UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE POSGRADO EN CIENCIAS DE LA TIERRA

ENSAMBLE SUAVIZADO IMPLEMENTADO EN PARALELO CON APLICACIONES A PROBLEMAS DE AGUA SUBTERRÁNEA

T E S I S

QUE COMO REQUISITO PARCIAL PARA OPTAR POR EL GRADO DE:
D O C T O R E N C I E N C I A S D E L A T I E R R A

P R E S E N T A

ESTHER LEYVA SUÁREZ

DIRECTOR DE TESIS:

Dra. Graciela del Socorro Herrera Zamarrón
Instituto de Geofísica

COMITÉ TUTOR:

Dr. Luis Miguel de la Cruz Salas
Instituto de Geofísica

Dr. Martín Díaz Viera
Instituto Mexicano del Petróleo

CIUDAD DE MÉXICO ABRIL DE 2019.

Dedicatoria

Principalmente a Dios, por darme la oportunidad de vivir esta hermosa experiencia.

A mis padres, hermanos y sobrinos, que me brindaron en todo momento su apoyo incondicional y cariño.

Agradecimientos

Agradezco a la Universidad Nacional Autónoma de México por brindarme la oportunidad de estar en uno de sus programas doctorales, al Consejo Nacional de Ciencia y Tecnología quien me otorgó la beca para realizar mis estudios de doctorado en la UNAM.

También quisiera agradecer a Supercómputo de la UNAM, que me brindó apoyo en el proyecto semilla LANCAD-UNAM-DGTIC-301 con código 5C15-1-S-68.

Asimismo, agradezco a la Dra. Graciela Herrera Zamarrón, directora de esta tesis, y al Dr. Luis Miguel de la Cruz Salas, por compartir conmigo sus conocimientos, sabiduría y brindarme su apoyo durante la realización de este trabajo. A los sinodales: Dr. Jorge Zavala Hidalgo, Dr. Eric Morales Casique y Dr. Demetrio Fabián García Nocetti, que revisaron mi tesis y brindaron una buena aportación.

Del mismo modo quisiera expresar mi agradecimiento a Leobardo Itehua, Gustavo Ramos, Antonio Carrillo y John Jairo Díaz, quienes me brindaron su ayuda en la parte de informática, así como su valiosa amistad.

Resumen

La asimilación de datos es un proceso que utiliza modelos de predicción y mediciones como fuentes de información para mejorar los resultados de dichos modelos. El método secuencial de asimilación de datos llamado filtro de Kalman ensamblado (*EnKF* por sus siglas en inglés), se diseñó para resolver dos de las mayores dificultades relacionadas con el uso del filtro de Kalman extendido (*EKF* por sus siglas en inglés) en problemas con dinámica no lineal en espacios grandes de estados, estos son, el uso de un esquema de cerradura aproximado y los enormes requerimientos computacionales asociados con el almacenamiento y la posterior integración de la matriz de covarianza del error.

El método de Ensamble Suavizado (*ES*) es un método similar al *EnKF* que fue propuesto por van Leeuwen y Evensen (1996). La diferencia entre el *ES* y el *EnKF* es que el *ES* hace estimaciones hacia atrás en el tiempo. Herrera (1998) propuso una versión del *ES,* al que llamaremos Ensamble Suavizado de Herrera (*ESH*), para la optimización espacio temporal de las redes de monitoreo del agua subterránea. En años recientes, este último método se ha utilizado para la estimación de parámetros y el estado en modelos de flujo y transporte del agua subterránea. El método *ES* utiliza la simulación Monte Carlo, que consiste en generar realizaciones repetidas de la variable aleatoria simulada a través de un modelo de flujo y transporte, sin embargo, con frecuencia se requiere un gran número de corridas de los modelos para que los momentos converjan, por lo que una computadora serial puede requerir muchas horas de uso continuo, dependiendo del problema que se trate y la capacidad de la computadora. Por este motivo se requiere paralelizar el proceso para que corra en un tiempo razonable.

En este trabajo se implementó y probó una estrategia de paralelización para reducir el tiempo de ejecución de la simulación Monte Carlo del código que implementa el *ESH*, el cual está programado en Fortran 90 en un paquete de software llamado *GWMC* (Groundwater Monte Carlo) y fue desarrollado por Graciela Herrera y colaboradores. En la versión original, se ejecutan en serie varias realizaciones del modelo y después se construye la matriz de covarianza espaciotemporal que requiere el ESH con base en estas, por lo que la comunicación entre los procesos es muy pequeña y es una tarea fácilmente paralelizable del tipo conocido como *pleasingly parallel.* La paralelización del *GWMC* se hizo con ayuda de *Python* usando *mpi4py*, aplicando la biblioteca *MPI*. Esta estrategia se aplicó a dos casos de estudio, ambos para problemas de contaminación de acuíferos: 1) un problema sencillo en un dominio rectangular en dos dimensiones de una sola capa y 2) un problema complejo en tres dimensiones con un dominio irregular y 9 capas de espesor variable. En el primer caso se usó el simulador Princeton Transport Code (*PTC*) para modelar tanto el flujo del agua subterránea como el transporte de solutos y en el segundo los simuladores *MODFLOW* para el flujo y *MT3D* para el transporte.

Los resultados obtenidos al aplicar la estrategia de paralelización escogida al problema sencillo muestran que la aceleración crece a medida que aumenta la carga de trabajo en cada procesador, consiguiendo una eficiencia 75% con 4000 realizaciones y 96 procesadores, con un costo moderado. Sin embargo, para el problema complejo, de mayores dimensiones, se tuvo una limitación importante con los recursos de cómputo en la memoria RAM, el costo fue mucho mayor, además de no obtener buenas eficiencias con un número grande de procesadores. Aunque se tuvo una reducción de tiempo significativa con 8 procesadores, con una eficiencia de 71% para 4,000 realizaciones. Por este motivo se sugieren dos modificaciones al programa para que aumente la eficiencia al aplicarse a problemas *grandes*.

Abstract
Data assimilation is a process that uses prediction models and measurements as sources of information to improve the results of these models. The sequential data assimilation method called Ensemble Kalman Filter (*EnKF*) was designed to solve two major difficulties related to the use of the Extended Kalman Filter (*EKF*) in problems with nonlinear dynamics in large state spaces, these are the use of an approximate closure scheme and the huge computational requirements associated with the storage and subsequent integration of the error covariance matrix.

The Ensemble Smoother method (*ES*) is a method similar to the *EnKF* that was proposed by van Leeuwen and Evensen (1996). The difference between the *ES* and the *EnKF* is that the *ES* calculates estimates backward in time. Herrera (1998) proposed a version of *ES*, which we will call Herrera Ensemble Smoother (*ESH*) for spatiotemporal optimization of groundwater monitoring networks. In recent years, the latter method has been used for parameter and state estimation in groundwater flow and transport models. The *ES* method uses Monte Carlo simulation, which consists in generating repeated realizations of the simulated random variable through a flow and transport model; however, a large number of model runs are often required for the moments to converge, so a serial computer may require many hours of continuous use, depending on the problem and the capacity of the computer. For this reason, it is necessary to parallelize the process so that it runs in a reasonable time.

In this work, a parallelization strategy to reduce the execution time of the Monte Carlo simulations of the code implementing the ESH was implemented and tested. The code is programmed in Fortran 90 in a software package called *GWMC* (Groundwater Monte Carlo) and was developed by Herrera et al. In the original version, several realizations of the model are executed in series and then they are used to build the spatiotemporal covariance matrix required by ESH, so the communication between the processes in very small and it is an easily parallelizable task of the type known as *pleasingly parallel*. The parallelization of the GWMC was done with the help of *Python* using *mpi4py*, applying the *MPI* library. This strategy was applied to two case studies, both for problems of aquifer contamination: 1) a simple problem in a two-dimensional rectangular domain with a single layer and 2) a complex problem in three-dimensions with an irregular domain and 9 layers of variable thickness. In the first case, the Princeton Transport Code (*PTC*) simulator was used to model both groundwater flow and solute transport, and in the second problem, the simulators *MODFLOW* for flow and *MT3D* for transport were used.

The results obtained by applying the chosen parallelization strategy to the simple problem show that the acceleration increases as the workload increases in each processor, achieving an efficiency of 75% with 4000 realizations and 96 processors, with a moderate cost. However, for the complex problem, of greater dimensions, there was an important limitation with the computational resources in the RAM memory, the cost was much higher, besides not obtaining good efficiencies with a large number of processors. Although there was a significant reduction of time with 8 processors, with an efficiency of 71% for 4,000 realizations. For this reason, two modifications to the program are suggested so that efficiency increases when applied to large problems.

Índice

Resumen 3

Abstract i

1. Introducción 1

1.1 Planteamiento del problema 1

1.2 Revisión del estado del arte 2

1.3 Objetivo general y metas 6

2. Métodos basados en el filtro de Kalman 7

2.1 Filtro de Kalman 7

2.1.1 El proceso de estimación 9

2.1.2 El algoritmo 10

2.2 Variantes del Filtro de Kalman 12

2.2.1 Filtro de Kalman Extendido 12

2.2.2 Filtro de Kalman Ensamblado (*EnKF*) 15

2.2.3 Ensamble Suavizado (*ES*) 15

2.2.3.1 Ensamble Suavizado de van Leeuwen y Evensen (*ESLE*) 16

2.2.3.2 Ensamble Suavizado de Herrera (*ESH*) 19

3.1 Herramientas de cómputo en paralelo 22

3.2 Tipos de paralelismo 23

3.3 Métricas de rendimiento 23

3.3.1 Aceleración y eficiencia 23

3.3.2 Ley de Amdahl 24

3.3.3 Costo 24

3.4 Memoria 24

4. Paralelización de Groundwater Monte Carlo (*GWMC*) 27

4.1 Implementación del GWMC en serie 27

4.1.1 GWMC – PTC 28
4.1.2 GWMC - MFMT 37
4.2 Estrategia de paralelización del GWMC 38
5. Casos de estudio y resultados 45
5.1 Caso de estudio sencillo (*GWMC-PTC*) 45
5.1.1 Modelo de flujo y transporte 46
5.1.2 Modelo estocástico 47
5.1.3 Vector espacio temporal de medias y matriz de covarianza 48
5.1.4 Resultados 48
5.1.5 Discusión 54
5.2 Caso de estudio complejo (*GWMC-MFMT*) 57
5.2.1 Modelo de flujo y transporte 59
5.2.2 Modelo estocástico 63
5.2.3 Vector espacio temporal de medias y matriz de covarianza 64
5.2.4 Resultados 66
5.2.5 Discusión 70
6. Conclusiones 73
Referencias 75
Anexo I 82

Índice de figuras

Figura 2. 1 Variables de estado del filtro de Kalman. 8
Figura 2. 2 El ciclo del filtro de Kalman (modificado de Solera Ramírez, 2003). 10

Figura 4. 1 Esquema del software que se requiere para ejecutar el *GWMC*, utilizando el simulador *PTC*..... 29
Figura 4. 2 Diagrama de flujo del *GWMC-PTC* en serie utilizando el simulador *PTC* (modificada de Olivares, 2002). 32
Figura 4. 3 Diagrama de la subrutina readparm del programa *GWMC-PTC*. 33
Figura 4. 4 Procedimiento del posicionamiento de las conductividades hidráulicas. 34
Figura 4. 5 Diagrama del posicionamiento de las series aleatorias del contaminante. 34
Figura 4. 6 Diagrama de la subrutina meshes del programa *GWMC-PTC*. 35
Figura 4. 7 Diagrama del ciclo de la simulación Monte Carlo. 35
Figura 4. 8 Diagrama de la subrutina *randsrc* del programa *GWMC*. 36
Figura 4. 9 Diagrama de la ejecución del simulador *PTC* del programa *GWMC*. 37
Figura 4. 10 Esquema del software que se requiere para ejecutarse el *GWMC*, utilizando el simulador *MODFLOW*/*MT3D*. 38
Figura 4. 11 Proceso de paralelización. Las tareas del E al L se hacen en paralelo en cada uno de los procesadores, mientras que las tareas D y M se calculan en el procesador 1 que funciona como maestro. 41
Figura 4. 12 Esquema de la paralelización del *GWMC*, utilizando el simulador *PTC*. 43
Figura 4. 13 Esquema de la paralelización del *GWMC*, utilizando el simulador *MODFLOW/MT3D*. 44

Figura 5. 1 a) Malla del *ESH* y las condiciones para el flujo (h está en m), b) malla de simulación estocástica y las condiciones de frontera para el transporte (modificada de Olivares-Vázquez, 2002). 45
Figura 5. 2 Aceleración vs números de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso sencillo. 50
Figura 5. 3 Eficiencia vs números de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso sencillo. 51
Figura 5. 4 Grafica del costo para 1,000, 2,000 y 4,000 realizaciones para el caso sencillo. 52

Figura 5. 5 Uso de la memoria RAM por matriz en diferentes números de procesadores para el caso sencillo. 54
Figura 5. 6 La comparación entre la ley de Amdahl y la aceleración de 1,000, 2,000 y 4,000 realizaciones para el caso complejo. 55
Figura 5. 7 Comparación del uso de memoria RAM al almacenar matrices auxiliares (a) y al almacenar realizaciones (b) para el caso sencillo. 57
Figura 5. 8 Localización del área de estudio. 59
Figura 5. 9 Modelo conceptualizado, una visión vertical (modificada de McLane Environmental, 2010). 61
Figura 5. 10 a) Malla y dominio, b) submalla de simulación estocástica y las condiciones de frontera (celdas color azul) para el modelo de flujo en *MODFLOW* (modificada de McLane Environmental, 2010). 62
Figura 5. 11 Distribución de la conductividad hidráulica en las capas no homogéneas (modificada de McLane Environmental, 2010). 63
Figura 5. 12 Posiciones de observación. 65
Figura 5. 13 Posiciones de estimación. 65
Figura 5. 14 Aceleración vs número de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso complejo. 67
Figura 5. 15 Eficiencia vs número de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso complejo 67
Figura 5. 16 Gráfica del costo para 1,000, 2,000 y 4,000 realizaciones para el caso complejo. 69
Figura 5. 17 Uso de memoria RAM por matriz en diferentes números de procesadores para el caso complejo. 70
Figura 5. 18 La comparación entre la aceleración de 1,000, 2,000 y 4,000 realizaciones con la Ley de Amdahl para 4000 realizaciones para el caso complejo. 71

Índice de tablas

Tabla 5. 1 Datos de tiempo (seg), aceleración (Sp), eficiencia (Ep) y la ley de Amdahl con diferentes números de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso sencillo. 49
Tabla 5. 2 Cálculos de los costos en diferentes números de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso sencillo..... 51
Tabla 5. 3 Cálculos de la memoria RAM por matriz a utilizar en diferentes números de procesadores para el caso sencillo..... 53
Tabla 5. 4 Calculo de la memoria RAM para diferentes números de realizaciones para el caso complejo..... 56
Tabla 5. 5 Datos de tiempo de ejecución (seg), aceleración (Sp), eficiencia (Ep) y ley de Amdahl con diferentes números de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso complejo..... 66
Tabla 5. 6 Cálculos de los costos en diferentes números de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso complejo..... 68
Tabla 5. 7 Cálculos de la memoria RAM por matriz a utilizar en diferentes números de procesadores para el caso complejo..... 69
Tabla 5. 8 Calculo de la memoria RAM para las realizaciones, para el caso complejo. 72

1. Introducción

El agua es un recurso de vital importancia a nivel mundial para la humanidad, por lo que existe un gran desafío para prevenir y/o proteger las reservas de agua subterránea de la contaminación que pueda sufrir.

La calidad del agua de un acuífero depende de la dinámica del flujo del agua en el mismo y para obtener información sobre esta dinámica y de la calidad del agua se pueden hacer estudios geológicos, hidrológicos y geoquímicos, entre otros. El análisis de esta información permite desarrollar un modelo conceptual de la dinámica de flujo del acuífero y de su calidad, que a su vez permite elaborar un modelo matemático que lo represente.

La modelación matemática es el método más efectivo para predecir el comportamiento de los diversos sistemas de interés. Los modelos de flujo y transporte de solutos son los más usados para modelar el agua subterránea, existe software especializado de dominio público o privado que implementa estos modelos (Leyva Suárez, 2010). Además, estos pueden contribuir a encontrar valores medios de los parámetros que se deben utilizar en los modelos de flujo a escala regional y determinar la influencia de la heterogeneidad no modelada de la calidad de las predicciones obtenidas con dichos modelos. La heterogeneidad espacial puede ser tratada de forma alternativa considerando los parámetros hidrogeológicos como funciones aleatorias o procesos estocásticos, que se puedan caracterizar mediante algunos parámetros estadísticos; lo que induce a que las ecuaciones diferenciales que describen el flujo y transporte de solutos se conviertan en ecuaciones diferenciales estocásticas o modelos estocásticos.

