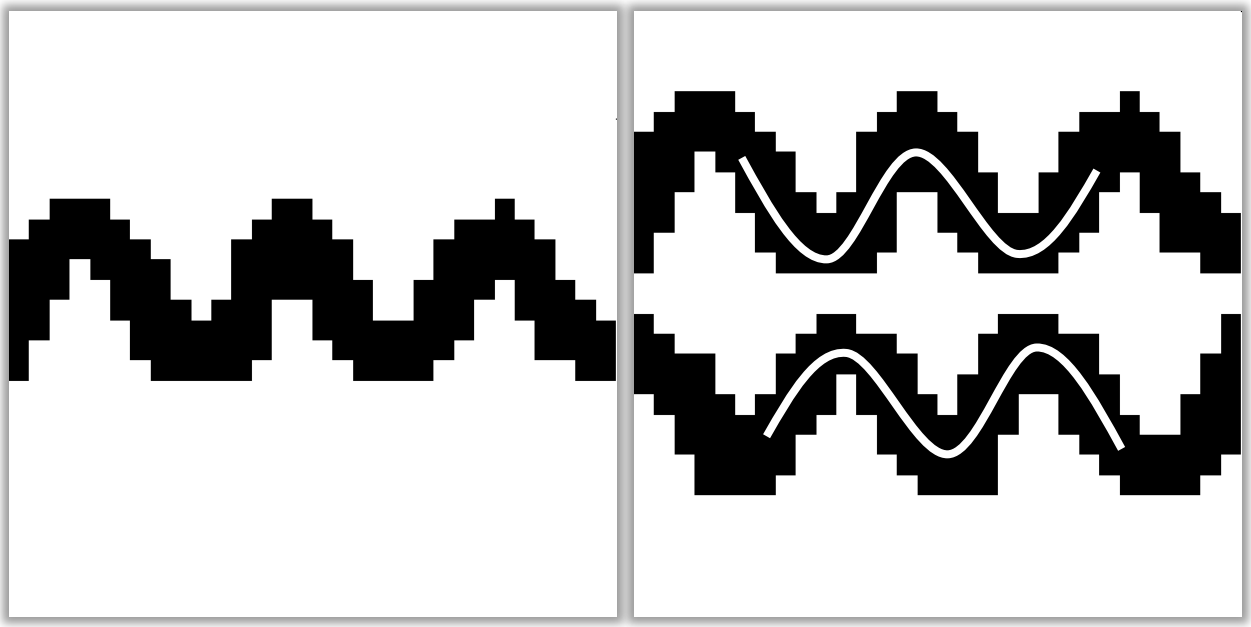


Figura 33.Ejemplos de IE, para análisis de modelado regional. a) Un meandro, b) 2 meandros.

Otra forma de comprobar que el patrón regional es más eficiente para representar este tipo de IE, es generar un modelo con patrones simples a partir de la misma IE de la figura 33.b, el modelo resultante se observa en la figura 34.c, en donde al hacer una comparación con la figura 34.b podemos darnos cuenta que en esta, se mantienen características principales de la IE, que el modelo con patrones simples no mantiene.

En la figura 34.b se resalta (blanco) algunas de las asimetrías entre los meandros, que se observan igualmente en la IE, por otro lado en el modelo realizado con patrones simples, en general no se observa este tipo de asimetría entre patrones, solo en algunos casos debido a la aleatoriedad del simulador, vale recalcar que se realizó con las mismas especificaciones, pero solo con patrones simples y no regionales.

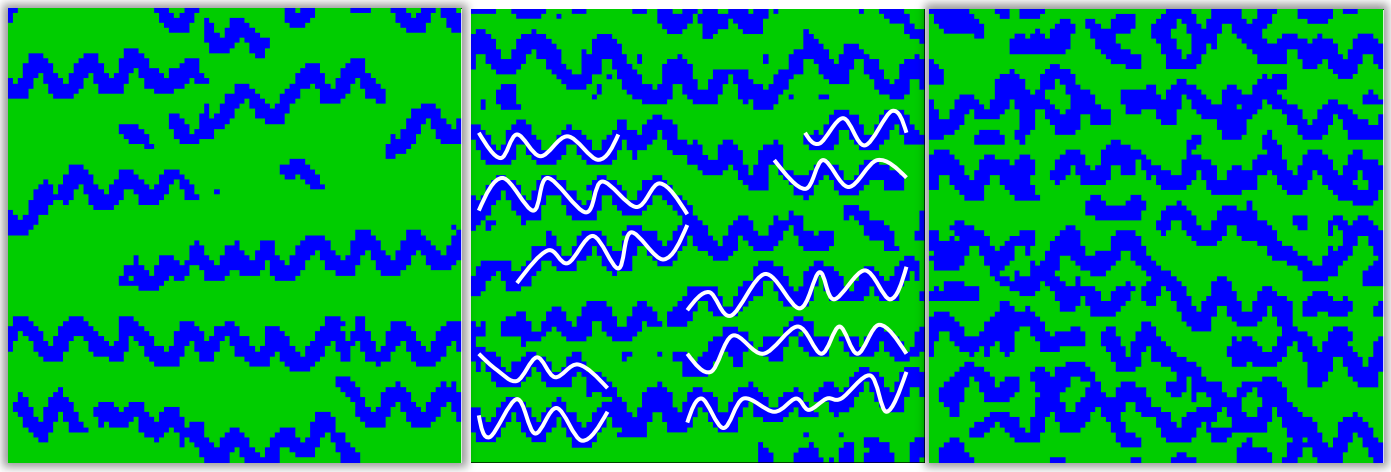


Figura 34. Modelos generados con IE de la figura 33. a) Generada con IE figura 33.a, b) generada con patrones regionales con IE figura 33.b, c) generada con patrones simples con IE figura 33.b.

Relación entre proporción de facies de interés en IE, facies observadas y modelos resultantes.

Como hablamos anteriormente, las relaciones entre las proporciones de facies en la IE y en el área a modelar tienden a mantenerse, siempre y cuando no existan facies observadas. Al incluir facies observadas sintéticas estas sesgan la proporción de facies en los modelos, las pruebas realizadas para evidenciar esta consecuencia, son las siguientes:

Se seleccionó la IE de la figura 7 para generar modelos en un área de 90x90 px, con el 5% de facies observadas distribuidas uniformemente en toda el área, luego se fue aumentando la relación entre facies de interés y de no-interés, en otras palabras, el muestreo de facies observadas siempre se mantuvo en un 5%, de este 5% se fue variando la cantidad de facies de interés, iniciando en 0% hasta un 100% aumentando constantemente 10% de probabilidad de ocurrencia de facies de interés en cada modelo.

En la figura 35 observamos los 11 modelos generados, en donde es evidente que al aumentar las cantidad de facies observadas de interés, aumenta consecuentemente la cantidad de facies de interés en el modelo, vale acotar que las facies aisladas se deben al hecho de que al ser facies generadas aleatoriamente esta no tienen correlación directa con la IE.

Para cuantificar objetivamente este efecto se creó el grafico de la figura 36, en donde en el eje horizontal se encuentra la cantidad de facies observada de interés simulada, y en el eje vertical la proporción encontrada en el modelo, igualmente se calculó cual es la proporción encontrada en la IE, en este caso fue aproximadamente de un 25% de facies de interés.

Es evidente que la curva tiene un crecimiento a medida que aumenta la cantidad de facies de interés.

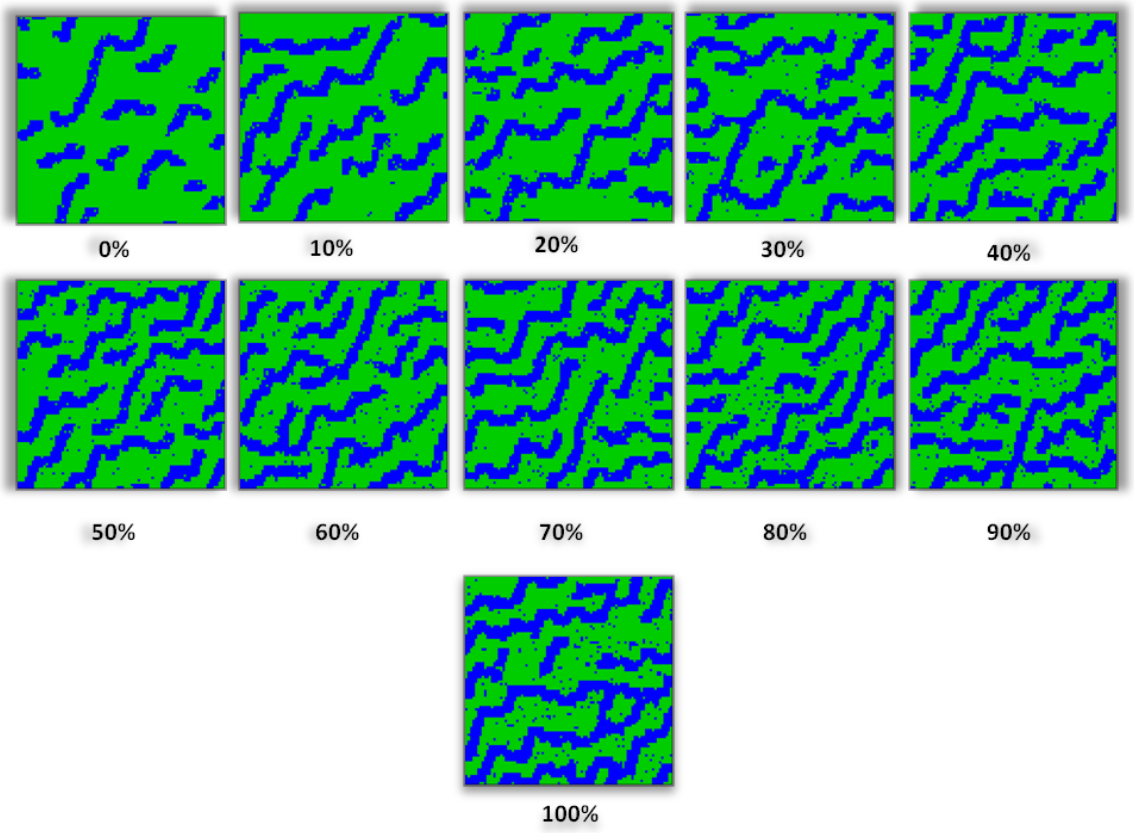


Figura 35. Modelos generados a partir de IE de la figura 7, con 5% de facies aleatorias observadas, aumentando la proporción de facies de interés.

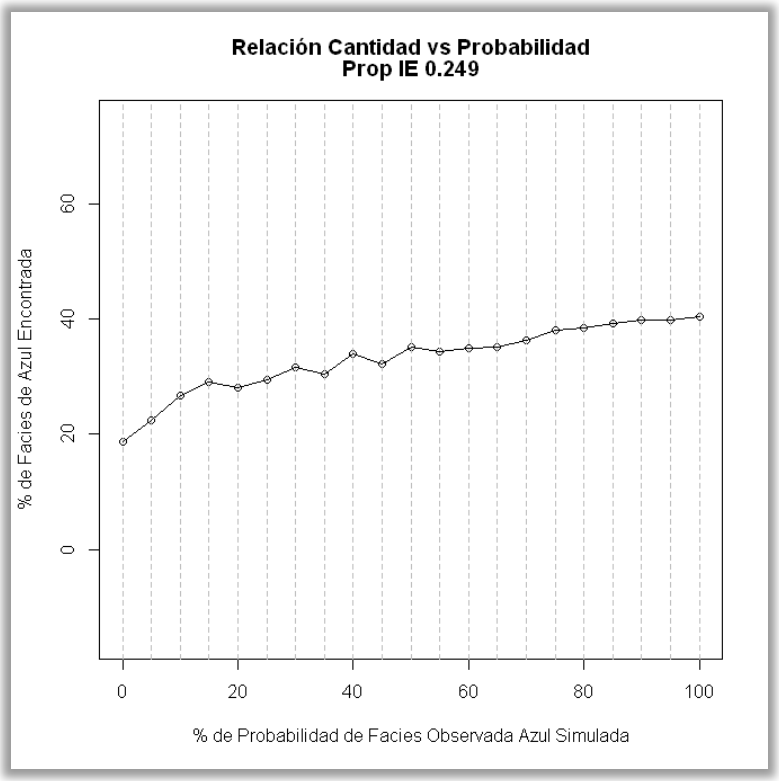


Figura 36. Relación de cantidad de facies simulada vs probabilidad de facies encontrada en modelo.

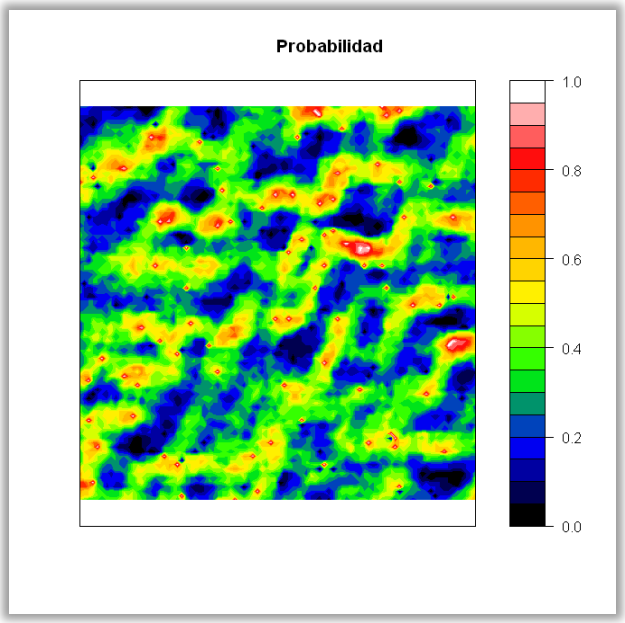
Al calcular la relación que existe entre las facies en la IE, esta nos arroja que existe cerca de un 25% de facies de interés, al observar la figura 36, podemos notar que al simular valores cercanos al 25% de facies de interés, obtenemos un modelo en donde la relación tiende a mantenerse. En otras palabras, al simular las facies con igual proporción a la que se tiene en la IE, obtenemos modelos que mantienen la misma relación.