1.1 Planteamiento del problema

La hidrología estocástica es un campo que trata con métodos estocásticos para describir y analizar procesos de aguas subterráneas (Renard, 2007). Una parte importante de ésta consiste en resolver modelos estocásticos que describen esos procesos, para estimar la función de densidad de probabilidad conjunta de los parámetros (por ejemplo: conductividad, coeficiente de almacenamiento) y/o variables de estado de esas ecuaciones (por ejemplo: niveles de agua subterránea, concentraciones) o más comúnmente algunos de sus momentos. La simulación Monte Carlo (SMC) es una alternativa para resolver estos modelos estocásticos, se basa en la idea de aproximar la solución de procesos estocásticos utilizando un gran número de realizaciones igualmente probables. Este método consiste en tres pasos: generar múltiples realizaciones de los

parámetros de la ecuación que se consideren inciertos, resolver para cada realización de los parámetros, las ecuaciones por medio de métodos numéricos y promediar las múltiples soluciones (realizaciones) del modelo para obtener los momentos buscados.

Para reducir la incertidumbre de los modelos estocásticos podemos usar como segunda fuente de información los datos disponibles. Como también los datos tienen incertidumbre, usamos un método llamado asimilación de datos que combina los resultados de los modelos con los datos para obtener una mejor estimación de del estado. Existen diferentes métodos de asimilación de datos, en este trabajo aplicamos el método ensamble suavizado de Herrera (*ESH*). La versión más conocida del ensamble suavizado (*ES*) fue desarrollada por van Leeuwen y Evensen en 1996. Herrera en 1998 desarrolló una versión de este método de forma independiente, a la que originalmente llamó filtro de Kalman estático, y a la que en esta tesis nos referimos como *ESH* para distinguirla de la versión de van Leeuwen y Evensen. Herrera (1998) propuso el *ESH* como método de estimación, como parte de una metodología para el diseño óptimo de redes de monitoreo de la calidad del agua subterránea.

La gran cantidad de realizaciones requeridas por la SMC puede ser muy exigente en recursos informáticos y el tiempo de cálculo puede resultar excesivo. Hoy en día existen muchas plataformas informáticas paralelas que se pueden utilizar para aliviar este problema. El software Groundwater Monte Carlo (*GWMC*), como su nombre lo indica, implementa la simulación Monte Carlo requerida por el ESH. En su versión original, se ejecutan en serie varias realizaciones del modelo y posteriormente se construye una matriz de covarianza con base en estas, por lo que la comunicación entre los procesos es muy pequeña y es una tarea fácilmente paralelizable del tipo conocido como *pleasingly parallel* (Ridgway, Terry y Babak, 2005). En esta tesis se paralelizó este código usando *MPI* (*Message Passing Interface*) y *Python*, y se ejecutó en una arquitectura de cómputo de memoria distribuida. El código ya paralelizado se aplicó a dos casos de estudio, ambos para problemas de contaminación de acuíferos: 1) un problema en un dominio rectangular en dos dimensiones de una sola capa y 2) un problema en tres dimensiones con un dominio irregular y 9 capas de espesor variable.

1.2 Revisión del estado del arte

En esta sección se presenta una revisión del estado del arte en el tema de asimilación de datos utilizando el método de filtro de Kalman ensamblado (*EnKF*) y sus derivados, con especial atención en el filtro de Kalman ensamblado (*EnKF*) y el ensamble suavizado (*ES*). Se tocan tres temas

relacionados con la investigación realizada en esta tesis, éstos son: el uso de estos métodos en problemas del agua subterránea, métodos de paralelización utilizados al aplicarlos en otras áreas del conocimiento y, por último, métodos de paralelización utilizados al aplicarlos en problemas del agua subterránea.

Los trabajos que se han dirigido a investigar problemas del agua subterránea utilizando el *EnKF* y el *ES,* han tenido diferentes objetivos. Entre ellos, estimar la distribución espacial de la conductividad hidráulica (o la transmisividad hidráulica) del acuífero y/o la carga hidráulica (Chen y Zhang, 2006; Bauser et al., 2010; Briseño, 2011; Bailey y Baù, 2010) utilizando diferentes tipos de datos, por ejemplo Cheng y Zhang (2006) estimaron el campo de la conductividad hidráulica mediante la asimilación de la carga de presión y las mediciones de conductividad hidráulica y explorar la sensibilidad del *EnKF* con distintos factores; Bauser et al. (2010), recopilaron valores diarios del nivel del río, la entrada lateral, la recarga natural y de bombeo durante cierto periodo para simular en línea el control óptimo de la recarga artificial; Briseño et al. (2011) utilizaron datos de carga hidráulica en diferentes pozos y tiempos de monitoreo para estimar la carga hidráulica y la incertidumbre del error de esta estimación; Bailey y Baù (2010) utilizaron mediciones de carga hidraúlica y volúmenes de flujo de retorno para condicionar la conductividad hidraúlica para las simulaciones de flujo transitorio en 2D. Estimar los parámetros del modelo de transporte y/o la concentración de solutos (Herrera, 1998; Herrera y Pinder, 2005; Zhang et al., 2005; Liu et al., 2008; Kollat et al., 2008; Nowak et al., 2010; Crestani et al., 2013) utilizando diferentes tipos de datos, por ejemplo, Herrera (1998) y Herrera y Pinder (2005) utilizaron datos de concentración de un contaminante para la estimación de la misma variable; Liu et al (2008) se basan en la carga hidráulica y los datos de concentración (tritio) recolectados en el campo, para aplicar el *EnKF* y estimar la distribución de la carga hidráulica y diversos parámetros del modelo de transporte; Kollat et al. (2008) utilizaron datos de concentración del cloruro de antimonio para un experimento de rastreo de 19 días, mostrando el aumento de la precisión del pronóstico al filtrar los errores sistemáticos del modelo en las predicciones del transporte de agua subterránea por separado y permitir que vuelvan a las predicciones de concentración; Nowak et al. (2010) con 24 mediciones de conductividad y carga hidraúlica colocadas, optimizadas para minimizar la varianza de predicción de concentración de contaminantes en estado estacionario, así como también el tiempo de llegada de los contaminantes a una ubicación ecológicamente sensible con la finalidad de transferir el diseño geoestadístico Bayesiano de kriging como aplicaciones a problemas geoestadísticos inversos; Crestani et al. (2013) utilizaron datos de concentración con el fin de

estimar el campo de la conductividad hidraúlica a través de la asimilación de mediciones, implentándolos en el mismo modelo de transporte para *EnKF* y *ES* e investigar la capacidad de dichos filtros.

Aplicaciones de estos métodos al diseño de redes de monitoreo del agua subterránea las han llevado a cabo: Herrera (1998) y Herrera y Pinder (2005) los utilizaron para proponer un método para el diseño espaciotemporal óptimo de redes de monitoreo de la calidad del agua subterránea, utilizando un modelo de transporte estocástico y el método de optimización que involucra de manera combinada el espacio y tiempo; Zhang et al. (2005) con datos de conductividad hidraúlica usando la técnica Latin Hypercube Sampling (*LHS*) diseñaron una red de monitoreo óptima de la calidad del agua subterránea combinando el *ESH*, un algoritmo genético, un modelo de flujo y transporte para reducir al máximo el coeficiente; Kollat et al. (2011) utilizaron datos de cloruro de antimonio para contribuir al marco Adaptive Strategies for Sampling in Space Time (*ASSIST*) para mejorar el diseño de monitoreo de aguas subterráneas a largo plazo; Alzraiee et al. (2013) utilizan datos de conductividad hidraúlica y carga hidraúlica para proponer un marco de optimización multiobjetivo para ayudar al diseño de redes de monitoreo de la conductividad y carga hidraúlica con el objetivo de optimizar sus predicciones espaciales y estimar los parámetros geoestadísticos del campo *K*.

Hay trabajos enfocados en problemas de diferentes áreas del conocimiento como: oceanografía (Keppenne, 2000; Keppenne y Rienecker, 2002; van Hees et al., 2003; Keppenne et al., 2005), hidrometerología (Riechle y Koster, 2003) y geotermia (Rostami et al., 2014) entre otros, que utilizan métodos ensamble con software paralelizado. La herramienta de paralelización más utilizada es Message Passing Interface (*MPI*), aunque algunos trabajos además de utilizar esta estrategia también emplearon la descomposición de dominio horizontal en dos dimensiones, para lograr un mínimo de comunicaciones entre procesadores e implementar de manera eficiente el método de asimilación de datos *EnKF* (Keppenne, 2000; Keppenne y Rienecker, 2002; Riechle y Koster, 2003; Keppenne et al., 2005), logrando mejores resultados en modelos tridimensionales, teniendo como desventaja que se requieran más recursos computacionales haciendo más difícil su aplicación (Riechle y Koster, 2003). Hay que destacar un trabajo donde utilizaron dos herramientas en paralelo, *MPI* y *OpenMP* en diferentes sistemas de cómputo (van Hees et al., 2003) en donde se infiere que es de gran importancia tanto el sistema de cómputo que se emplea como la estrategia de paralelización a utilizar. Más reciente Rostami et al. (2014) utilizan una estrategia de

paralelización que reutiliza el modelo de memoria de paralelización *OpenMP* para realizar paralelización en *MPI*, mostrando que la aceleración aumenta conforme incrementa el número de procesadores, demostrando que su código paralelo escala mejor cuando el número de realizaciones aumenta.

Sin embargo, son pocos los trabajos que involucran métodos de asimilación de datos y estrategias de paralelización en problemas de agua subterránea. Entre estos trabajos se tiene el de Xu et al. (2013) quienes presentaron un algoritmo paralelo para los pasos de pronóstico y análisis, para reducir el tiempo de ejecución del *EnKF* para tamaños grandes de ensamble, logrando reducir significativamente el tiempo al ejecutar el *EnKF*, además de ser más eficiente cuanto más grande es el tamaño del ensamble. Kurtz et al. (2014) aplicaron el *EnKF* en un sistema fluvial-acuífero usando de manera conjunta datos piezométricos y temperaturas del agua subterránea, para predecir en tiempo real el estado del acuífero e identificar los parámetros hidráulicos del subsuelo. Paralelizaron el *EnKF*, en las llamadas de corridas hacia adelante y en el paso de actualización, logrando escalar hasta 128 procesadores con una eficiencia alrededor del 70%.

Algunos trabajos previos se han enfocado en problemas del agua subterránea utilizando métodos de paralelización para la simulación Monte Carlo, problema que está muy relacionado con la paralelización de métodos de ensamble y con lo que se hace en esta tesis. Por ejemplo, Dong et al. (2012) describen una estrategia de paralelización de este problema utilizando el Marco de Procesamiento Paralelo de Java (*JPPF*, sus siglas en inglés), esta herramienta es muy poderosa y se puede usar como un *grid middle-ware*, que es una arquitectura de protocolo en la que existen los servicios necesarios para admitir un conjunto común de aplicaciones en un entorno de red distribuido (Foster et al., 2001), para distribuir tareas a través de varios sistemas informáticos. En este caso los autores la aprovechan para evitar modificar *MODFLOW* (Mc Donald y Harbaugh, 1998) y programas relacionados. Implementaron un servidor JPPF en el nodo maestro y un nodo JPPF en cada nodo informático. Los nodos JPPF pueden procesar simultáneamente hasta ocho tareas (una tarea por subproceso), porque cada nodo tiene ocho núcleos. Además, utilizan la combinación de dos niveles de paralelismo en el software, uno de grano grueso en el cálculo de MODFLOW y otro de grano fino en el solucionador del gradiente conjugado precondicionado (PCG, por sus siglas en inglés; Dong y Li, 2009). Los diferentes niveles de paralelismo se explicarán en la sección 3.2. En nuestro caso, sólo se utilizó paralelismo de grano grueso.

1.3 Objetivo general y metas

El objetivo general de esta tesis de doctorado es escoger, implementar y probar una estrategia de paralelización para reducir el tiempo de ejecución de la simulación Monte Carlo del código que implementa el ensamble suavizado de Herrera (*ESH*).

La tesis está ordenada como sigue: en el capítulo dos se describe el filtro de Kalman y algunas de sus variantes. En el capítulo tres se explican los conceptos básicos del cómputo en paralelo, así como las métricas de rendimiento. En el capítulo cuatro se da una introducción al paquete GWMC y el software que implementa el *ESH*. En el capítulo cinco se presentan dos casos de estudio, uno sencillo y otro complejo, que se utilizan para probar el programa en paralelo. Las conclusiones y recomendaciones de esta tesis se presentan en el capítulo seis.

2. Métodos basados en el filtro de Kalman

En Osses (2008) se define la asimilación de datos como un conjunto de métodos en los que se puede combinar: la información a priori, las observaciones, un modelo de pronóstico y un modelo de observaciones para estimar el estado más probable.

El filtro de Kalman es un método de asimilación de datos que puede utilizarse para la estimación estocástica, considerando que es un conjunto de ecuaciones matemáticas que se pueden calcular por un medio computacional de manera recursiva eficiente para estimar el estado de un proceso, minimizando el error estimado de la covarianza, cuando algunas condiciones son dadas (Castañeda Cárdenas et al., 2013; Welch y Bishop, 2002).

2.1 Filtro de Kalman

Kalman (1960) introdujo el filtro que lleva su nombre, en éste detalla la manera en que se resuelve el problema de filtrado lineal con datos discretos.

De acuerdo con Solera Ramírez (2003) el filtro de Kalman es el conjunto de ecuaciones matemáticas que proporcionan una solución recursiva y eficiente al problema de mínimos cuadrados. Esta solución calcula un estimador lineal, insesgado y óptimo del estado de un proceso en el paso de tiempo k con base en la información disponible en el paso de tiempo anterior, y con información adicional en el tiempo $k+1$ actualizar el estimador lineal. El filtro de Kalman es el principal algoritmo para estimar sistemas dinámicos representados en el espacio de los estados. También son muy convenientes para abordar el manejo de un amplio rango de modelos de series de tiempo (Solera Ramírez, 2003; Kikut V., 2003). La representación en el espacio de los estados es conveniente para la estimación de modelos estocásticos con errores tanto en la medición del sistema como en la información adicional que se incorpora, se supone que estos errores tienen una distribución normal con media cero y varianza dada, lo que permite abordar el manejo de un amplio rango de modelos. Entre los usos particulares se encuentra la modelación de componentes no observables y parámetros que cambian en el tiempo (Solera Ramírez, 2003).

La derivación del filtro de Kalman descansa en el supuesto de normalidad del vector de estado inicial y de las perturbaciones del sistema, así como de los errores de la información adicional que se incorpora, como se comentó antes.

Debido a la linealidad del problema, de los supuestos se deriva que el estado también tiene una distribución normal, simplificando el cálculo de la solución óptima que satisface que la estimación insesgada, con errores de estimación de varianza mínima.

El filtro de Kalman es un algoritmo para procesar datos de manera iterativa. No requiere reprocesar todas las observaciones anteriores cada vez que se actualiza. Éste produce estimaciones que se actualizan con la llegada de cada nueva observación, pudiendo modelar sistemas cuyos parámetros cambian a través del tiempo. La solución es óptima por cuanto el filtro combina toda la información observada y el conocimiento previo acerca del comportamiento del sistema para producir una estimación del estado de tal manera que el error es minimizado estadísticamente.

El filtro de Kalman es el principal algoritmo para estimar sistemas dinámicos representados en el espacio de los estados. En esta representación el sistema se describe por medio de un conjunto de variables denominadas de estado (Figura 2.1). El estado contiene toda la información relativa al sistema a un cierto punto en el tiempo. Esta información debe permitir la inferencia del comportamiento pasado del sistema, con el objetivo de predecir su comportamiento futuro.

Figura 2. 1 Variables de estado del filtro de Kalman.

Lo que hace al filtro tan interesante es precisamente su habilidad para predecir el estado de un sistema en el pasado, presente y futuro, aun cuando la naturaleza precisa del sistema modelado es desconocida. En la práctica, las variables de estado individuales de un sistema dinámico no se pueden determinar exactamente por una medición directa. Dado lo anterior, su medición se realiza por medio de procesos estocásticos que involucran algún grado de incertidumbre en la medición.