Otra forma de entender este comportamiento, es entendiendo que al simular con la IE sin facies observadas, los modelos generados mantendrán la misma relación de facies de la IE, como vimos anteriormente lo que hace que esta relación se sesgue es la presencia de facies observadas.

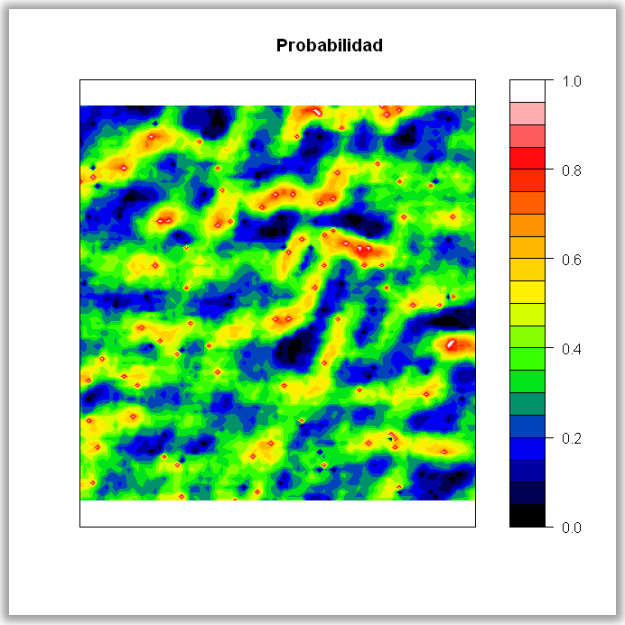
Cantidad optima de modelos a realizar.

Para logran obtener un mapa de probabilidad fiable es necesario generar una cantidad de modelos que garantice una menor varianza en los resultados, entonces ¿Cuál es el numero óptimo de modelos para obtener mayor certidumbre en los resultados? Según Jimba (2008), esta es una pregunta difícil de responder, ya que se debe conocer las características de los resultados que se desean obtener, él establece que el número de modelos necesario para obtener un mapa confiable es de 100 modelos.

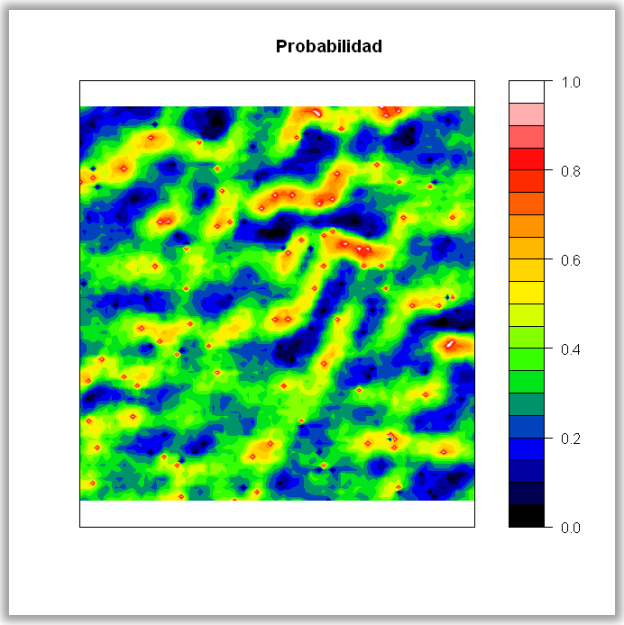
A continuación se muestran un conjunto de mapas de probabilidad (Figura 37) en donde se generaron 20, 40, 60, 80 y 100 modelos a partir de la IE de la figura 7, con el 2% de facies observadas, distribuidas uniformemente.



Mapa de probabilidad generado a partir de 20 modelos.

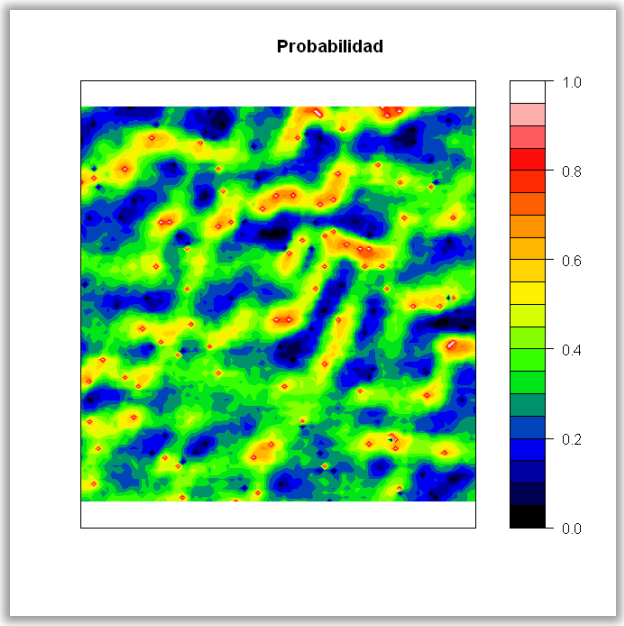


Mapa de probabilidad generado a partir de 40 modelos

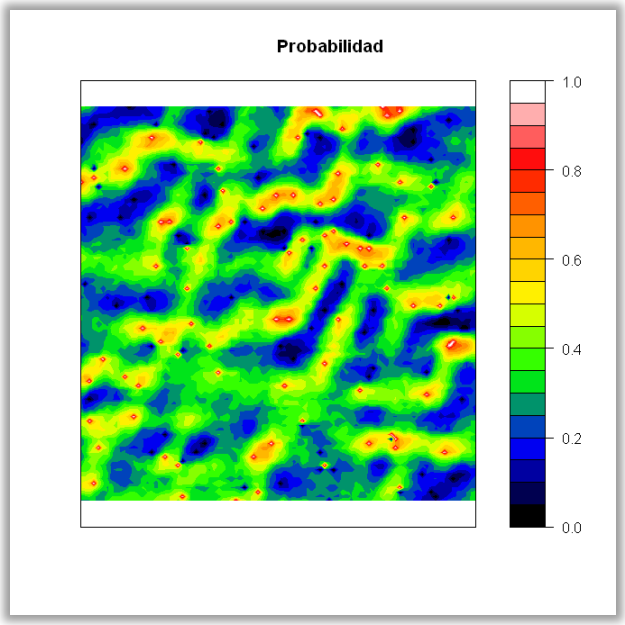


.

Mapa de probabilidad generado a partir de 60 modelos.



Mapa de probabilidad generado a partir de 80 modelos.



Mapa de probabilidad generado a partir de 100 modelos.

Figura 37. Mapas de probabilidad generados a partir de 20, 40, 60, 80 y 100 modelos.

Como podemos observar, en las imágenes de la figura 37, no existe una diferencia significativa en los mapas, esto se debe principalmente a que al tener facies observadas que sesguen la forma en que el algoritmo simula, estos mapas no tendrán mayores variaciones ya que están condicionadas en estas facies y no se pueden generar muchas más opciones de mapas.

Otra de las razones es debido a que al ser modelos sencillos, que solo contienen dos facies, estos modelos no tienen una complejidad tal que necesiten gran cantidad de modelos para poder obtener resultados que representen valores atípicos.

Conclusiones y recomendaciones.

Conclusiones generales:

a) Dado que la simulación multipunto logra capturar relaciones complejas entre varios puntos simultáneamente, se logran generar patrones geológicos complejos que se aproximan de mejor manera a los rasgos propios de los yacimientos.
b) La simulación multipunto combina información objetiva de los datos observados con información subjetiva suministrada por la IE, sin embargo, los modelos son generados a partir de la cuantificación y distribución estadística de la descomposición en patrones de esta imagen.

Conclusiones específicas:

a) Establecer el tamaño de los patrones mediante el método automático garantiza que la descomposición de la IE en patrones, estos representen de manera óptima lo que muestra la IE.
b) La IE debe ser consistente con las facies observadas, (en el caso de aplicar métodos de simulación, de igual manera las facies generadas deben ser consistentes con la IE), de lo contrario se obtendrán valores aislados e incoherencias en los modelos.
c) Las proporciones entre la facies de interés y de no-interés de la IE y de las facies observadas, deben ser similares. Al tener proporciones similares garantizamos que los patrones obtenidos de la IE puedan generar efectivamente conexiones coherentes entre las facies observadas.
d) Es recomendable utilizar el método de corrección de modelo (limpieza), ya que reacomoda la distribución de los patrones dando como resultado un modelo más consistente.

e) Aplicando la plantilla regional logramos capturar mejor la distribución espacial de la IE y representar de mejor forma lo que se ve en los modelos, sin embargo esta solo es necesaria cuando en la IE la distribución de las facies y formas geológicas tengan una configuración espacial que un patrón simple no sea capaz de capturar.
f) Al generarse modelos simples (dos facies), no es necesario generar gran cantidad de modelos para obtener resultados significativos.
g) Se recomienda extender el método a múltiples facies y 3d teniendo en cuenta que es necesario tener mayor capacidad de cómputo.

Bibliografía

Arbat, B. (2005). *Sequential simulation with patterns.* Standford University.

Azpurua, M., & Dos Ramos, K. (2010). *A comparison of spatial interpolation methods for estimation of average electromagnetic field magnitud.* Instituto de Ingeniería, Caracas.

Caers, J., & Srinivansan, S. (2001). *Combining geological information with seismic and production data.* Oxford: Elsevier Publishers.

Davis, J. (1986). *Statistics and Data Analysis in Geology* (Segunda ed.). New York: John Wiley & Sons.

Deutsch, C. (2002). *Geostatistical Reservoir Modeling.* Oxford: Oxford University Press.

Dubrule, O. (2003). *Geostatistics for Seismic Data Integration.*

Jimba, O. (2008). *Basic Geostatistics.*

Matheron, G. (2008). *Las variables regionalizadas y su distribución.*

Obregon, N., & Fragala, F. (2002). Predicción de caudales medios mensuales en la estación La Pradera mediante RNA. *Ponencia presentada en Memorias del II Congreso Colombiano y el encuentro Andino de Investigación de Operaciones.* Bogotá.

Olea,R. (1991). *Geostatistical Glossary and Multilingual Dictionary.* New York: Oxford University Press.

Sobol, I. M. (1983). *Método de Montecarlo.* Moscú: MIR.

Spiegel, M. R. (1969). *Estadística.* Libros McGraw-Hill.

Strebelle, S. (2002). *Conditional simulation of complex geological structures using multipoint statistics.* Mathematical Geology.

ANEXOS

Código fuente del algoritmo.