El filtro de Kalman tiene como objetivo resolver el problema general de estimar el estado.

2.1.1 El proceso de estimación

El objetivo del filtro de Kalman discreto es calcular un estimador lineal, insesgado y óptimo del estado de un sistema en el tiempo $k+1$ con base en la información disponible en el tiempo k, y actualizar dicha estimación con los datos adicionales disponibles en el tiempo k (Welch y Bishop, 2002).

El filtro de Kalman discreto tiene como objetivo resolver el problema general de estimar el estado $\mathbf{X} \in \Re^n$ de un proceso controlado en tiempo discreto, el cual se representa por una ecuación dinámica estocástica lineal en diferencias de la siguiente forma:

$$\mathbf{X}_{k+1} = \mathbf{A}_k \mathbf{X}_k + \mathbf{w}_k \qquad (2.1)$$

Con una medición $\mathbf{Z} \in \Re^m$, que es

$$\mathbf{Z}_k = \mathbf{H}_k \mathbf{X}_k + \mathbf{v}_k \qquad (2.2)$$

donde $\mathbf{X}_k$ es el estado en el tiempo k, $\mathbf{A}_k$ es la matriz de dimensión $n \times n$ que relaciona al estado en el tiempo k con el estado en el tiempo $k+1$ y $\mathbf{H}_k$ es la matriz de dimensión $m \times n$ que relaciona el estado en el tiempo k con la medición $\mathbf{Z}_k$ en el mismo tiempo.

Las variables aleatorias $\mathbf{w}_k$ y $\mathbf{v}_k$ representan el error del proceso y de las mediciones respectivamente. Se supone que son independientes entre ellas, que son ruido blanco, con distribución de probabilidad normal:

$$p(\mathbf{w}_k) \sim N(0, \mathbf{Q}_k) \qquad (2.3)$$

$$p(\mathbf{v}_k) \sim N(0, \mathbf{R}_k) \qquad (2.4)$$

Las matrices de covarianza del error del proceso $\mathbf{Q}_k$ y del error de las mediciones $\mathbf{R}_k$, pueden cambiar en el tiempo.

Sea la secuencia de observaciones $\mathbf{Z}_l = \{z_1, \ldots, z_l\}$ y dada una realización de la misma, el problema de estimación discreta consiste en calcular la estimación de $\mathbf{X}_k$ basada en $\mathbf{Z}_l$ (Jazwinski, 1970). Si:

- ✓ $k < l$, el problema se llama problema de suavizado discreto.
- ✓ $k = l$, el problema se llama problema de filtrado discreto.
- ✓ $k > l$, se llama el problema de predicción discreto.

2.1.2 El algoritmo

El filtro de Kalman estima el estado anterior utilizando un proceso de retroalimentación, esto es, estima el estado en algún tiempo utilizando el modelo de diferencias finitas y entonces obtiene la retroalimentación por medio de los datos observados.

Las ecuaciones que se utilizan al aplicar el filtro de Kalman se pueden dividir en dos grupos:

- *Las ecuaciones de predicción o pronóstico*: estas ecuaciones se encargan de la predicción del estado en el tiempo $k+1$ tomando como referencia el estado en el tiempo k y del cálculo de la matriz de covarianza del error de la predicción del estado.

- *Las ecuaciones de actualización o corrección*: estas ecuaciones se encargan de la estimación del estado con nuevos datos observados, es decir, incorporan nueva información a la estimación anterior con lo cual se llega a una estimación mejorada del estado.

El algoritmo de estimación final puede definirse como un algoritmo de pronóstico–corrección para resolver numerosos problemas. Así, el filtro de Kalman funciona por medio de un mecanismo de proyección y corrección al pronosticar el nuevo estado y su incertidumbre y corregir la proyección con la nueva medida. Este ciclo se muestra en la figura 2.2.

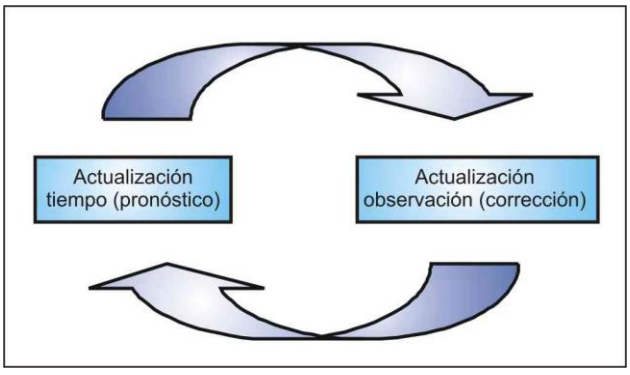


Figura 2. 2 El ciclo del filtro de Kalman (modificado de Solera Ramírez, 2003).

El primer paso consiste en generar un pronóstico del estado hacia adelante en el tiempo tomando en cuenta toda la información disponible en ese momento y en un segundo paso, se genera un pronóstico mejorado del estado, de tal manera que el error se minimiza estadísticamente.

Las ecuaciones específicas para el pronóstico del filtro de Kalman discreto son:

$$\hat{\mathbf{X}}_{k+1}^{k} = \mathbf{A}_k \hat{\mathbf{X}}_k^k \qquad (2.5)$$

$$\mathbf{P}_{k+1}^{k} = \mathbf{A}_k \mathbf{P}_k^k \mathbf{A}_k^T + \mathbf{Q}_k \qquad (2.6)$$

Las ecuaciones (2.5) y (2.6) proyectan las estimaciones del estado y la covarianza hacia adelante del tiempo k al $k+1$. La matriz $\mathbf{A}_k$ relaciona el estado en el tiempo previo k con el estado en el tiempo actual $k+1$, $\mathbf{Q}_k$ representa la covarianza del error del proceso que modela el estado, $\mathbf{P}_{k+1}^{k}$ es la matriz de covarianza.

Las ecuaciones de la corrección del estado del filtro de Kalman discreto son:

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}^{k} \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_{k+1}^{k} \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \qquad (2.7)$$

$$\hat{\mathbf{X}}_{k+1}^{k+1} = \hat{\mathbf{X}}_{k+1}^{k} + \mathbf{K}_{k+1}(\mathbf{Z}_{k+1} - \mathbf{H}_k \hat{\mathbf{X}}_{k+1}^{k}) \qquad (2.8)$$

$$\mathbf{P}_{k+1}^{k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_k)\mathbf{P}_{k+1}^{k} \qquad (2.9)$$

El subíndice identifica el tiempo en que se estima el estado y el superíndice identifica el tiempo de la medición más reciente que se utiliza para obtener la estimación. La ecuación (2.7) es la ganancia de Kalman, la ecuación (2.8) es la nueva estimación del estado y la ecuación (2.9) es la nueva estimación de la covarianza.

Lo primero que se realiza durante la corrección de la proyección del estado, es el cálculo de la ganancia de Kalman, $\mathbf{K}_{k+1}$ (ecuación 2.7). Esta matriz de ponderación o ganancia es la que minimiza la varianza del error de la nueva estimación del estado. El siguiente paso es el proceso para obtener $\mathbf{Z}_{k+1}$ y generar una nueva estimación del estado que incorpora la nueva observación con la ecuación (2.8). El paso final es obtener una nueva estimación de la covarianza del error mediante la ecuación (2.9).

Después de cada actualización, el proceso se repite tomando como punto de partida las nuevas estimaciones del estado y de la covarianza del error, es decir, se repiten las ecuaciones (2.7) a (2.9), por lo que se hace recursivo. Esta naturaleza recursiva es una de las características importantes del filtro de Kalman.

2.2 Variantes del Filtro de Kalman

De las múltiples variantes existentes del Filtro de Kalman, sólo mencionaremos algunas, que son de importancia para este trabajo.

2.2.1 Filtro de Kalman Extendido

El filtro de Kalman se utiliza para problemas lineales, pero si se trata de abordar problemas no lineales, ya sea en la dinámica o en la relación de las mediciones con la variable a estimar, una posibilidad es usar el filtro de Kalman extendido (*EKF*, por sus siglas en inglés), en el cual se utiliza una ecuación de aproximación linealizada para llevar a cabo la predicción o establecer la relación de la variable a estimar con los datos.

Para ejemplificar la derivación del filtro de Kalman extendido (*EKF*, por sus siglas en inglés) utilizaremos un modelo no lineal escalar (Evensen, 2009). Se tiene un modelo no lineal

$$X_{k+1} = G(X_k) + q_k \qquad (2.10)$$

donde $G(X_k)$ es una función no lineal del proceso, q_k es el error del proceso con una distribución normal, con media cero y varianza del error del proceso Q_k, $N(0, Q_k)$.

Entonces, se propone que la evolución del modelo se aproxime de acuerdo con la ecuación:

$$\hat{X}^k_{k+1} = G(\hat{X}^k_k) \qquad (2.11)$$

Restando las ecuaciones (2.10) y (2.11) se obtiene la linealización

$$X_{k+1} - \hat{X}^k_{k+1} = G(X_k) - G(\hat{X}^k_k) + q_k \qquad (2.12)$$

Utilizando la expansión en series de Taylor para $G(X_k)$ alrededor de $\hat{X}^k_k$, se tiene:

$$G(X_k) = G(\hat{X}^k_k) + G'(\hat{X}^k_k)(X_k - \hat{X}^k_k) + \frac{1}{2}G''(\hat{X}^k_k)(X_k - \hat{X}^k_k)^2 + \ldots \qquad (2.13)$$

Sustituyendo la ecuación (2.13) en la ecuación (2.12), se tiene:

$$X_{k+1} - \hat{X}_{k+1}^{k} = G'(\hat{X}_k^k)(X_k - \hat{X}_k^k) + \frac{1}{2}G''(\hat{X}_k^k)(X_k - \hat{X}_k^k)^2 + \ldots + q_k \qquad (2.14)$$

Elevando al cuadrado de la ecuación (2.14), se tiene:

$$\left(X_{k+1} - \hat{X}_{k+1}^{k}\right)^2 = \left(G'(\hat{X}_k^k)(X_k - \hat{X}_k^k) + \frac{1}{2}G''(\hat{X}_k^k)(X_k - \hat{X}_k^k)^2 + \ldots + q_k \right)^2$$

$$= (X_k - \hat{X}_k^k)^2 \left(G'(\hat{X}_k^k)\right)^2 + \frac{1}{2}(X_k - \hat{X}_k^k)^3 G'(\hat{X}_k^k)G''(\hat{X}_k^k) + \frac{1}{4}(X_k - \hat{X}_k^k)^4 \left(G''(\hat{X}_k^k)\right)^2 + \cdots + (q_k)^2$$

(2.15)

Tomando el valor esperado

$$E\left\{\left(X_{k+1} - \hat{X}_{k+1}^{k}\right)^2\right\} = E\left\{\left(G'(\hat{X}_k^k)(X_k - \hat{X}_k^k) + \frac{1}{2}G''(\hat{X}_k^k)(X_k - \hat{X}_k^k)^2 + \ldots + q_k \right)^2\right\} \qquad (2.16)$$

Por propiedades del valor esperado, se tiene

$$E\left\{\left(X_{k+1} - \hat{X}_{k+1}^{k}\right)^2\right\} = E\left\{\begin{array}{l} (X_k - \hat{X}_k^k)^2 \left(G'(\hat{X}_k^k)\right)^2 + \frac{1}{2}(X_k - \hat{X}_k^k)^3 G'(X_k^k)G''(\hat{X}_k^k) + \\ \frac{1}{4}(X_k - \hat{X}_k^k)^4 \left(G''(\hat{X}_k^k)\right)^2 + \cdots + (q_k)^2 \end{array}\right\}$$

$$E\left\{\left(X_{k+1} - \hat{X}_{k+1}^{k}\right)^2\right\} = E\left\{(X_k - \hat{X}_k^k)^2 \left(G'(\hat{X}_k^k)\right)^2\right\} +$$

$$E\left\{\frac{1}{2}(X_k - \hat{X}_k^k)^3 G'(X_k^k)G''(\hat{X}_k^k)\right\} +$$

$$E\left\{\frac{1}{4}(X_k - \hat{X}_k^k)^4 \left(G''(\hat{X}_k^k)\right)^2\right\} + \cdots + E\left\{(q_k)^2\right\}$$

$$= E\left\{(X_k - \hat{X}_k^k)^2\right\}\left(G'(\hat{X}_k^k)\right)^2 + \frac{1}{2}E\left\{(X_k - \hat{X}_k^k)^3\right\}G'(X_k^k)G''(\hat{X}_k^k)$$

$$+ \frac{1}{4}E\left\{(X_k - \hat{X}_k^k)^4\right\}\left(G''(\hat{X}_k^k)\right)^2 + \cdots + E\left\{(q_k)^2\right\}$$

De la ecuación resultante, se desprecian el tercer y los subsecuentes momentos,

$$E\left\{\left(X_{k+1}-\hat{X}_{k+1}^{k}\right)^2\right\}=E\left\{(X_k-\hat{X}_k^k)^2\right\}\left(G'(\hat{X}_k^k)\right)^2 \qquad (2.17)$$

Entonces la aproximación de la varianza del error resulta:

$$P_{k+1}^k \cong P_k^k(G'(\hat{X}_k^k))^2 \qquad (2.18)$$

Las ecuaciones dinámicas (2.11) y (2.18) constituyen junto con las ecuaciones (2.7), (2.8) y (2.9), el filtro de Kalman extendido para el caso de una variable de un estado escalar. Es claro que ahora se tiene una ecuación aproximada para la evolución de la covarianza del error, debido a la linealización usada.

La derivación del *EKF* en forma matricial, se basa en los mismos principios que en el caso escalar y se puede encontrar en varios libros sobre teoría de control (Jazwinski, 1970). Si se tiene un modelo no lineal, ahora el vector del estado real en el tiempo k se calcula a partir de:

$$\mathbf{X}_{k+1}=\mathbf{G}(\mathbf{X}_k)+\mathbf{w}_k \qquad (2.19)$$

Y el pronóstico se calcula a partir de la aproximación:

$$\hat{\mathbf{X}}_{k+1}^k=\mathbf{G}(\hat{\mathbf{X}}_k^k) \qquad (2.20)$$

La estadística del error de predicción se describe entonces a través de la matriz de covarianza del error $\mathbf{P}_{k+1}^k$. El filtro de Kalman extendido se basa en la suposición de que la contribución de todos los términos de orden alto es despreciable (Evensen G. , 2009). Removiendo esos términos, queda una ecuación aproximada de la covarianza del error:

$$\mathbf{P}_{k+1}^k \cong \mathbf{G}_k^t\mathbf{P}_k^k\mathbf{G}_k^T \qquad (2.21)$$

Donde $\mathbf{G}_k^t$ es la matriz jacobiana,

$$\mathbf{G}_k^t=\left.\frac{\partial\mathbf{G}(\mathbf{X})}{\partial\mathbf{X}}\right|_{X_t} \qquad (2.22)$$

Si la relación entre las mediciones y el estado es lineal, se utiliza la ecuación (2.2) y se usan las ecuaciones (2.7), (2.8) y (2.9), si no fuera lineal, de manera similar, la matriz $\mathbf{H}$ en estas ecuaciones se sustituye por el jacobiano de la función no lineal que establece la relación entre éstos.

Una derivación similar se puede hacer para la ecuación de medición no lineal

$$\mathbf{Z}_k = \mathbf{H}(\mathbf{X}_k) + \mathbf{v}_k \qquad (2.23)$$

donde $\mathbf{Z}_k \in \Re^{m_k}$, $\mathbf{H}(\cdot)$ es un operador de medición no lineal que relaciona las observaciones con la variable del estado del modelo, de dimensión $(m_k \times n)$ igual al número de mediciones en el tiempo dado.

2.2.2 Filtro de Kalman Ensamblado (*EnKF*)

Otro método secuencial de asimilación de datos, se llama filtro de Kalman ensamblado (*EnKF*, por sus siglas en inglés). Evensen G. (1994) lo propuso como una alternativa estocástica o Monte Carlo para resolver los problemas relacionados con el uso del *EKF* para modelos altamente no-lineales.

El *EnKF* utiliza las ecuaciones de corrección del estado del filtro de Kalman (ecuaciones 2.7, 2.8 y 2.9) y la ecuación (2.5) se sustituye la estimación de la media por la media del ensamble. Los cálculos no se llevan a cabo de manera directa, sino que se almacenan las realizaciones y cada una de ellas se actualiza de manera independiente, en cambio, la matriz de covarianza no se almacena. Por otro lado, se utilizan mediciones perturbadas (generando para cada medición real un conjunto de errores sin sesgo que se adicionan a la misma) ya que Burgers et al. (1998) mostró que esto se requiere para obtener la covarianza correcta. Los detalles del cálculo y la implementación se pueden ver en Evensen (2003). Las ecuaciones del EnKF son iguales a las del ES que se presentan a continuación, pero aplicadas al estado en un solo tiempo.