Lenguaje de programación: R

```
{

graphics.off()

setWindowTitle(title="Simulación Multipunto")

windows.options(restoreConsole=T)

## Librerias

if(!any(rownames(installed.packages())=="png"))

suppressWarnings(suppressMessages(install.packages("png",repos
="http://cran.rstudio.com")))

suppressWarnings(library(png))

if(!any(rownames(installed.packages())=="gWidgets"))

{

suppressWarnings(suppressMessages(install.packages("gWidgets",repos
="http://cran.rstudio.com")))

suppressWarnings(suppressMessages(install.packages("gWidgetsRGtk2",r
epos ="http://cran.rstudio.com")))

}

suppressWarnings(suppressMessages(library(gWidgets)))

TrainImage<-function(x)

{

IE<-abs(1-floor(x[,,1]/max(x[,,1])))

return(IE)

}

Corrimage<-function(x,c1=3,c2=4,lineas=T,plt=T,marg=T)

{

if(any(is.na(unique(as.vector(x))))&length(unique(as.vector(x))))){
```

```
c1<-4

c2<-3

}

if(plt==T)

x11()

c3<-c(c1,c2)

if(all(x==0,na.rm=T))

c3<-rev(c3)

x_corr<-t(apply(x,2,function(x) x[length(x):1]))

if(marg==T)

par(mai=c(0,0,0,0))

image(0:length(x[1,]),0:length(x[,1]),x_corr,col=c3,xaxt="n",yaxt="n
",xlab="",ylab="",asp=1)

if(lineas==T)

{

d1<-0:length(x[1,])

d2<-0:length(x[,1])

abline(h=d2,v=d1)

}

}

Patterns<-function(x,dimen=3)

{

patt<-list()

k<-0

for(i in1:(length(x[,1])-(dimen-1)))

{

for(j in1:(length(x[1,])-(dimen-1)))

{

k<-k+1

patt[[k]]<-x[i:(i+(dimen-1)),j:(j+(dimen-1))]
```

```
}

}

return(patt)

}

Patt_unique<-function(x)

{

patrnames<-
unlist(lapply((lapply(x,as.vector)),function(x)paste(x,collapse=""))
)

names(x)<-patrnames

patrnames<-unique(patrnames)

r<-x[patrnames]

return(r)

}

Probpat<-function(patron)

{

patrnames<-
unlist(lapply((lapply(patron,as.vector)),function(x)paste(x,collapse
=""))))

prob.patr<-t(t((table(patrnames))))

prob.patr<-cbind(prob.patr,prob.patr/sum(prob.patr))

colnames(prob.patr)<-c("Rep","Prob")

return(prob.patr)

}

Similar<-function(x,y,versimilares=F,similarexacto=F)

{

x<-Patt_unique(x)

acomp<-as.vector(y)

r<-
sort(unlist(lapply(lapply(lapply(x,as.vector),function(x)x==acomp),f
unction(x)sum(x,na.rm=T))),decreasing=T)

r1<-r[r==max(r)]
```

```
r2<-x[names(r1)]
probab<-Probpat(x)[,2][names(r1)]
if(versimilares==F)
{
r3<-sample(r2,1,prob=probab)[[1]]
return(r3)
}
if(versimilares!=F)
return(r2)
if(similarexacto==T)
{
r4<-lapply(r2,function(x)all(x==y,na.rm=T))
if(any(unlist(r4)))
return(r3)
else
return(y)
}
}
Patimage<-function(x,y,plt=T,lineas=F,ret=F,c1=3,c2=4)
{
h<-1
s<-NULL
for(j in1:(length(x[,1])-(length(y[[1]][,1])-1)))
{
w<-y[[h]]
for(i in(1+h):(length(x[1,])-(length(y[[1]][,1]))+h))
{
w<-cbind(w,y[[i]])
h<-h+1
```

```
}
s<-rbind(s,w)
h<-h+1
}
Corrimage(s,lineas=F,c1=c1,c2=c2,plt=T)
if(lineas==T)
{
d1<-seq(0,length(s[1,]),nrow(patrones[[1]]))
d2<-seq(0,length(s[,1]),ncol(patrones[[1]]))
abline(h=d2,v=d1)
}
if(ret==T)
return(s)
}
Box<-function(x)
{
r<-rep(NA,x^2)
dim(r)<-c(x,x)
return(r)
}
Probimage<-function(IP,suav=1)
{
ipaux<-IP
if(suav!=0)
{
colNA<-rep(NA,dim(ipaux)[1])
for(r in1:suav)
ipaux<-cbind(colNA,ipaux,colNA)
filaNA<-rep(NA,dim(ipaux)[2])
```

```
for(r in1:suav)

ipaux<-rbind(filaNA,ipaux,filaNA)

}

x<-IP

for(i in1:(dim(IP)[1]))

{

for(j in1:(dim(IP)[2]))

{

x[i,j]<-
mean(ipaux[c(i:(i+2*(suav))),c(j:(j+2*(suav)))],na.rm=T)/ifelse(suav
==0,1,suav)

}

}

x_corr<-t(apply(x,2,function(x) x[length(x):1]))

par(mai=c(0,0,0,0))

image(0:length(x[1,]),0:length(x[,1]),col=rev(heat.colors(length(uni
que(as.vector(x))))),x_corr,xaxt="n",yaxt="n",xlab="",ylab="",asp=1)

return(x)

}

ObsFac<-function(fac_obs,caj,prob1=.5,probimagen=NA)

{

pos<-expand.grid(1:dim(caj)[1],1:dim(caj)[2])

pos2<-sample(dim(pos)[1],fac_obs)

probabilidad<-c(prob1,(1-prob1))

posx<-pos[pos2,1]

posy<-pos[pos2,2]

for(i in1:length(posx))

{

if(all(!is.na(probimagen)))

{

if(all(dim(probimagen)!=dim(caj)))
```

```
{

stop("Dimensiones diferentes de caja e imagen de
probabilidad",call.=F)

}

probabilidad<-c(probimagen[posx[i],posy[i]],1-
probimagen[posx[i],posy[i]])

}

caj[posx[i],posy[i]]<-sample(c(1,0),1,T,prob=probabilidad)

}

return(corrimage(caj))

}

Fill<-function(caj,patrones,lin=F,plt=F)

{

t1<-Sys.time()

caj2<-caj

for(i in1:(ncol(patrones[[1]])))

caj<-cbind(caj,rep(NA,nrow(caj)))

for(i in1:(ncol(patrones[[1]])))

caj<-cbind(rep(NA,nrow(caj)),caj)

for(i in1:(nrow(patrones[[1]])))

caj<-rbind(caj,rep(NA,ncol(caj)))

for(i in1:(nrow(patrones[[1]])))

caj<-rbind(rep(NA,ncol(caj)),caj)

patrunic<-Patt_unique(patrones)

x<-1:(length(caj[1,])-((length(patrunic[[1]][1,])-1)))

y<-1:(length(caj[,1])-((length(patrunic[[1]][,1])-1)))

z<-expand.grid(x,y)

possample<-sample(nrow(z))

z<-z[possample,]

w<-0
```

```
aa<-0

bb<-0

patpro<-Probpat(patrones)

pb <- winProgressBar(title="Progreso", label="0%", min=0, max=100,
initial=0,width =450)

noigual<-0

while(sum_na<-sum(is.na(caj))!=0)

{

w<-w+1