El *EnKF* ha ganado popularidad debido a su formulación conceptual simple y su relativa facilidad de implementación, además de que los requerimientos computacionales son asequibles. La mayor ventaja del *EnKF*, es que no requiere hacer la linealización de las ecuaciones.

2.2.3 Ensamble Suavizado (*ES*)

Entre los métodos secuenciales de asimilación de datos, también se tiene el Ensamble Suavizado (*ES*), tanto van Leeuwen y Evensen (1996) como Herrera (1998) desarrollaron una versión del *ES*

de manera independiente, por lo que, en este trabajo doctoral para diferenciarlos le llamaremos Ensamble Suavizado de van Leeuwen y Evensen (*ESLE*), y Ensamble Suavizado de Herrera (*ESH*), respectivamente.

2.2.3.1 Ensamble Suavizado de van Leeuwen y Evensen (*ESLE*)

Van Leeuwen y Evensen (1996) desarrollaron este método como un suavizado lineal de mínima varianza. La explicación que se da aquí del ensamble suavizado sigue la de Evensen (2009). Sea $\left\{\mathbf{X}_k^{(1)},...,\mathbf{X}_k^{(N_r)}\right\}$ un conjunto de N_r realizaciones del estado $\mathbf{X}$ en el tiempo k definido en la ecuación (2.10) (conjunto al que nos referiremos, como suele hacerse en la literatura del tema, como ensamble). Se define la media del ensamble $\overline{\mathbf{X}}_k$ como

$$\overline{\mathbf{X}}_k = \frac{1}{N_r}\sum_{i=1}^{N_r}\mathbf{X}_k^{(i)} \qquad (2.29)$$

y denotaremos la covarianza del ensamble en el tiempo k como $\mathbf{P}_k^e$. Esta matriz de covarianza es una estimación de la covarianza de la población y, como ocurre con cualquier estadígrafo, sus valores dependerán del conjunto de realizaciones que se utilicen al calcularla.

Definimos la matriz $\mathbf{B}_k \in \Re^{n\times N_r}$, que contienen a estas realizaciones, como

$$\mathbf{B}_k = \left(\mathbf{X}_k^{(1)},...,\mathbf{X}_k^{(N_r)}\right) \qquad (2.30)$$

donde el número de realización se utiliza como superíndice.

La media del ensamble se almacena en cada columna de $\overline{\mathbf{B}}_k \in \Re^{n\times N_r}$, la cual se define como

$$\overline{\mathbf{B}}_k = \mathbf{B}_k \cdot \mathbf{J}_{N_r} \qquad (2.31)$$

donde $\mathbf{J}_{Nr} \in \Re^{N_r\times N_r}$ es la matriz con elementos iguales a $\frac{1}{N_r}$.

Definimos la matriz de perturbación del ensamble como

$$\mathbf{B'}_k = \mathbf{B}_k - \overline{\mathbf{B}}_k = \mathbf{B}_k\left(\mathbf{I} - \mathbf{J}_{N_r}\right) \qquad (2.32)$$

La covarianza del ensamble $\mathbf{P}_k^e \in \Re^{n\times n}$, se pueden escribir como

$$\mathbf{P}_k^e = \frac{\mathbf{B'}_k \left(\mathbf{B'}_k\right)^T}{N_r - 1} \qquad (2.33)$$

Dadas las matrices del ensamble para los diferentes tiempos $\mathbf{B}_k$, $k = 1,...,i$, se puede definir la matriz del ensamble para el estado del tiempo t_0 al tiempo t_i como

$$\widetilde{\mathbf{B}}_i = \begin{pmatrix} B_0 \\ \vdots \\ B_i \end{pmatrix} \qquad (2.34)$$

Ahora introducimos la matriz de perturbación espaciotemporal

$$\widetilde{\mathbf{B}}'_i = \widetilde{\mathbf{B}}_i - \overline{\widetilde{\mathbf{B}}i} \qquad (2.35)$$

La covarianza del ensamble espaciotemporal entre el estado del modelo en los tiempos t_0 a t_i es:

$$\widetilde{\mathbf{P}}_i^e = \frac{\widetilde{\mathbf{B}}'_i \left(\widetilde{\mathbf{B}}'_i\right)^T}{N_r - 1} \qquad (2.36)$$

Sea $\mathbf{Z}_j \in \Re^{m_j}$ un vector de mediciones, con m_j el número de mediciones en el tiempo $t_{i(j)}$, Se pueden definir N_r vectores de mediciones perturbadas como

$$\mathbf{d}_j^{(l)} = \mathbf{Z}_j + \in_j^{(l)}, \qquad l = 1,...,N_r \qquad (2.37)$$

las cuales se almacenan en las columnas de una matriz, es decir

$$\mathbf{D}_j = \left(\mathbf{d}_j^{(1)}, \mathbf{d}_j^{(2)}, ..., \mathbf{d}_j^{(N_r)}\right) \in \Re^{m_j \times N_r} \qquad (2.38)$$

El ensamble de las mediciones perturbadas, con media igual a cero se puede almacenar en la matriz

$$\mathbf{E}_j = \left(\in_j^{(1)}, \in_j^{(2)}, ..., \in_j^{(N_r)}\right) \in \Re^{m_j \times N_r} \qquad (2.39)$$

con los cuales se puede construir la representación del ensamble de la matriz de covarianza del error de medición como

$$\mathbf{R}^{e}_{i(j)} = \frac{\mathbf{E}_j \mathbf{E}_j^T}{N_r - 1} \qquad (2.40)$$

Se puede demostrar que si cada miembro del ensamble se actualiza de manera independiente utilizando las observaciones perturbadas dadas en la ecuación (2.38), el ensamble actualizado tendrá la media y covarianza correctos (Evensen, 2009). Esto se puede representar de manera algebraica con las siguientes ecuaciones

$$\widetilde{\mathbf{B}}^{a}_{k} = \widetilde{\mathbf{B}}_k + \mathbf{H}^T \left[\mathbf{P}^e\right]\left(\mathbf{H}^T \left[\mathbf{H}\left[\mathbf{P}^e\right]\right] + \mathbf{R}^e_n\right)^{-1}\left(\mathbf{D} - \mathbf{H}\left[\widetilde{\mathbf{B}}_k\right]\right) \qquad (2.41)$$

donde $\widetilde{\mathbf{B}}_k$ se definió en la ecuación (2.34) y si m es el número de tiempos en los que hay mediciones entre el tiempo t_0 al tiempo t_k

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}_1 \\ \vdots \\ \mathbf{D}_m \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_m \end{pmatrix} \qquad (2.42)$$

y $$\mathbf{R}^e_k = \begin{pmatrix} \mathbf{R}^e_{i(1)} & & \\ & \ddots & \\ & & \mathbf{R}^e_{i(m)} \end{pmatrix} \qquad (2.43)$$

El número total de mediciones es $M = \sum_{j=1}^{m} m_j$. Por tanto, se tiene que $\mathbf{D} \in \Re^{M \times N_r}$, $\mathbf{H} \in \Re^{M}$ y $\mathbf{R}^e_k \in \Re^{M \times M}$. Se define el vector de innovación del ensamble como

$$\mathbf{D}' = \mathbf{D} - \mathbf{H}\left[\widetilde{\mathbf{B}}_k\right] \qquad (2.44)$$

también se definen las mediciones de las perturbaciones del ensamble $\mathbf{S} \in \Re^{M \times N_r}$ como

$$\mathbf{S} = \mathbf{H}\left[\widetilde{\mathbf{B}}'_k\right] \qquad (2.45)$$

y la matriz $\mathbf{P} \in \Re^{M \times M}$ se define como

$$\mathbf{P} = \mathbf{S}\mathbf{S}^T + (N_r - 1)\mathbf{R}^e_k \qquad (2.46)$$

Utilizando las ecuaciones (2.44) a (2.46) junto con las definiciones de las matrices de covarianza del error del ensamble dada en la ecuación (2.36) y (2.40), el análisis (dada en la ecuación 2.41) se puede expresar como

$$\begin{aligned}\tilde{\mathbf{B}}_k^a &= \tilde{\mathbf{B}}_k + \tilde{\mathbf{B}}'_k \mathbf{H}^T \left[\tilde{\mathbf{B}}'_k\right]\left(\mathbf{H}\left[\tilde{\mathbf{B}}'_k\right]\mathbf{H}^T\left[\tilde{\mathbf{B}}'_k\right] + (N_r - 1)\mathbf{P}^e\right)^{-1}\mathbf{D}' \\ &= \tilde{\mathbf{B}}_k + \tilde{\mathbf{B}}_k\left(\mathbf{I} - \mathbf{J}_N\right)\mathbf{S}^T\mathbf{P}^{-1}\mathbf{D}' \\ &= \tilde{\mathbf{B}}_k\left(\mathbf{I} + \left(\mathbf{I} - \mathbf{J}_N\right)\mathbf{S}^T\mathbf{P}^{-1}\mathbf{D}'\right) \\ &= \tilde{\mathbf{B}}_k\left(\mathbf{I} + \mathbf{S}^T\mathbf{P}^{-1}\mathbf{D}'\right)\end{aligned} \tag{2.47}$$

donde se utilizó la ecuación (2.32) y el término $\mathbf{J}_N\mathbf{S}^T \equiv 0$. De esta manera, la actualización del ensamble se puede considerar como una combinación de la predicción de los miembros del ensamble.

La diferencia entre *EnKF* y el *ES* se localiza en la rutina de actualización, debido a que el *EnKF* actualiza únicamente el estado actual del modelo, sin embargo, el *ES* incluye todos los estados previos del modelo, tomando en cuenta las mediciones del tiempo actual.

2.2.3.2 Ensamble Suavizado de Herrera (*ESH*)

Herrera (1998) desarrolló esta versión como un método de suavizado en el contexto de la optimización espaciotemporal de las redes de monitoreo del agua subterránea. A su método le llamó originalmente filtro de Kalman estático, luego filtro de Kalman ensamblado espaciotemporal y posteriormente Nowak (2010) lo llamó filtro de Kalman ensamblado estático.

En años recientes, este método se ha utilizado para la asimilación de datos y la estimación de parámetros en modelos de flujo y transporte del agua subterránea (Briseño-Ruíz, 2012).

Si el operador de mediciones fuera lineal, el *ESH* está dado por:

$$\hat{\mathbf{X}}^{n+1} = \hat{\mathbf{X}}^n + \mathbf{K}_{n+1}\left(\mathbf{Z}_{n+1} - \mathbf{H}_{n+1}\hat{\mathbf{X}}^n\right) \tag{2.48}$$

$$\mathbf{P}^{n+1} = \mathbf{P}^n - \mathbf{K}_{n+1}\mathbf{H}_{n+1}\mathbf{P}^n \tag{2.49}$$

$$\mathbf{K}_{n+1} = \mathbf{P}^n\mathbf{H}_{n+1}^T\left(\mathbf{H}_{n+1}\mathbf{P}^n\mathbf{H}_{n+}^T + \mathbf{R}_{n+1}\right)^{-1} \tag{2.50}$$

Este método se aplica de la misma forma que el filtro de Kalman ensamblado (mismas ecuaciones de corrección y mismos requerimientos), la diferencia estriba en:

a) Se aumenta el vector de estado (variable a estimar) para todos los tiempos de interés.
b) Usando las realizaciones del estado, se calculan la matriz de covarianza espaciotemporal del ensamble y el promedio espaciotemporal del ensamble para el período de interés, para que después de esta estimación y su matriz de covarianza del error sean corregidas utilizando las ecuaciones del filtro de Kalman.
c) No se utilizan las mediciones perturbadas.

Una de las diferencias que existe entre el *ESH* y el *ESLE*, es que en el *ESLE* se actualiza cada miembro del ensamble o realización de manera independiente y después realizan el promedio para obtener los momentos que se necesiten, aunque para ello se requiera perturbar las observaciones para obtener la matriz de covarianza correcta al promediar.

3. Conceptos básicos del cómputo en paralelo

El *GWMC* es un paquete que está escrito en *Fortran* y su versión original está escrita para ejecutarse en serie, la ejecución de este programa puede requerir muchas horas de uso continuo por lo que, para lograr su ejecución en un tiempo razonable, se puede recurrir a la paralelización del programa mediante la división de fragmentos que se pueden realizar de manera simultánea. En este capítulo se introducen algunos conceptos básicos del cómputo en paralelo que se utilizarán más adelante en este trabajo de tesis.

La metodología de paralelización a utilizar dependerá del problema que requiera de esta. De hecho, no todo problema puede ser paralelizado, porque hay problemas que son inherentemente secuenciales. Para lograr desarrollar la paralelización de un problema es necesaria una arquitectura paralela que permita ejecutar el programa y un lenguaje que permita reflejar el paralelismo del problema (García Carrasco, s/a).

Existen varias razones por las cuales se requiera resolver los problemas empleando paralelización, unas de las principales razones son: disminuir el tiempo total de ejecución de una aplicación, resolver problemas más complejos de grandes dimensiones, permitir la ejecución simultánea de tareas. Algunas razones secundarias podrían ser: obtener ventajas de los recursos no locales, disminuir costos, sobrepasar los límites de almacenamiento en memoria y disco; los límites físicos de computación de una máquina secuencial, que actualmente ya están en la frontera de lo que puede ser practicable en términos de velocidad interna de transmisión de datos y de velocidad de *CPU* (Giménez et al., s/a).

La paralelización es muy útil en el cómputo científico para superar algunas de las limitaciones impuestas por un solo *CPU* en un ordenador. Además, ofrece soluciones rápidas y por lo tanto las aplicaciones que se han paralelizado pueden resolver problemas más grandes y complejos, cuyos datos de entrada o resultados intermedios exceden la capacidad de la memoria de un solo *CPU*. De esta manera, las simulaciones se pueden ejecutar en la resolución más fina, y los fenómenos físicos pueden representar el modelo más realista.

En la práctica, diseñar programas paralelos no es simple y el aprendizaje de técnicas avanzadas requiere de mayor esfuerzo y tiempo. Además, con frecuencia un programa hecho para ejecutarse

en un solo procesador es difícil de modificarse para que aproveche las diferentes arquitecturas de los multiprocesadores existentes hoy en día.

3.1 Herramientas de cómputo en paralelo

Algunas herramientas de software que utilizamos en este trabajo para llevar a cabo dicha paralelización son: el código en *Fortran* y *Python* se implementa en el sistema operativo *Linux*, y para la paralelización se usa *MPI.* A continuación, se da una breve explicación de este software.

Fortran es el más antiguo de los lenguajes de alto nivel, se utiliza principalmente en aplicaciones científicas y análisis numérico. Inicialmente este lenguaje sólo usaba un paradigma estructurado (en bloques independientes de procedimientos, funciones que se controlaban mediantes estructuras de control), pero las últimas versiones incluyen programación orientada a objetos. El lenguaje ha sido ampliamente adoptado por la comunidad científica para escribir aplicaciones con cómputos intensivos (Muller Santa Cruz, 2007).

Python es un intérprete, que usa lenguaje de programación interactivo y extensible, además utiliza una amplia variedad de aplicaciones. En particular, para la computación científica, existen muchas herramientas que facilitan el desarrollo de códigos computacionales (Jones et al., 2001). *Python* se puede combinar fácilmente con otros lenguajes de programación, como *C*, *C++* y *Fortran*, también se puede utilizar para explotar las arquitecturas de cómputo de alto rendimiento mediante el uso de *MPI* (*Message Passing Interface*; Gropp et al., 1999b) o *CUDA* (Compute Unified Device Arquitectura; Kirk y Hwu, 2010). En la actualidad, casi cualquier sistema operativo es compatible con *Python* de manera tal que este lenguaje de programación proporciona portabilidad a través de muchas plataformas de computación.

Linux es uno de los sistemas operativos más fiables, robustos, estables y eficientes que hay en la actualidad. Este sistema operativo fue creado por Linus Torvalds a finales de 1991 con la idea de crear un *UNIX* para *PC* que cualquier persona pudiera utilizar en su ordenador, es de licencia pública general (*GNU*), al cual contribuyeron programadores de todo el mundo (Stutz, 2001).

Tareas que se pueden descomponer fácilmente en componentes que se pueden realizar de manera independiente se conocen como *embarrassingly parallel*, *perfectly parallel* o *pleasingly parallel* (Ridgway, Terry y Babak, 2005; Herlihy y Shavit, 2008; Gunarathne et al., 2011).

3.2 Tipos de paralelismo

Existen tres formas de paralelizar un programa, estas son:

- **Paralelización de grano fino:** se le conoce también con el nombre de paralelismo de instrucción. Cada procesador realizará una parte de cada paso del algoritmo, es decir, se podrán ejecutar en paralelo las instrucciones individuales del algoritmo (García Regis y Cruz Martínez, 2003; Hidalgo Pérez y Cervigón Rückauer, 2004).
- **Paralelización de grano medio:** también se le conoce con el nombre de paralelismo a nivel de bucle o a nivel de procedimiento (Madrid, s/a). Usualmente este tipo de paralelización se realiza de manera automática en los compiladores (Hidalgo Pérez y Cervigón Rückauer, 2004).
- **Paralelización de grano grueso:** se le conoce también con el nombre de paralelismo de tareas. Se realiza cuando hay tareas independientes y se pueden ejecutar como procesos independientes en diferentes procesadores, comúnmente es aplicable en esquemas maestro/esclavo, cliente/servidor, productor/consumidor (Madrid, s/a). Este tipo de paralelización se puede utilizar fácilmente para tareas *pleasingly parallel.*

En este trabajo se utilizó una paralelización de grano grueso.

3.3 Métricas de rendimiento

En esta sección se mencionan algunas métricas de rendimiento que son útiles para analizar la robustez de la paralelización de un programa (Jiménez-González, s/a). Las métricas más utilizadas para calcular la utilidad de un algoritmo paralelo son la aceleración y la eficiencia, que se describen en la siguiente subsección.

3.3.1 Aceleración y eficiencia

La aceleración (S_p) se define como:

$$S_p = \frac{T_1}{T_N} \qquad (3.1)$$

Donde T_1 es el tiempo de ejecución del algoritmo en un procesador y T_N es el tiempo de ejecución del algoritmo en N procesadores.

La eficiencia (E_p) se define como:

$$E_p = \frac{S_p}{N} \qquad (3.2)$$

Donde N es el número de procesadores en los que se lleva a cabo la ejecución del programa y S_p es la aceleración. El valor máximo que puede alcanzar la eficiencia es el 1, que representa el 100% de aprovechamiento.

Estas métricas las utilizaremos para verificar que tan eficiente es la paralelización del *GWMC*.

3.3.2 Ley de Amdahl

Por lo general, para cualquier programa, hay una parte que no se puede paralelizar, y que debe ejecutarse de forma secuencial, lo que provoca que se vea limitada la mejora del rendimiento que se obtendría al paralelizar, y en consecuencia, la eficiencia que se obtenga. Se recurre entonces a la ley de Amdahl, que nos indica que la mejora del rendimiento viene limitada por una fracción de tiempo, donde el programa es secuencial, cuando el programa se está ejecutando en paralelo (Jiménez-González, s/a). La ley de Amdahl es la siguiente:

$$S_p \leq \frac{1}{f + (1-f)/p} \qquad (3.3)$$

Donde f representa la fracción secuencial del programa y p es el número de procesadores.

3.3.3 Costo

Para resolver un problema en paralelo, es útil conocer el costo computacional que tendrá la solución del trabajo. El costo se define como el tiempo de ejecución del programa en paralelo por el número de procesadores que se utilizan, esto es:

$$C = NT_N \qquad (3.4)$$

donde C es el costo, N el número de procesadores y T_N el tiempo de ejecución del algoritmo en paralelo.

3.4 Memoria

En la actualidad, las computadoras cuentan con un espacio de almacenamiento en que se guardan programas, datos, etc.; a este espacio se le llama memoria.

La memoria física, también conocida como memoria principal o memoria *RAM* (*Random Access Memory*), también se le conoce como memoria volátil, debido a que los datos no se guardan de manera permanente (RAM, 2014).

En la mayoría de los sistemas actuales, la velocidad de la memoria principal o *RAM* está entre la velocidad del disco duro y la memoria caché.

La memoria caché es aquella área de almacenamiento que funciona de manera semejante a la memoria principal, aunque que es de menor tamaño y acceso mucho más rápido (Caché, s.f.).

La memoria principal está constituida por un conjunto de celdas, y en cada una de estas se puede almacenar información, además las celdas cuentan con un número (llamada dirección) que la unidad de control conoce. La memoria principal se puede percibir como un módulo en el cual recibe una dirección, una orden de escritura o lectura y ésta la almacena o entrega la información a la celda o dirección correspondiente.

Existen diferentes tipos de memorias *RAM*, las básicas son:

DRAM (*Dynamic Random Access Memory*): es un tipo de memoria con gran capacidad, que tiene que ser constantemente actualizada varias veces por segundo (b_diaz, s.f.).

SRAM (*Static Random Access Memory*): es un tipo de memoria que no tiene la necesidad de actualizarse como la memoria *DRAM*, además de ser capaz de mantener los datos. Este tipo de memoria suele ser más cara, aunque más rápida y con un consumo eléctrico menor que la memoria *DRAM* (b_diaz, s.f.).

La memoria virtual también conocida como memoria swap (traducida al español como intercambio) es aquella zona, partición o espacio en el disco duro, la cual estará destinada como espacio de intercambio (Swap, 2018), y se usará como si fuera memoria *RAM*; ésta se utiliza para almacenar datos temporales y reducir el uso de la *RAM*. Combinando la memoria *RAM* y la memoria swap se crea una memoria virtual mayor a la memoria principal o *RAM*.

La velocidad de la memoria swap comparada con la velocidad de la memoria principal o *RAM* es mucho más lenta, sin embargo, está disponible cuando el sistema lo requiera. Algunas ventajas de la memoria swap son: le permite al sistema ejecutar una cantidad mayor de programas, permite optimizar el uso de memoria, permite que un proceso sea más grande que toda la memoria

principal. Sin embargo, las desventajas de la memoria swap es el tiempo que requiere para hacer las copias a la memoria del disco duro, haciéndolo más costoso por el tiempo de copiado y lectura de los procesos, prolongando los tiempos de ejecución del programa, que se le conoce como "thrashing[1]"; para solucionar este tipo de problema es incrementar la memoria, sino renovar el diseño de las aplicaciones y con ello disminuir el consumo de memoria swap.

[1] "Fenómeno que se produce en los sistemas con memoria virtual, cuando el procesador dedica todo su tiempo a tareas de paginación, debido a que la cantidad de memoria real es pequeña comparada con el conjunto de procesos activos. En esta situación la tasa de paginación se eleva y los procesos se bloquean esperando la transferencia de páginas con lo que el rendimiento del sistema se reduce drásticamente." (S.A., (sf)).

4. Paralelización de Groundwater Monte Carlo (*GWMC*)

Como se comentó anteriormente, en esta tesis se paralelizó el código del paquete *GWMC.* Este está escrito en Fortran y fue desarrollado por Herrera (1998), y posteriormente modificado por Olivares-Vázquez (2002) y Briseño-Ruíz (2012). Se diseñó originalmente como la componente que aplica la simulación Monte Carlo del paquete *GWQMonitor* (Groundwater Quality Monitor) para el diseño óptimo de redes de monitoreo de la calidad de agua subterránea (Herrera, 1998).

4.1 Implementación del GWMC en serie

El *GWMC* implementa la simulación Monte Carlo usando un simulador de flujo y transporte en el que la conductividad hidráulica y la concentración en la fuente del contaminante se modelan como parámetros aleatorios. Estos se consideran aleatorios porque suelen ser parámetros muy variables, de los que cuenta con pocos datos y que se estiman de manera indirecta, por lo que suelen tener mucha incertidumbre. Por tanto, se obtienen múltiples realizaciones de esos dos parámetros y para cada realización, las ecuaciones de flujo y transporte se resuelven por medio del simulador a utilizar, ya sea el Princeton Transport Code (*PTC;* Babu et al., 1993) para ambos o bien, el *MODFLOW* (Mc Donald y Harbaugh, 1998) para flujo y el *MT3D* (Zheng, 1992) para transporte. Finalmente, las soluciones de concentración para cada par de realizaciones de conductividad hidráulica y de concentración en la fuente de contaminante se promedian para obtener su media y matriz de covarianza espaciotemporal.

En la implementación del *GWMC* se supone que el campo de conductividad hidráulica tiene una distribución lognormal (esto es, que ln(K) tiene una distribución normal), como se ha constatado con frecuencia al analizar datos de esta variable en diferentes publicaciones (ver, por ejemplo, Lynn W., 1993). Para obtener realizaciones de esta variable se utiliza el programa *Sgsim* (Deutsch y Journel, 1998) que genera campos aleatorios gaussianos y con ayuda del programa *Nrm2log* estas se transforman a un campo con distribución lognormal con media igual a la del modelo determinista y varianza dada (se puede ver la fórmula que se aplica en la pag. 20 de Lynn W., 1993). Estas realizaciones de *K* se utilizan como datos de entrada en el *GWMC*. Por otro lado, la concentración del contaminante en la fuente se modela con series de tiempo con media igual a la del modelo determinista y una varianza dada, esta serie se genera con ayuda del programa *RandT2*. El *Sgsim* utiliza una malla rectangular regular que cubre toda el área modelada y si el

simulador a utilizar usa una malla diferente a la de *Sgsim*, entonces se realiza un mapeo de cada nodo de la malla del simulador al nodo más cercano de la malla utilizada por *Sgsim*.

Por otro lado, se deben especificar posiciones y tiempos de muestreo y de estimación, las posiciones y tiempos de muestreo son en los que se tienen datos que se quieren asimilar y, los de muestreo son en los que se requiere obtener estimaciones de concentración.

A continuación, se explica con más detalle el paquete *GWMC* cuando se utiliza PTC para la simulación de flujo y transporte, refiriéndonos a este como *GWMC-PTC*, y cuando se utilizan *MODFLOW/MT3D*, al que nos referiremos como *GWMC-MFMT*.

4.1.1 GWMC – PTC

Como se comentó antes, el *GWMC* tiene la posibilidad de trabajar con concentraciones aleatorias en las fuentes del contaminante. Estas se generan a partir de los valores originales en los nodos de las mismas en el modelo determinista de *PTC*, que se perturban usando el software *RandTS2,* dando como resultado una serie temporal aleatoria; además el *GWMC* reemplaza la conductividad hidráulica (*K*) original en el modelo *PTC* con las realizaciones generadas por la combinación de dos softwares, primero el *Sgsim* que obtiene realizaciones normales con media cero y varianza 1, que son transformadas por el software *Nrm2log* en un campo lognormal con una media igual a la conductividad del modelo determinista en el nodo y una varianza dada. En la sección 5.1 se presenta con más detalle un ejemplo de la configuración de esto para un caso de estudio.

El paquete *GWMC-PTC* está conformado por el software que se muestra en el esquema de la figura 4.1 y que se describe a continuación:

- ✓ ***RandTS2****:* se utiliza de manera externa para crear series de tiempo aleatorias que se asocian con los nodos de las fuentes del contaminante. El *GWMC* utiliza estas series para asignar las concentraciones en los nodos de las fuentes del contaminante a partir de los valores originales en el modelo determinista de *PTC*.
- ✓ ***Sgsim:*** es el Simulador Secuencial Gaussiano (Sequential Gaussian Simulator) del *GSLIB* (Deutsch y Journel, 1998). Se usa externamente para crear las realizaciones de un campo aleatorio gaussiano estándar.
- ✓ ***Nrm2log:*** transforma las realizaciones obtenidas por *Sgsim* a realizaciones del campo lognormal de la conductividad hidráulica (*K)*. El *GWMC* sustituye la *K* del modelo determinista de *PTC* por las realizaciones generadas durante la simulación estocástica.

- ✓ ***PTC*** (Babu et al., 1993); se utiliza un modelo determinista ya configurado según los requerimientos de *PTC*, por tanto, los archivos iniciales que éste requiere ya existen. El simulador se corre repetidas veces al ejecutarse *GWMC mediante llamadas al sistema operativo*.

Archivos de entrada para *PTC*: archivos del dominio y la malla; elevaciones de las capas; los parámetros requeridos (al menos conductividades hidráulicas en las tres direcciones y porosidad); las condiciones de frontera para los modelos de flujo y transporte. Archivos de entrada para *GWMC*: archivos de malla de PTC y de especificaciones de la malla de sgsim; los valores transformados del *Sgsim y Nrm2log*; los nodos y la concentración de solutos asignada en las fuentes del contaminante; número de tiempos y posiciones de estimación, capas aleatorias, coordenadas y tiempos de muestreo; y concentraciones resultantes de la corrida de PTC.

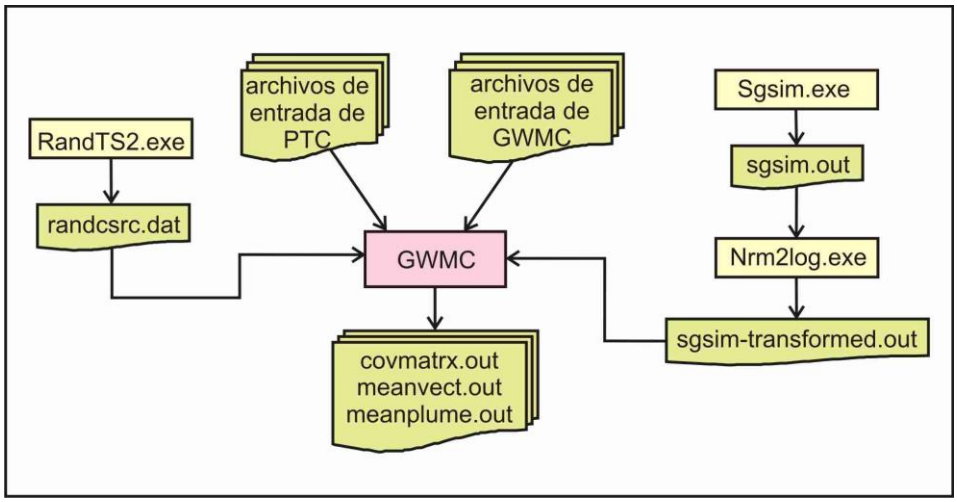


Figura 4. 1 Esquema del software que se requiere para ejecutar el *GWMC*, utilizando el simulador *PTC*.

Los archivos de salida son: matriz de covariancia espaciotemporal de concentración (covmatrx.out en la Figura 4.1); la media de la pluma del contaminante en una capa preasignada por el usuario (meanplume.out en la Figura 4.1); y la media espaciotemporal de concentración (meanvect.out en la Figura 4.1).

En la figura 4.2 se muestra el diagrama de flujo del *GWMC* en serie, utilizando el simulador *PTC*. Los pasos para ejecutar el *GWMC* son los siguientes:

Paso 1. Se generan los archivos de entrada para *PTC.*

Paso 2. Se generan los archivos de entrada para *GWMC*.

Paso 3. Se generan un número n de realizaciones de un campo normal estándar utilizando el programa de Simulación Secuencial Gaussiana (*Sgsim*; Deutsch y Journel, 1998) y se escriben en un archivo.

Paso 4. Se leen los archivos de salida de Sgsim, se calcula una transformación a un campo lognormal con media y varianza dada (las conductividades hidráulicas) para cada realización utilizando el programa *Nrm2log* y se escriben en un archivo.

Paso 5. Se genera el mismo número de realizaciones de series temporales para cada nodo de la fuente del contaminante con media igual a la del modelo determinista y varianza dada usando el programa *RandTS2* y se escriben en un archivo.

Paso 6. Se lee el archivo de entrada de *Sgsim* y a cada nodo de la malla de *PTC* se le asigna el nodo más cercano de la malla rectangular de *Sgsim*.

Paso 7. Se lee el archivo de salida de *Nrm2log*, el valor de la conductividad hidráulica generado por *Nrm2log* se asigna al nodo correspondiente de la malla en *PTC* y en los archivos de conductividad hidráulica de *PTC* se sustituyen los valores originales por los nuevos valores.

Paso 8. Se lee el archivo de salida de RandTS2, en el archivo de *PTC* para las condiciones de frontera del transporte las concentraciones del contaminante en los nodos de la fuente se sustituyen por los valores generados por *RandTS2*.

Paso 9. El *PTC* se ejecuta externamente para cada realización de conductividad y concentración en la fuente (*PTC* resuelve numéricamente la ecuación de flujo y transporte) y se escriben los resultados en archivos.