nas<-floor(100*(sum(!is.na(caj))/(dim(caj)[1]*dim(caj)[2])))

nas2<-100*(sum(!is.na(caj))/(dim(caj)[1]*dim(caj)[2]))

info <- paste0(nas,"%","  ","I",w,"  ","V",aa,"  ","S",bb,"
","NA",sum(is.na(caj)),"  ","E",noigual)

setWinProgressBar(pb, nas2, label=info,title=paste0(nas,"%","
Completado"))

posx<-z[w,1]

posy<-z[w,2]

inbox<-caj[posx:(posx+(length(patrunic[[1]][,1])-
1)),posy:(posy+(length(patrunic[[1]][1,])-1))]

pat<-sample(names(patrunic),1,prob=patpro[,2])

if(all(is.na(inbox)))

{

aa<-aa+1

for(i in0:(length(patrunic[[1]][1,])-1))

{

for(j in0:(length(patrunic[[1]][,1])-1))

{

caj[posx+i,posy+j]<-patrunic[[pat]][i+1,j+1]

}

}

}
```

```
if(!all(is.na(inbox))&any(is.na(inbox)))

{

bb<-bb+1

patsim<-Similar(patrones,inbox)

if(any(inbox!=patsim,na.rm=T))

noigual<-noigual+1

for(i in0:(length(patrunic[[1]][1,])-1))

{

for(j in0:(length(patrunic[[1]][,1])-1))

{

if(is.na(caj[posx+i,posy+j]))

caj[posx+i,posy+j]<-patsim[i+1,j+1]

}

}

}

}

noigual<<-noigual

caj<-caj[(ncol(patrones[[1]])+1):(ncol(caj)-
ncol(patrones[[1]])),(ncol(patrones[[1]])+1):(ncol(caj)-
ncol(patrones[[1]]))]

close(pb)

if(plt)

Corrimage(caj,lineas=lin,plt=plt)

t2<-Sys.time()

tiempohour<-as.numeric(difftime(t2,t1,units="hours"))

tiempomin<-as.numeric(difftime(t2,t1,units="mins"))

tiemposec<<-as.numeric(difftime(t2,t1,units="secs"))

message("\n",paste0("Tiempo","\n",round(tiempohour,2)," ","Horas","
=  ",round(tiempomin,2)," ","Minutos","  =  ",round(tiemposec,2),"
","Segundos"))

flush.console()
```

```
return(caj)
}
Patterns2<-function(x,lado=3,sep=1)
{
patt<-list()
k<-0
for(i in1:(length(x[,1])-((lado-1)*sep+lado-1)))
{
for(j in1:(length(x[1,])-((lado-1)*sep+lado-1)))
{
k<-k+1
patt[[k]]<-matrix(rep(NA,lado*lado),ncol=lado)
S<-seq(1,(lado-1)*sep+lado,(sep+1))
patt[[k]]<-x[S+i-1,S+j-1]
}
}
return(patt)
}
Matsep<-function(x,caj,sep=1)
{
y<-rep(NA,(dim(x)[1]*dim(x)[2])*4)
dim(y)<-c(dim(x)[1]*2,dim(x)[2]*2)
a<-seq(1,ncol(y),(sep+1))
b<-seq(1,nrow(y),(sep+1))
nacol<-rep(NA,nrow(x))
narow<-rep(NA,ncol(x))
for(i in1:length(a))
for(j in1:length(b))
y[b[j],a[i]]<-x[j,i]
```

```
if(all(dim(y)!=dim(caj)))
y<-y[-nrow(y),-ncol(y)]
return(y)
}

AutoPatterns<-function(IE)
{
for(i in2:dim(IE)[1])
{
p<-Patterns(IE,i)
pu<-Patt_unique(p)
if(length(p)==length(pu))
{
templete<-i-1
p<-Patterns(IE,templete)
pu<-Patt_unique(p)
break()
}
}
patrones<-p
return(patrones)
}
Resize<-function(IE,lado)
{
side<-lado
if(side<20)
{
}
else
```

```
{
png("IEreesc.png",side+2,side+2)
Corrimage(IE,lineas=F,plt=F,c2="black",c1="white")
dev.off()
x2<-readPNG("IEreesc.png")
file.remove("IEreesc.png")
IE2<-TrainImage(x2)
IE2<-IE2[-dim(IE2)[1],-dim(IE2)[2]]
IE2<-IE2[-1,-1]
return(IE2)
}
}
jet.colors <-
  colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan","#7FFF7F",
"yellow", "#FF7F00", "red", "#7F0000"))
rainbow2 <-
  colorRampPalette(rainbow(10,end=.85))
xcel<-
      colorRampPalette(rev(c("white","red","orange","yellow","green",
"blue","black")))
bw<-
      colorRampPalette(rev(c("black","white")))
Filled2<-
function(z,escala=T,color=1,lim=NA,marg=F,tituloescala="Escala",inte
grado=FALSE)
{
    x <-(1:nrow(z))-.5
      y <-(1:ncol(z))-.5
      xlim <-range(x, finite =TRUE)
      ylim <-range(y, finite =TRUE)
      zlim <-range(z, finite =TRUE)
```

```
      nlevels=20
      if(any(!is.na(lim)))
      zlim <- lim
      levels=pretty(zlim, nlevels)
color.palette =c(xcel,
# mat.colors,
terrain.colors,heat.colors,rainbow2,jet.colors,topo.colors,cm.colors
,bw)
      col2 = color.palette[[color]](length(levels)-1)
if(integrado)
filled.contour(z,col=col2,plot.axes=FALSE,asp=1)
if(!integrado)
{
      if(escala==T)
      {
      plot.new()
par(mar=c(3,10,3,10))
      plot.window(xlim =c(0, 1), ylim =range(levels), xaxs ="i",
    yaxs ="i")
      rect(0, levels[-length(levels)], 1, levels[-1L], col= col2)
      title(tituloescala)
axis(4)
      x11()
      if(marg)
      par(mai=c(0,0,0,0))
}
plot.new()
plot.window(xlim, ylim, xaxs ="i", yaxs ="i", asp =1)
if(!is.double(z))
        storage.mode(z)<-"double"
```

```
    .Internal(filledcontour(as.double(x), as.double(y), z,
as.double(levels),

col= col2))

}

}

Marc<-
function(cajfac,marcar=F,puntos=T,tam=1,color=c(1,2),grues=1,forma1=
1,forma2=4)

{

corrcajfac<-corrimage(cajfac)

int<-which(corrcajfac==1,arr.ind=T)

nint<-which(corrcajfac==0,arr.ind=T)

if(puntos==T)

{

points(int-.5,pch=forma1,cex=tam,,col=color[1])

points(nint-.5,pch=forma2,cex=tam,col=color[2])

}

if(marcar==T)

{

for(i in1:nrow(int))

{

x<-int[i,1]-1

y<-int[i,2]-1

x0<-c(0,0,1,0)+x

y0<-c(0,1,1,0)+y

x1<-c(0,1,1,1)+x

y1<-c(1,1,0,0)+y

segments(x0,y0,x1,y1,lwd=grues,col=color[1])

}

for(i in1:nrow(nint))

{
```

```
x<-nint[i,1]-1
y<-nint[i,2]-1
x0<-c(0,0,1,0)+x
y0<-c(0,1,1,0)+y
x1<-c(0,1,1,1)+x
y1<-c(1,1,0,0)+y
segments(x0,y0,x1,y1,lty=3,lwd=grues,col=color[2])
}
}
}
CorrBox<-function(caj,sep=1)
{