Paso 10. Se leen los archivos de concentraciones de salida de PTC obtenidas en el *paso 8*, se calculan dos vectores auxiliares, uno con la suma de las concentraciones para todos los nodos de una capa del modelo (al que nos referiremos como vector auxiliar de la pluma media), y otro con la suma de las concentraciones para cada posición espaciotemporal de muestreo y de estimación especificados por el usuario (al que nos referiremos como vector auxiliar medio), y también se calcula una matriz auxiliar con la suma de los productos de concentraciones para cada posible par de estas posiciones espaciotemporales (a la que nos referiremos como matriz de covarianza auxiliar).

Paso 11. Combinando la información contenida en el vector auxiliar medio y la matriz de covarianza auxiliar, se calcula la media y la matriz de covarianza espaciotemporal de la concentración del contaminante, y también se calcula la media de la pluma del contaminante en una capa preasignada. Se escribe un archivo de salida para cada uno de estos vectores y para la matriz de covarianza.

Paso 12. El *ESH* se aplica para estimar la concentración del contaminante.

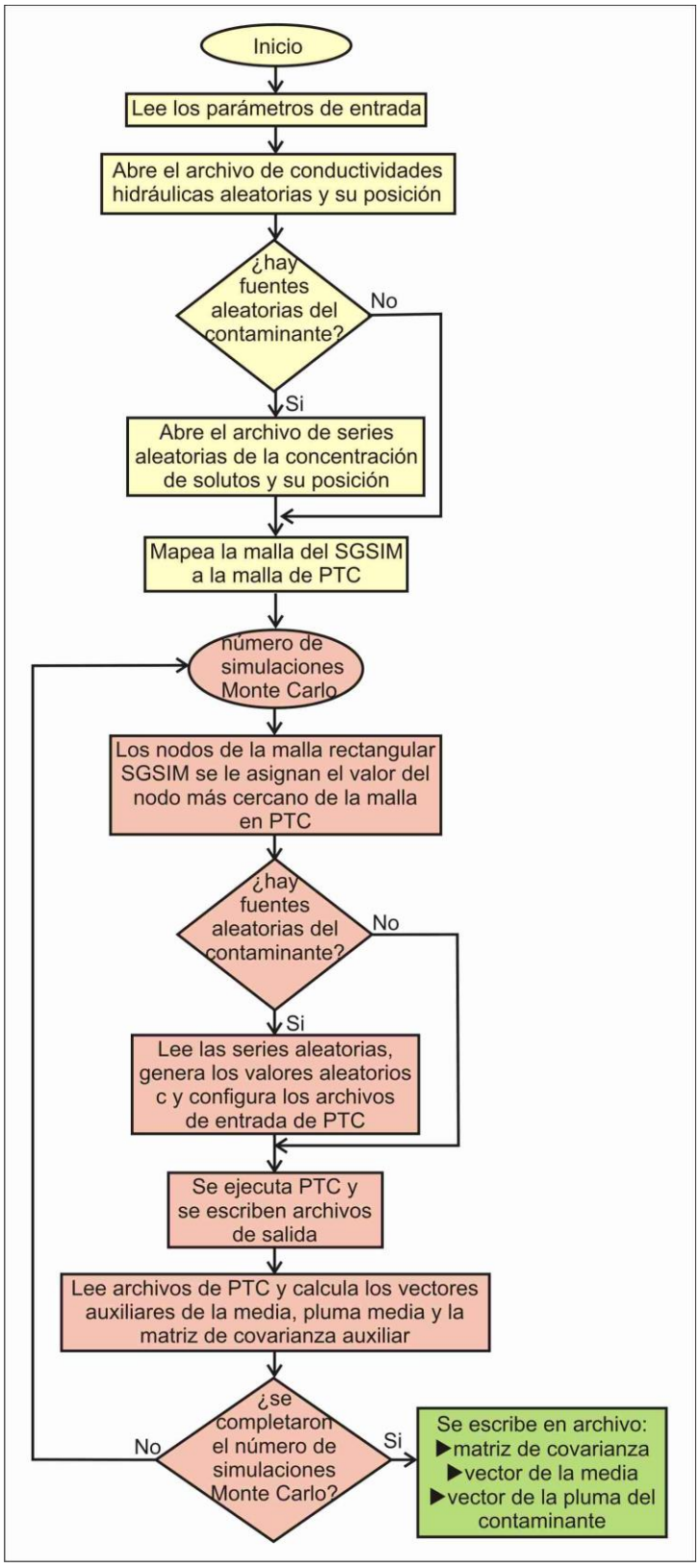


Figura 4. 2 Diagrama de flujo del *GWMC-PTC* en serie utilizando el simulador *PTC* (modificada de Olivares, 2002).

El diagrama de flujo se explica a continuación en dos partes, la primera está indicada en color amarillo y la segunda parte en color rosado.

Al iniciar la ejecución del programa *GWMC-PTC* (figura 4.3), se leen los parámetros de entrada, el programa tiene una subrutina llamada *readparm* que lee todos los parámetros que se requieren, después se verifica si existen fuentes aleatorias, si existen se llama a las subrutinas *getncnode* y *getrndcnodes*, las cuales obtienen el número de nodos con las condiciones de frontera del modelo de transporte en *PTC* y obtienen los nodos que serán aleatorios. Posteriormente se llama a las subrutinas *allocate_space* y *alloc_space_covmatx*, que asignan a los arreglos (vectores y matrices) el tamaño determinado por el usuario durante la ejecución.

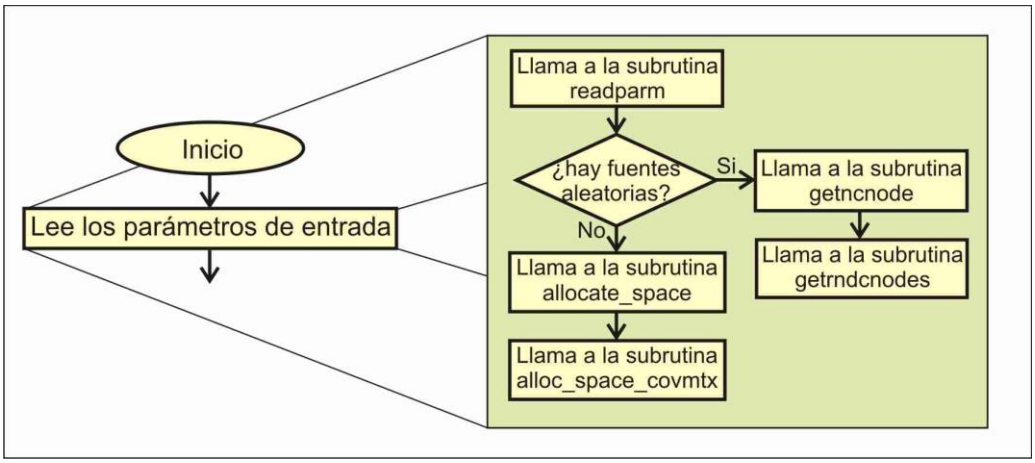


Figura 4. 3 Diagrama de la subrutina readparm del programa *GWMC-PTC*.

Cuando se pasa al siguiente procedimiento (figura 4.4), indicado como "abre el archivo aleatorio de conductividades hidráulicas y su posición", primero se crean los archivos *errorlog* (es el que archivara todos los errores que ocurran en la ejecución del programa y en consecuencia, se aborte la ejecución) y *coorslog* (en este archivo se guardan las coordenadas de los nodos, el número de nodos en cada dirección de la malla de sgsim y el número de nodo del nodo más cercano de esta malla al nodo de la malla de simulación), luego se abre el archivo generado por el programa *Nrm2log*, llamado en la figura 4.1 *sgsim-transformed.out* y se posiciona en el reglón de la primera realización aleatoria para leer los valores del archivo.

Después el programa verifica si hay fuentes aleatorias del contaminante (ver figura 4.5), si existen fuentes aleatorias entonces se abre el archivo generado por el *RandTS2* llamado en la figura 4.1

randcsrc.dat y lo lee, también abre el archivo de condiciones de frontera de *PTC*, *bctran.dat*, que tiene las posiciones y las condiciones de frontera.

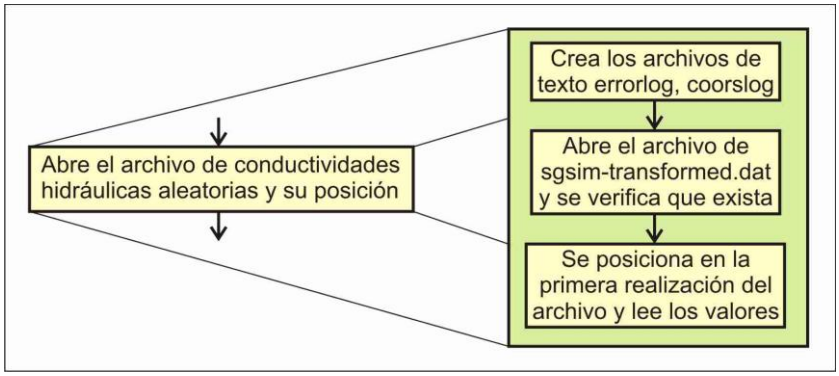


Figura 4. 4 Procedimiento del posicionamiento de las conductividades hidráulicas.

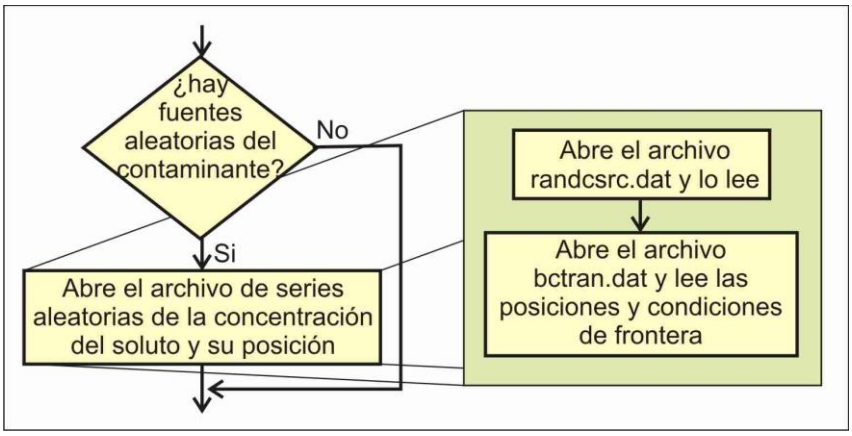


Figura 4. 5 Diagrama del posicionamiento de las series aleatorias del contaminante.

Entonces continúa hacia el mapeo de la malla del *Sgsim* a la malla de *PTC* (ver figura 4.6), en el que a cada nodo de la malla de *PTC* se le asigna el nodo más cercano de la malla rectangular de *Sgsim* (es importante notar que los nodos de la malla de *Sgsim* corresponden a los centros de las celdas). Aquí se llama a la subrutina *meshes* que lee la malla de simulación del modelo de *PTC*, y si la malla de simulación de *PTC* no es regular, crea la malla regular usada por *Sgsim*. Dentro de la subrutina *meshes*, se llama a la subrutina *gnrecc1* que genera las coordenadas de la malla regular, después se llama a la subrutina *readmesh* que lee la geometría de la malla del archivo del modelo de *PTC*.

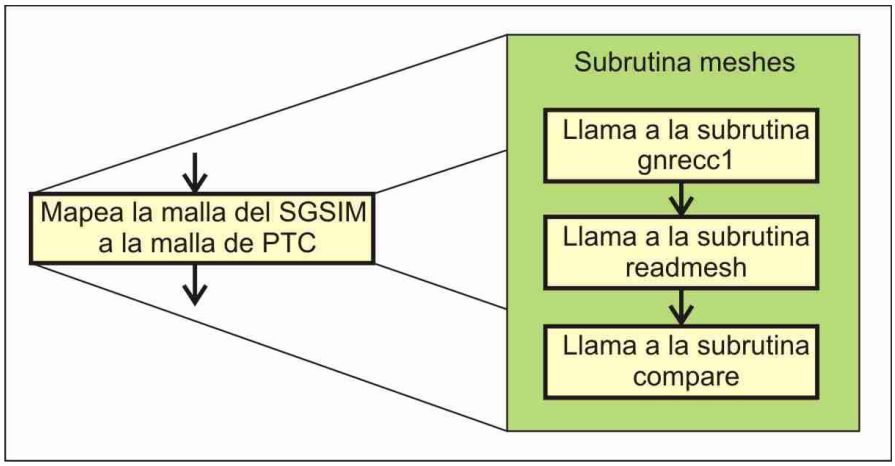


Figura 4. 6 Diagrama de la subrutina meshes del programa *GWMC-PTC*.

La subrutina *compare* es la que se encarga de asignar a cada nodo de la malla de PTC un nodo de la malla regular de Sgsim. Lo primero que hace es verificar que cada nodo de la malla de PTC esté dentro del dominio que abarca la malla de Sgsim y, si no es así, manda un mensaje de error, en caso contrario localiza el nodo de la malla de Sgsim más cercano al nodo de PTC y asocia sus coordenadas con este. Ya que se realiza este mapeo, empieza el ciclo en el que se hacen las simulaciones Monte Carlo (figura 4.7). Primero se asigna a cada nodo de la malla de *PTC* la conductividad generada por *Nrm2log* correspondiente al nodo que se le asoció de la malla de Sgsim.

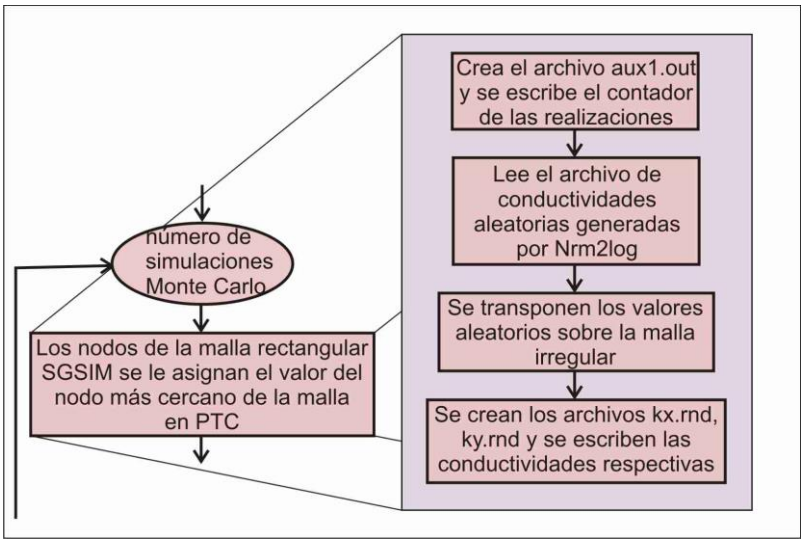


Figura 4. 7 Diagrama del ciclo de la simulación Monte Carlo.

Después se verifica si hay fuentes aleatorias de contaminante (figura 4.8); si hay, entonces se procede a generar las perturbaciones en las fuentes de concentración de contaminante, es decir, se calcula una realización para las fuentes de concentración a partir de la serie aleatoria que se genera en la subrutina *randsrc* que transforma las fuentes del contaminante del modelo *PTC* en fuentes de contaminante aleatorio con media igual al valor determinista y varianza dada.

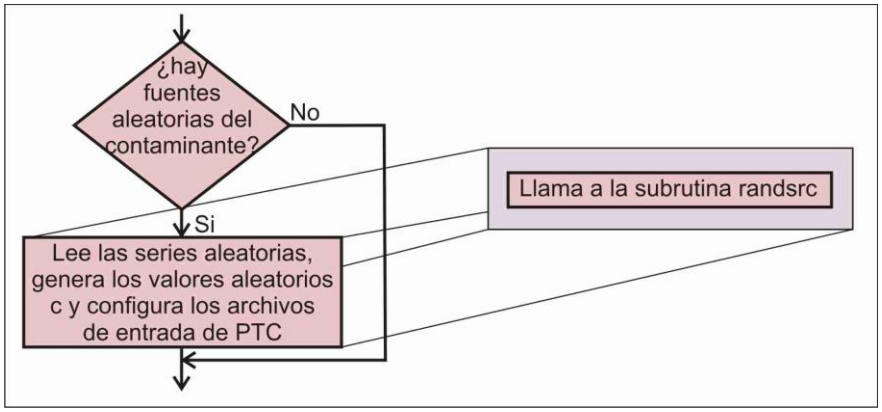


Figura 4. 8 Diagrama de la subrutina *randsrc* del programa *GWMC*.

Acto seguido, se ejecuta el simulador *PTC* que resuelve las ecuaciones de flujo y transporte con las nuevas conductividades y concentraciones en la fuente de contaminante (figura 4.9) y los resultados de las concentraciones se escriben en un archivo. Se llama a la subrutina *covmatrx*, lee el archivo de de salida de concentraciones de *PTC*, con estos valores se calculan la matriz de covarianza auxiliar, el vector auxiliar de la media, el vector auxiliar de la pluma del contaminante intermedio en el conjunto de posiciones de muestreo y de estimación dados. Al llegar al número de simulaciones Monte Carlo que el usuario fijó al principio, el programa finaliza calculando la matriz de covarianza de las concentraciones, el vector medio de las concentraciones y el vector medio de la pluma del contaminante en una capa específica del problema que se está tratando, y escribiendo un archivo de salida para cada uno (llamados covmatrx.out, meanvect.out y meanplume.out, respectivamente, en la Figura 4.1).

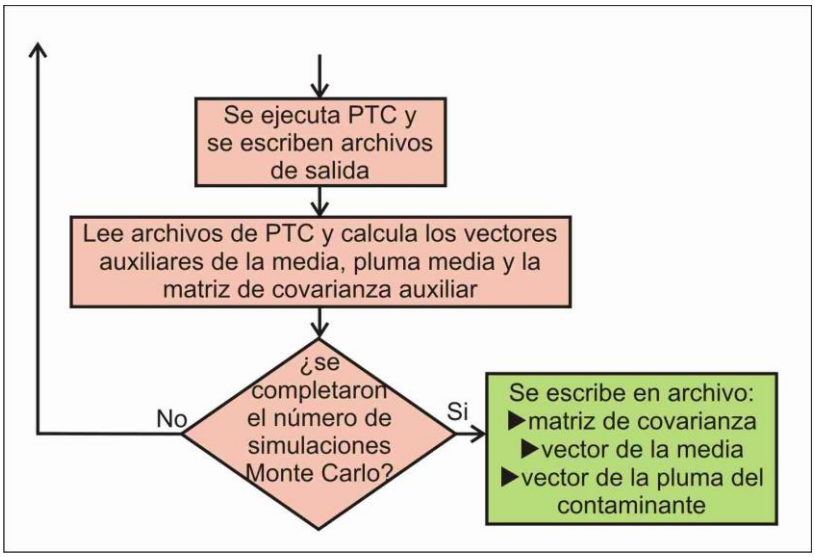


Figura 4. 9 Diagrama de la ejecución del simulador *PTC* del programa *GWMC*.

4.1.2 GWMC - MFMT

Para utilizar los simuladores *MODFLOW/MT3D*, el paquete *GWMC* se tuvo que modificar un poco, los programas *Nrm2log* y *RandTS2* explicados en la subsección anterior, se incluyeron dentro del *GWMC* como subrutinas, debido a que *MODFLOW* y *MT3D* trabajan de diferente manera que el *PTC*. En la figura 4.10 se muestra un esquema del software que se utiliza para ejecutarse el *GWMC.* Al igual que antes, partimos de un modelo en *MODFLOW*, para el que los archivos iniciales ya se han creado, así como el archivo indicando las posiciones y los tiempos de muestreo y de estimación. En este caso al inicio únicamente se emplea el *Sgsim* de manera externa, de la misma manera que se explicó en la subsección anterior. En la sección 5.2 se presenta un ejemplo usando GWMC - MFMT para un caso de estudio explicado con más detalle.

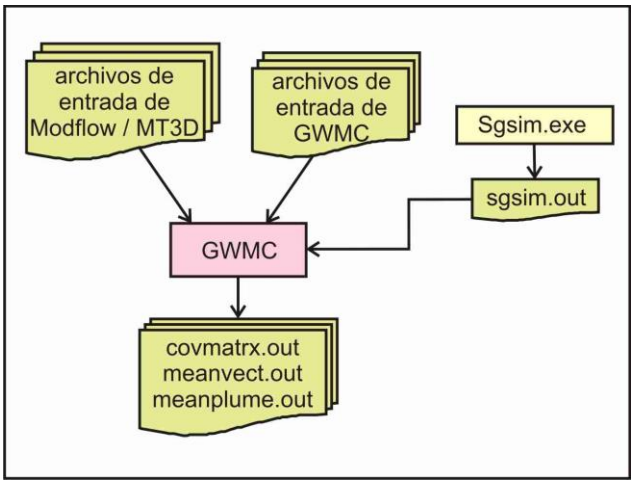


Figura 4. 10 Esquema del software que se requiere para ejecutarse el *GWMC*, utilizando el simulador *MODFLOW/MT3D*.

En diagrama de flujo del *GWMC* en serie, utilizando el simulador *MODFLOW/MT3D* es muy similar al que se ocupa el simulador *PTC* y la metodología es muy parecida a la que se presenta en la subsección anterior. Algunas modificaciones que se realizaron al *GWMC* para utilizar el simulador *MODFLOW/MT3D* fue agregar como subrutinas a los programas *Nrm2log* y *RandTS2*, y adecuarlas para que *MODFLOW* y *MT3D* se pudieran ejecutar.

4.2 Estrategia de paralelización del GWMC

Como se puede ver en la explicación anterior, la simulación Monte Carlo es del tipo *pleasingly parallel*, ya que son independientes las corridas de los modelos para cada juego de parámetros obtenidos con Sgsim y RandT2. Entonces, la estrategia que se utilizó para realizar la paralelización del *GWMC* fue hacer un preprocesamiento en serie, luego correr los modelos de flujo y transporte para cada juego de parámetros en un procesador diferente y, finalmente, construir en un solo procesador el vector medio, la matriz de covarianza y la estimación de la media de la pluma del contaminante en una capa preasignada. Con respecto a la terminología introducida en las Sección anterior, esta es una paralelización de grano grueso.

La metodología utilizada para llevar a cabo esto consistió en poner en un procesador maestro, a través de un servidor cliente, los archivos de entrada de los programas Sgsim, RandT2, GWMC y los del simulador a usar (PTC o MODFLOW/MT3D), y el procesador maestro, mediante un script escrito en Python en el que se especifican el número total de realización a ejecutar (nr), divide este número entre número de procesadores (p), modifica los archivos de entrada de Sgsim y RandT2, especificando el número de realizaciones (parte entera de nr/p) que llevará a cabo cada

procesador y copia estos archivos a cada procesador. Cada procesador esclavo y el maestro genera n/p realizaciones de conductividades y el mismo número de realizaciones de concentraciones en la fuente del contaminante (si se requieren), realiza n/p simulaciones de flujo y transporte para obtener los datos de salida de concentraciones y calcula el vector auxiliar medio, la matriz de covarianza auxiliar y el vector auxiliar de la pluma media (pasos del 3 al 10 explicados en la sección anterior). Cuando los procesadores terminan los cálculos designados, regresan el vector auxiliar medio, la matriz de covarianza auxiliar y el vector auxiliar de la pluma media al procesador maestro al procesador maestro y este realiza el cálculo de la matriz de covarianza, del vector de la media y del vector de la pluma media (paso 11 explicado en la sección anterior).

Los pasos utilizados para realizar la paralelización del *GWMC* se enlistan a continuación y se puede observar en la figura 4.11.

Paso A. Se generan los archivos de entrada para *PTC.*

Paso B. Se generan los archivos de entrada para *GWMC*.

Paso C. El usuario a través de un servidor cliente copia los archivos generados en el paso 1 y 2.

Paso D. El procesador maestro divide el número total de realizaciones (nr) entre el número de procesadores (p), modifica los archivos de entrada de Sgsim y de RandT2, y copia estos archivos a cada procesador.

Paso E. Se generan las nr/p realizaciones utilizando el programa de Simulación Secuencial Gaussiana (*Sgsim*; Deutsch y Journel, 1998) y se escriben en un archivo.

Paso F. Se leen los archivos de salida de Sgsim, se calcula las conductividades hidráulicas para cada realización utilizando el programa *Nrm2log* y se escriben en un archivo.

Paso G. Se generan las realizaciones para cada nodo de la fuente del contaminante usando el programa *RandTS2* y se escriben en un archivo.

Paso H. Se lee el archivo de entrada de Sgsim y se mapea la malla de *PTC* a la de *Sgsim*.

Paso I. Se lee el archivo de salida de *Nrm2log*, el valor de la conductividad hidráulica generado por *Nrm2log* se asigna al nodo correspondiente de la malla en *PTC* y en los archivos de conductividad hidráulica de *PTC* se sustituyen los valores originales por los nuevos valores.

Paso J. Se lee el archivo de salida de *RandTS2*, en el archivo de *PTC* para las condiciones de frontera del transporte las concentraciones del contaminante en los nodos de la fuente se sustituyen por los valores generados por *RandTS2*.

Paso K. El *PTC* se ejecuta externamente para cada realización de conductividad y concentración en la fuente y se escriben los resultados en archivos.

Paso L. Se leen los archivos de concentraciones de salida de PTC obtenidas en el *paso H*, se calculan el vector auxiliar de la pluma media, el vector auxiliar medio y la matriz de covarianza auxiliar, y se escriben en disco.

Paso M. Se leen los vectores y matriz auxiliar de los archivos generados en el paso L, combinando la información contenida en estos se calcula la media y la matriz de covarianza espaciotemporal de la concentración del contaminante, y la media de la pluma del contaminante. Se escribe un archivo de salida para cada uno de estos vectores y para la matriz de covarianza.

Paso N. El *ESH* se aplica para estimar la concentración del contaminante.

La parte que se realiza en paralelo son los pasos del E al L, cabe mencionar que dentro de este proceso en paralelo está incluido el programa *GWMC* (pasos del H al L).

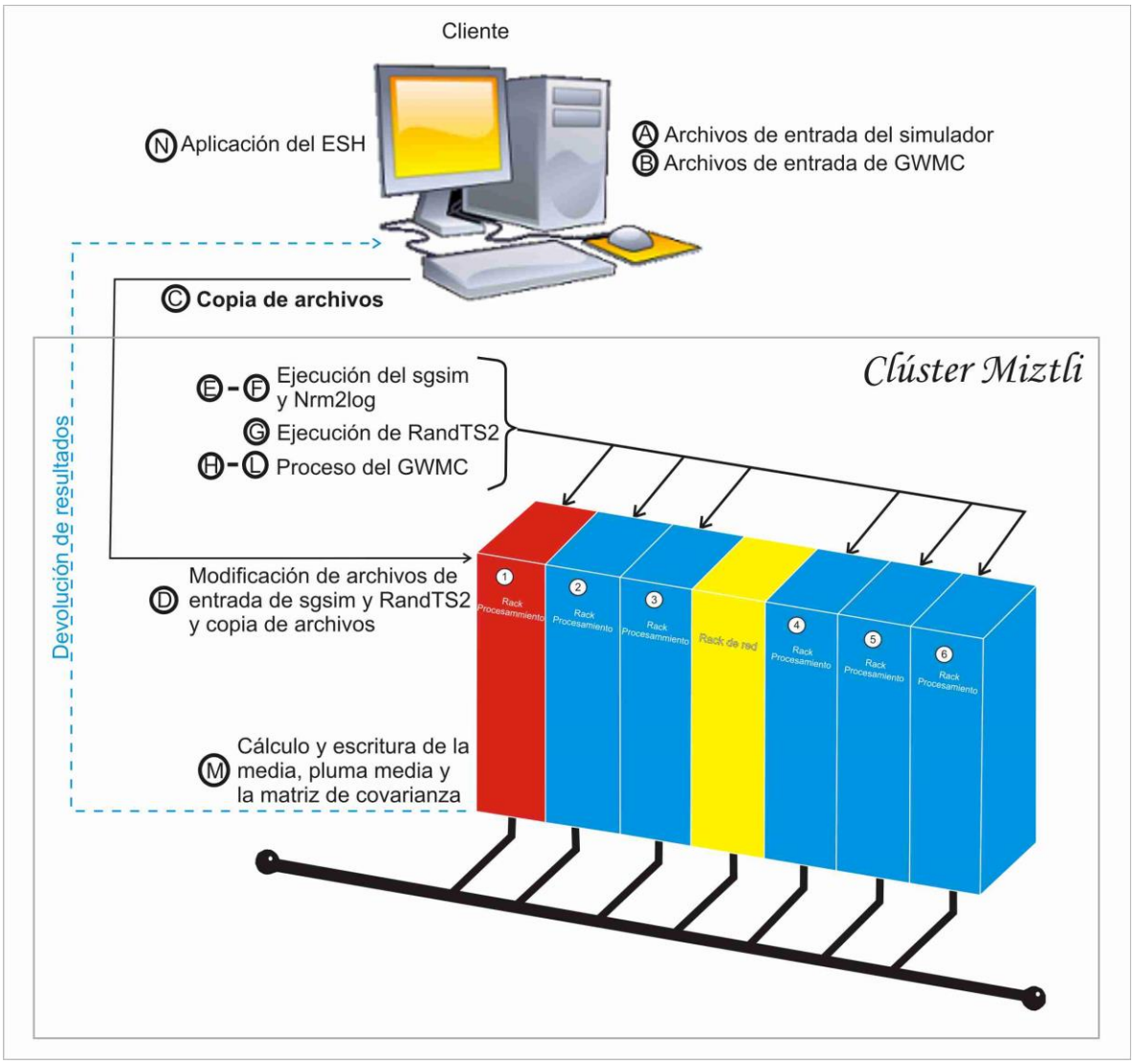


Figura 4. 11 Proceso de paralelización. Las tareas del E al L se hacen en paralelo en cada uno de los procesadores, mientras que las tareas D y M se calculan en el procesador 1 que funciona como maestro.

Una premisa de este trabajo es utilizar los códigos de *Fortran* que se tienen, realizando mínimas modificaciones, con este fin se utilizaron *Python* y *MPI* para *Python* (*mpi4py;* Dalcin, 2012). *Mpi4py* proporciona un enfoque orientado a objetos para *MPI* que permite distribuir tareas con scripts de *Python*.

Primero se inicializan todas las variables y se leen las entradas para los diferentes códigos ejecutables. Esta parte del proceso, que es en serie, se realiza en una máquina cliente. Después de eso, la máquina cliente se encarga de enviar los datos a los procesadores del clúster. Una vez que comienza la ejecución en paralelo, cada procesador genera sus propios archivos de entrada etiquetados con el número de procesador. Con las entradas locales generadas, se ejecutan la parte

entera de nr/p realizaciones en cada procesador. El equilibrio de carga se realiza en el script, distribuyendo el mismo número de tareas para cada procesador, sin embargo, cada realización resuelve el mismo problema, pero con diferentes entradas, por lo que el tiempo requerido por cada una puede ser un poco diferente; por tanto, se usa una barrera al final de la ejecución en paralelo, sin embargo, el tiempo de espera es insignificante. Los cálculos del vector de la media y la matriz de covarianza espaciotemporal requieren la información de todos los procesadores, y se realiza en el procesador 1. En la versión en serie del programa esto se realizaba dentro de *GWMC*, pero para paralelizarlo esto se modificó, removiendo el código correspondiente del programa en *Fortran* y corriendo un programa independiente al final del script. Finalmente, el último paso (aplicación del *ESH*) se realiza como una etapa de procesamiento posterior en la máquina cliente. Esta parte no se paralelizó y no está considerada en los tiempos reportados en las pruebas que se presentan en esta tesis.

En la figura 4.12 se muestra el esquema de la paralelización del *GWMC-PTC* y en la figura 4.13 se muestra el de la paralelización *GWMC-MFMT*, los rectángulos representan el nombre del software utilizado. Se puede observar que en ambos casos el *Sgsim* genera el número de realizaciones de la con una distribución gaussiana estándar; en el caso del *GWMC-PTC*, el *Nrm2log* transforma estas realizaciones aleatorias a una distribución lognormal y el *RandTS2* genera el mismo número de realizaciones aleatorias de la concentración del soluto en su fuente, en el caso del *GWMC-MFMT*, dentro del ciclo del *GWMC* la subrutina *Nrm2log* transforma las realizaciones aleatorias generadas con el *Sgsim* a realizaciones aleatorias con una distribución lognormal y la subrutina *randsrc* genera las realizaciones aleatorias de la concentración del soluto en su fuente (por eso no se muestran explícitamente en el esquema); después de esto se resuelven las ecuaciones de flujo y transporte con los simuladores que correspondan, el número de veces correspondiente a cada procesador y se obtienen los archivos con los cálculos de la matriz de covarianza auxiliar (*covmatrx.out*), el vector de la media de los puntos de estimación (*meanvect.out*) y el vector de la pluma media para una capa (*meanplume.out*). Teniendo todos los archivos de la matriz auxiliar de covarianza, vector auxiliar de la media de los puntos de estimación y el vector auxiliar de la pluma media para una capa, éstos se mandan al procesador 1 donde se ejecuta el software *Matriz* (código escrito en *Fortran*), que se encarga de realizar la matriz de covarianza completa, así como el vector de la media de los puntos de estimación y el vector de la pluma media para una capa.