a<-seq(1,ncol(caj),(sep+1))
b<-seq(1,nrow(caj),(sep+1))
caj<-caj[a,b]
return(caj)
}
CoarseFill<-function(caj,patrones,facobs=NA,plt=F)
{
if(length(facobs)==1)
facobs<-matrix(rep(NA,nrow(caj)^2),nrow=nrow(caj))
if(all(dim(caj)!=dim(facobs)))stop("Problema de dimensiones")
t1<-Sys.time()
caj2<-caj
for(i in1:(ncol(patrones[[1]])))
caj<-cbind(caj,rep(NA,nrow(caj)))
for(i in1:(ncol(patrones[[1]])))
caj<-cbind(rep(NA,nrow(caj)),caj)
for(i in1:(nrow(patrones[[1]])))
```

```
caj<-rbind(caj,rep(NA,ncol(caj)))

for(i in1:(nrow(patrones[[1]])))

caj<-rbind(rep(NA,ncol(caj)),caj)

facobs2<-facobs

for(i in1:(ncol(patrones[[1]])))

facobs<-cbind(facobs,rep(NA,nrow(facobs)))

for(i in1:(ncol(patrones[[1]])))

facobs<-cbind(rep(NA,nrow(facobs)),facobs)

for(i in1:(nrow(patrones[[1]])))

facobs<-rbind(facobs,rep(NA,ncol(facobs)))

for(i in1:(nrow(patrones[[1]])))

facobs<-rbind(rep(NA,ncol(facobs)),facobs)

patrunic<-Patt_unique(patrones)

x<-1:(length(caj[1,])-((length(patrunic[[1]][1,])-1)))

y<-1:(length(caj[,1])-((length(patrunic[[1]][,1])-1)))

z<-expand.grid(x,y)

possample<-sample(nrow(z))

z<-z[possample,]

caj[which(!is.na(facobs),arr=T)]<-
facobs[which(!is.na(facobs),arr=T)]

w<-0

aa<-0

bb<-0

patpro<-Probpat(patrones)

pb <- winProgressBar(title="Progreso", label="0%", min=0, max=100,
initial=0,width =450)

noigual<-0

while(sum(is.na(caj))!=0)

{

w<-w+1
```

```
nas<-floor(100*(sum(!is.na(caj))/(dim(caj)[1]*dim(caj)[2])))

nas2<-100*(sum(!is.na(caj))/(dim(caj)[1]*dim(caj)[2]))

info <- paste0(nas,"%","  ","I",w,"  ","V",aa,"  ","S",bb,"
","NA",sum(is.na(caj)),"  ","E",noigual)

setWinProgressBar(pb, nas2, label=info,title=paste0(nas,"%","
Completado"))

posx<-z[w,1]

posy<-z[w,2]

inbox<-caj[posx:(posx+(length(patrunic[[1]][,1])-
1)),posy:(posy+(length(patrunic[[1]][1,])-1))]

pat<-sample(names(patrunic),1,prob=patpro[,2])


if(all(is.na(inbox)))

{

aa<-aa+1

for(i in0:(length(patrunic[[1]][1,])-1))

{

for(j in0:(length(patrunic[[1]][,1])-1))

{

caj[posx+i,posy+j]<-patrunic[[pat]][i+1,j+1]

}

}

}

if(!all(is.na(inbox))&any(is.na(inbox)))

{

bb<-bb+1

patsim<-Similar(patrones,inbox)

if(any(inbox!=patsim,na.rm=T))

noigual<-noigual+1

for(i in0:(length(patrunic[[1]][1,])-1))

{
```

```
for(j in0:(length(patrunic[[1]][,1])-1))

{

if(is.na(facobs[posx+i,posy+j]))

caj[posx+i,posy+j]<-patsim[i+1,j+1]


}

}

}

}

noigual<<-noigual

a<-ncol(caj)

b<-ncol(patrones[[1]])

caj<-caj[(b+1):(a-b),(b+1):(a-b)]

close(pb)

if(plt)

Corrimage(caj,lineas=F,plt=plt)

t2<-Sys.time()

tiempohour<-as.numeric(difftime(t2,t1,units="hours"))

tiempomin<-as.numeric(difftime(t2,t1,units="mins"))

tiemposec<<-as.numeric(difftime(t2,t1,units="secs"))

message("\n",paste0("Tiempo","\n",round(tiempohour,2)," ","Horas","
=  ",round(tiempomin,2)," ","Minutos","  =  ",round(tiemposec,2),"
","Segundos"))

flush.console()

return(caj)

}

Clean<-function(model,patrones,facobs="",plt=F)

{

if(suppressWarnings(suppressMessages(class(facobs)=="character")))

facobs<-matrix(rep(NA,nrow(model)^2),nrow=nrow(model))
```

```
t1<-Sys.time()

pb <- winProgressBar(title="Progreso", label="0%", min=0, max=100,
initial=0,width =450)

caj<-model

patrunic<-Patt_unique(patrones)

x<-1:(length(caj[1,])-((length(patrunic[[1]][1,])-1)))

y<-1:(length(caj[,1])-((length(patrunic[[1]][,1])-1)))

z<-expand.grid(x,y)

possample<-sample(nrow(z))

z<-z[possample,]

noigual<-0

for(w in1:length(z[,1]))

{

aa<-round(w/length(z[,1])*100,2)

info <- paste0(aa,"%","  ","E",noigual)

setWinProgressBar(pb, aa, label=info,title=paste0(aa,"%","
Completado (Corrección)"))

posx<-z[w,1]

posy<-z[w,2]

inbox<-caj[posx:(posx+(length(patrunic[[1]][,1])-
1)),posy:(posy+(length(patrunic[[1]][1,])-1))]

patsim<-Similar(patrones,inbox)

if(any(inbox!=patsim,na.rm=T))

noigual<-noigual+1

for(i in0:(length(patrunic[[1]][1,])-1))

{

for(j in0:(length(patrunic[[1]][,1])-1))

{

if(is.na(facobs[posx+i,posy+j]))

caj[posx+i,posy+j]<-patsim[i+1,j+1]

}
```

```
}

}

noigual<<-noigual

close(pb)

if(plt)

Corrimage(caj,lineas=F,plt=T)

t2<-Sys.time()

tiempohour<-as.numeric(difftime(t2,t1,units="hours"))

tiempomin<-as.numeric(difftime(t2,t1,units="mins"))

tiemposec<<-as.numeric(difftime(t2,t1,units="secs"))

message("\n",paste0("Tiempo","\n",round(tiempohour,2)," ","Horas","
=  ",round(tiempomin,2)," ","Minutos","  =  ",round(tiemposec,2),"
","Segundos"))

flush.console()

return(caj)

}

Moda<-function(k)

{

r1<-table(k)[table(k)==max(table(k))]

r2<-data.frame(N=as.numeric(names(r1)),R=as.numeric(r1))

return(r2)

}

ModelMode<-function(listamodelos)

{

mod<-
matrix(rep(NA,dim(listamodelos[[1]])[1]*dim(listamodelos[[1]])[2]),n
col=dim(listamodelos[[1]])[2])

for(i in1:dim(listamodelos[[1]])[1])

{

for(j in1:dim(listamodelos[[1]])[2])

{
```

```
mod[i,j]<-sample(Moda(unlist(lapply(listamodelos,function(y)
y[i,j])))[,1],1)

}

}

return(mod)

}

corrimage<-function(x)

{

r<-t(apply(x,2,function(x) x[length(x):1]))

return(r)

}

ModelProb<-function(listamodelos)

{

mod<-listamodelos

colum<-ncol(mod[[1]])

modsum<-matrix(rep(0,nrow(mod[[1]])*colum),ncol=colum)

for(i in1:length(listamodelos))

{

modsum<-modsum+mod[[i]]

}

modprob<-modsum/length(listamodelos)

r<-corrimage(modprob)

return(r)

}

MPV<-function(listamodelo)

{

modmodel<-ModelMode(listamodelo)

probmodel<-ModelProb(listamodelo)

varmodel<-probmodel*(1-probmodel)

r<-
```

```
list("Moda"=modmodel,"Probabilidad"=probmodel,"Varianza"=varmodel)

return(r)

}

MPVmaps<-function(MPV,escala=T,tipo=c(1,2,3),integ=FALSE)

{

if(any(tipo==1))

{

Corrimage(MPV$Moda,marg=F,lineas=F)

title("Moda")

}

if(any(tipo==2))

{

x11()

Filled2(MPV$Probabilidad,escala,color=1,lim=c(0,1),tituloescala="Esc
ala Probabilidad",integrado=integ)

title("Probabilidad")

}

if(any(tipo==3))

{

x11()

Filled2(MPV$Varianza,escala,color=5,lim=c(0,.25),tituloescala="Escal
a Varianza",integrado=integ)

title(main=c("Varianza",paste("\n","Varianza
Total",round(VarTotal(MPV$Varianza),3))))

}

}

Distance<-function(ima,dista=1)

{

m<-as.matrix(expand.grid(1:dim(ima)[1],1:dim(ima)[2]))

Dis<-list()

for(k in0:dista)
```

```
{
distancia<-k
x<-list()
for(i in1:nrow(m))
{
pos1<-m[i,]
difr<-abs(m[,1]-pos1[1])
posf<-m[difr<=distancia,]
difc<-abs(posf[,2]-pos1[2])
dis<-posf[difc<=distancia,]
x[[i]]<-dis
}
Dis[[k+1]]<-x
}
return(Dis)
}
RealDist<-function(listDist)
{
R<-listDist
zpos<-list()
zpos2<-list()
zpos3<-list()
for(j in1:(length(R)-1))
{
for(i in1:length(R[[1]]))
{
if(j==1)
{
zpos[[i]]<-which(R[[j+1]][[i]][,1]%in% R[[j]][[i]][1]&
R[[j+1]][[i]][,2]%in% R[[j]][[i]][2])
```

```
zpos2[[i]]<-R[[j+1]][[i]][-zpos[[i]],]

}

if(j>1)

{

zpos[[i]]<-which(R[[j+1]][[i]][, 1]%in% R[[j]][[i]][, 1]&
R[[j+1]][[i]][, 2]%in% R[[j]][[i]][, 2])

zpos2[[i]]<-R[[j+1]][[i]][-zpos[[i]],]

}

}

zpos3[[j+1]]<-zpos2

}

zpos3[[1]]<-R[[1]]

return(zpos3)

}

InvDist<-function(ima,dista=1,expo=1,plt=F)

{

IPinv<-ima

if(dista>0)

{

Dis<-Distance(ima,dista)

Rdis<-RealDist(Dis)

pesos<-(1/(1:length(Rdis)^expo))

sumapesos<-sum(pesos)

for(k in2:length(Rdis))

{

for(j in1:length(Rdis[[k]]))

{

pos<-Rdis[[k]][[j]]

IPinv[Rdis[[1]][[j]][1],Rdis[[1]][[j]][2]]<-
sum(ima[pos]*pesos[k])/sumapesos
```

```
}

}

}

IPinv2<-IPinv/max(IPinv)

IPinv3<-IPinv2[(dista+1):(ncol(IPinv2)-
dista),(dista+1):(ncol(IPinv2)-dista)]

if(plt)

{

x11()

par(mai=par()$mai)

filled.contour(corrimage(IPinv3),color.palette=xcel,asp=1,plot.axes=
F)

}

probim<-IPinv3

probim<-apply(probim,1,function(x)ifelse(x==max(probim),.975,x))

probim<-apply(probim,1,function(x)ifelse(x==min(probim),.025,x))

return(corrimage(probim))

}

ToCoord<-
function(mtx,limsur=0,limnorte=1,limoeste=0,forma=1,color=4,rotar=0)

{

mtx<-corrimage(mtx)

prop<-matrix(as.vector(mtx),ncol=1)

limx<-c(limoeste,(limnorte-limsur)+limoeste)

limy<-c(limsur,limnorte)

loutx<-ncol(mtx)

louty<-nrow(mtx)

posx<-seq(limx[1],limx[2],length.out=loutx)

posy<-seq(limy[1],limy[2],length.out=louty)


w<-0
```

```
for(j in1:ncol(mtx))
{
for(i in1:nrow(mtx))
{
w<-w+1
prop[w,]<-mtx[i,j]
}
}
coord<-as.matrix(expand.grid(posx,posy))
if(rotar!=0)
{
alfa<-(pi/180)*rotar
x1<-coord[,1]*cos(alfa)+coord[,2]*sin(alfa)
y1<--coord[,1]*sin(alfa)+coord[,2]*cos(alfa)
coord[,1]<-x1
coord[,2]<-y1
}
coord2<-cbind(coord,prop)
return(coord2)
}
VarTotal<-function(varmap)
{
v<-as.vector(varmap)
r<-sum(v)/(.25*length(v))
return(r)
}
Alpha<-function(color,alph=.5)
{
x<-col2rgb(color)
```

```
x2<-rgb(x[1],x[2],x[3],max=255,alpha=255*alph)

return(x2)

}

FacToBox<-function(caja,facies)

{

for(j in1:ncol(caja))

for(i in1:nrow(caja))

if(is.na(caja[i,j]))

caja[i,j]<-facies[i,j]

return(caja)

}

Kriging<-function(mtx,plt=F)

{

if(!any(rownames(installed.packages())=="kriging"))

suppressWarnings(suppressMessages(install.packages("kriging",repos
="http://cran.rstudio.com")))

if(!any("Kriging"%in%ls()))

suppressWarnings(suppressMessages(library(kriging)))

a<-ToCoord(mtx)

b<-a[!is.na(a[,3]),]

x<-b[,1]*nrow(mtx)

y<-b[,2]*nrow(mtx)

z<-b[,3]

k1<-kriging(x,y,z,pixels=(nrow(mtx)+1),model="spherical")

k2<-ToMat(k1$map)

if(plt)

Filled2(k2,marg=T)


return(k2)

}
```

```
ToMat<-function(coord)
{
x<-length(unique(coord[,1]))
y<-length(unique(coord[,2]))
r<-matrix(rep(NA,x*y),ncol=y)
w<-0
for(i in1:x)
for(j in1:y)
{
w<-w+1
r[i,j]<-coord[w,3]
}
return(r)
}
Espere<-function(mens="Iniciando\n Espere...",n=25)
{
x11(width=3,height=2,title="Espere...",xpos=-1,ypos=0)
par(mar=rep(0,4),bg=8)
n<-n
image(0:n,0:n,matrix(rep(1,n*n),ncol=n),col=sample(c(3,4),1))
text(n/2,n/2,paste(mens),cex=3,col=1,font=3)
}
csvlist<-function(lista,nombre)
{
r<-lista[[1]]
for(i in2:length(lista))
r<-rbind(r,rep(NA,nrow(lista[[1]])),lista[[i]])
write.table(r,nombre,row.names=F,col.names=F,sep=",")
}}
```