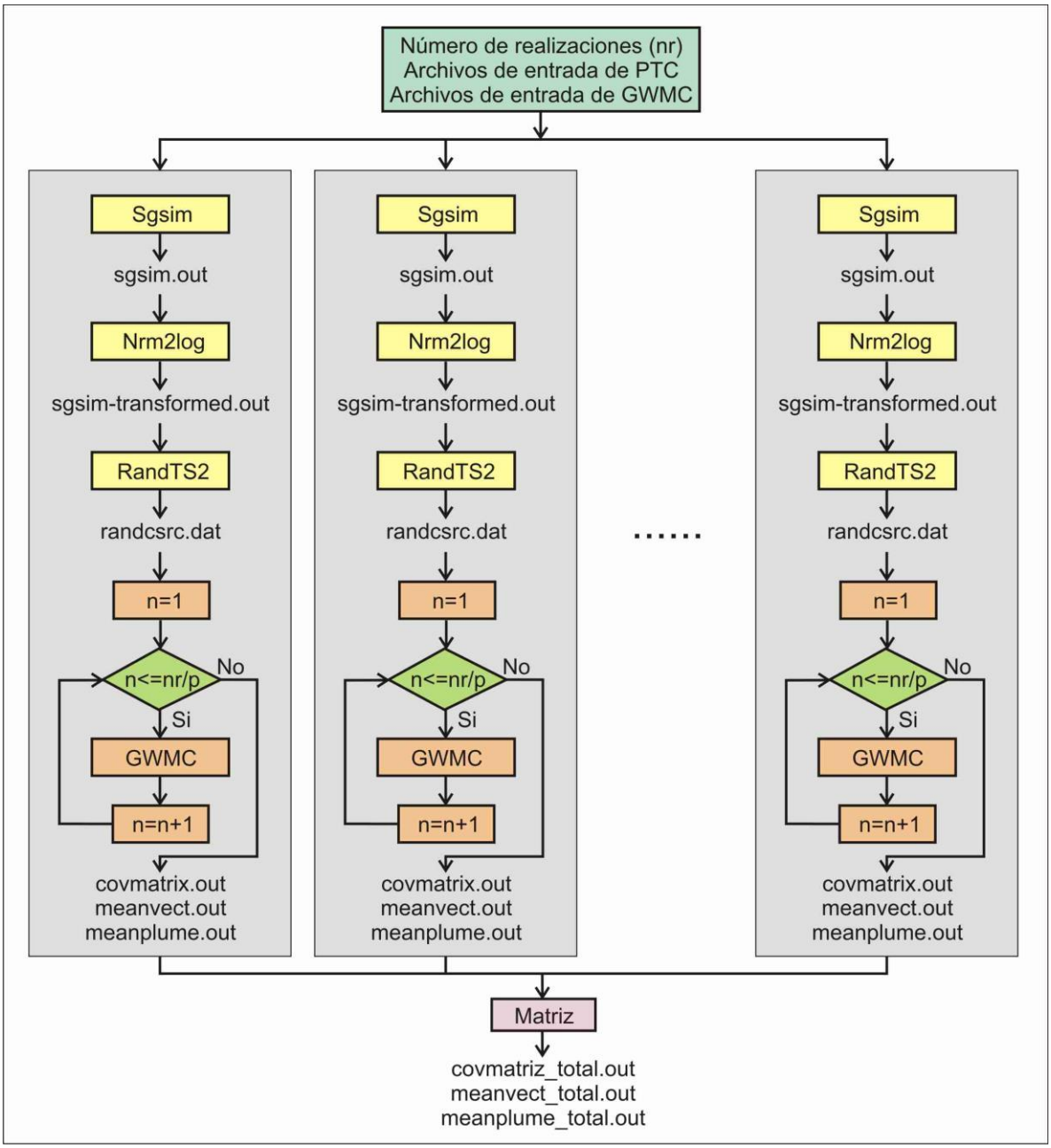


Figura 4. 12 Esquema de la paralelización del *GWMC*, utilizando el simulador *PTC*.

En el anexo I se describen algunas partes del script que se utilizó para paralelizar el proceso para GWMC-PTC descrito en esta sección.

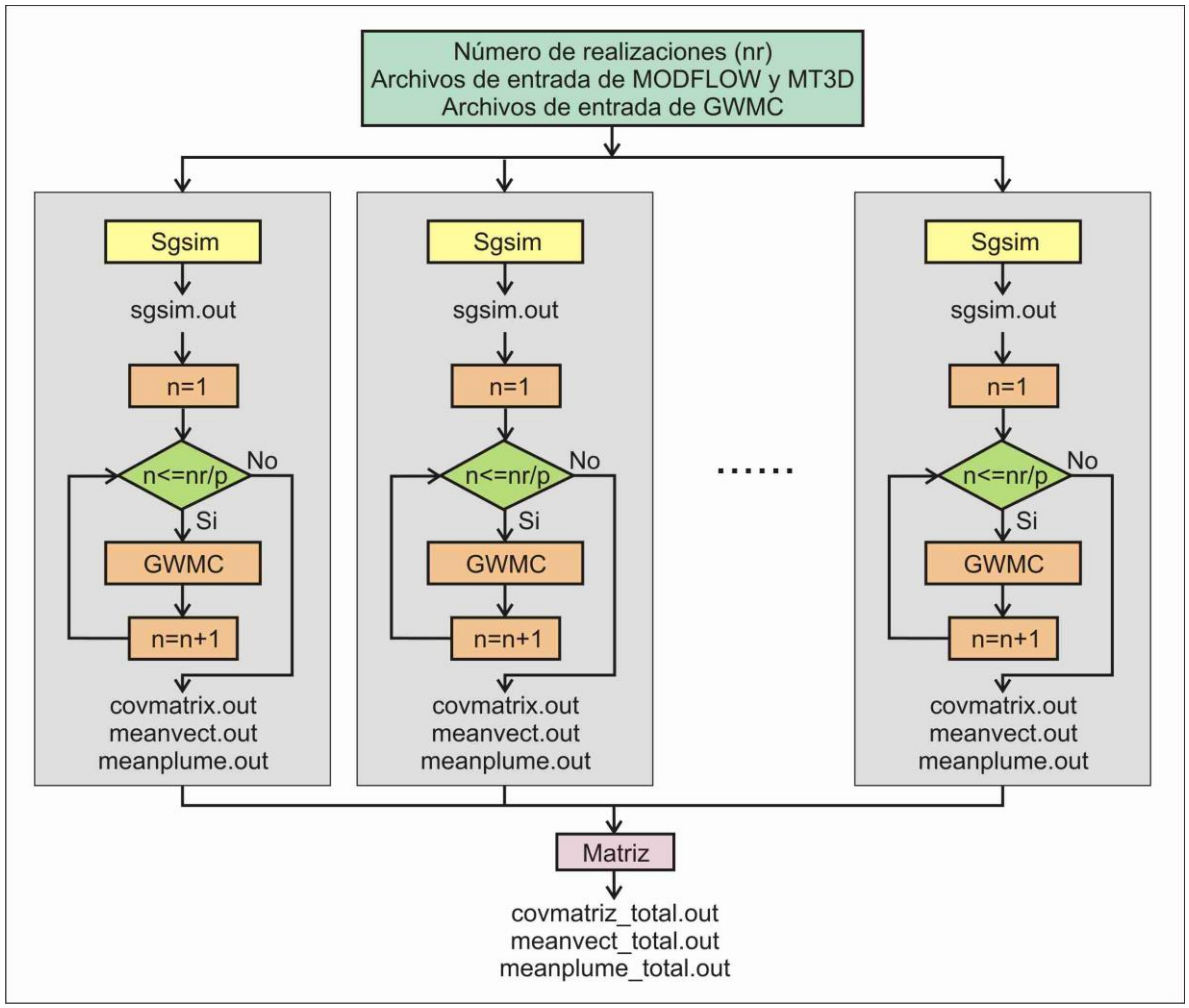


Figura 4. 13 Esquema de la paralelización del *GWMC*, utilizando el simulador *MODFLOW/MT3D*.

5. Casos de estudio y resultados

En este capítulo se presentan dos casos de estudio, para los cuales se calculan, en paralelo, el vector de medias y el matriz espaciotemporal que requiere el *ESH* que se describió en el capítulo dos. El primero es un caso sencillo que se corrió con *GWMC-PTC* (publicado en Leyva et al., 2015) y el segundo es un caso complejo que se corrió con *GWMC-MFMT*, los cuales se describen con más detalle a continuación.

5.1 Caso de estudio sencillo (*GWMC-PTC*)

Este problema se modificó ligeramente del presentado por Herrera y Pinder (2005). Se considera un acuífero en un área de 0.5 millas por 0.5 millas (804.7 m por 804.7 m) como el que se muestra en la figura 5.1a. Del lado izquierdo se localiza una fuente de contaminante y la región está delimitada por un río del lado derecho. Se considera que el contaminante que se modela es conservativo, es decir, que su concentración no varía al interactuar con el medio y que, por tanto, al atravesar el acuífero la masa se conserva.

El objetivo de este ejemplo es calcular el vector de medias y la matriz de covarianza necesarios para aplicar el Ensamble Suavizado de Herrera (*ESH*) para estimar la concentración del contaminante en una zona determinada del acuífero al asimilar los datos "existentes" a las salidas de un modelo de transporte de contaminantes para un periodo de dos años.

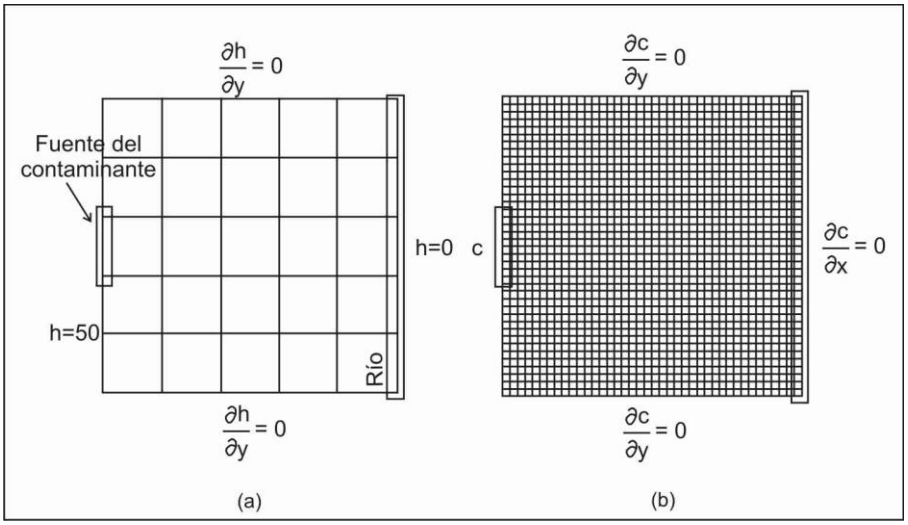


Figura 5. 1 a) Malla del *ESH* y las condiciones para el flujo (h está en m), b) malla de simulación estocástica y las condiciones de frontera para el transporte (modificada de Olivares-Vázquez, 2002).

5.1.1 Modelo de flujo y transporte

Se cuenta con un modelo de flujo y transporte de una sola capa, en dos dimensiones. El flujo del agua en el acuífero está en estado de equilibrio.

Se usan las ecuaciones de flujo y transporte acopladas por la ley de Darcy para describir la evolución de la pluma del contaminante, las cuales se resuelven para las cargas hidráulicas y las concentraciones del contaminante:

$$\nabla \cdot (\mathbf{K} \cdot \nabla h) = 0 \qquad (5.1)$$

$$\frac{\partial c}{\partial t} - \nabla \cdot (\mathbf{D} \cdot \nabla c - \mathbf{V} c) = 0 \qquad (5.2)$$

$$\mathbf{V} = -\frac{\mathbf{K}}{\phi} \nabla h \qquad (5.3)$$

Donde $\mathbf{K}$ es la conductividad hidráulica, h la carga hidráulica, c la concentración del soluto, $\mathbf{D}$ es la dispersión hidrodinámica, $\mathbf{V}$ la velocidad de poro, ϕ porosidad efectiva, C es la concentración del fluido bombeado. La ecuación de flujo (ecuación 5.1) describe el flujo del agua a través del acuífero, la ecuación de transporte (ecuación 5.2) describe los cambios en la concentración del contaminante a través del tiempo para un soluto conservador. Con la ley de Darcy (ecuación 5.3) se calcula la velocidad de poro del agua subterránea utilizando las cargas y la conductividad hidráulica de la ecuación de flujo (ecuación 5.2).

Las condiciones de frontera para el flujo y transporte se incluyen en las figuras 5.1a y 5.1b respectivamente. Las concentraciones están dadas en partes por millón (ppm) y las cargas hidráulicas en metros (m).

A la malla que se utiliza para resolver las ecuaciones de flujo y transporte la llamaremos “malla de simulación estocástica”, que consta de 40×40 elementos del mismo tamaño (ver figura 5.1 b).

Para el modelo de transporte, se usan 48 pasos de tiempo para simular un periodo de dos años, 15.2083 días cada uno. Para el modelo de flujo, todos los nodos del lado izquierdo tienen un valor $h = 50\,m$, y todos los nodos del límite derecho tiene un valor de $h = 0\,\mathrm{m}$. Durante todo este

periodo la fuente de contaminante está activa, con una concentración constante $c = 50\,\text{ppm}$. Los nodos que no son parte de la fuente del contaminante satisfacen la condición $\frac{\partial c}{\partial x} = 0$. Se le asigna al acuífero un espesor de 55 m, una porosidad de 0.25, una dispersividad de 33 m, en la dirección x e y un valor de 3.3 m respectivamente. El *PTC* se usa en modo bidimensional para resolver el modelo de flujo y transporte.

5.1.2 Modelo estocástico

La conductividad hidráulica es una fuente importante de incertidumbre en los modelos de flujo y transporte, por esta razón, la conductividad hidráulica en este ejemplo se representa como un campo aleatorio espacialmente correlacionado, y en consecuencia el campo de velocidades, el campo de dispersión y el campo de concentración resultante también se convierten en campos aleatorios espacialmente correlacionados.

Por tanto, las variables aleatorias a utilizar en el ejemplo son la conductividad hidráulica y la concentración del contaminante en la fuente.

Para este ejemplo supondremos que el campo de la conductividad hidráulica tiene una distribución log normal, homogénea, estacionaria e isotrópica. El valor medio del campo del logaritmo de las conductividades es de 3.055 y el semivariograma que representa la estructura de correlación espacial es un modelo exponencial, es decir:

$$\gamma_F(h) = \sigma_F^2 \left[1 - \exp\left(-\frac{h}{\lambda_F} \right) \right] \qquad (5.4)$$

Donde σ_F^2 es la varianza de $F = \ln(K(x))$ con un valor de 0.257813 y λ_F es la escala de correlación espacial de F igual a 80.467 m.

Se aproxima el campo aleatorio continuo con un campo discreto asociado a la malla numérica. Los efectos de la integración local en la discretización de longitud Δx se puede evitar si λ_F y la varianza F , σ_F^2, satisfacen la relación (Ababou, 1988):

$$\frac{\lambda_F}{\Delta x} \geq 1 + \sigma_F^2 \qquad (5.5)$$

Las concentraciones de la fuente se modelan como funciones aleatorias independientes idénticamente distribuidas. La concentración en cada nodo de la fuente del contaminante se representa como una serie de tiempo mediante:

$$c(t) = \exp(-14 + 3t + e(t)) \qquad (5.6)$$

Donde $e(t)$ es una perturbación aleatoria de media cero, con una distribución normal y varianza 0.1948. Para cada nodo de la fuente, en cada paso de tiempo de simulación, se utiliza una perturbación aleatoria diferente. La correlación temporal de las perturbaciones aleatorias se modela con el semivariograma

$$\gamma_e(t) = 0.1948\left[1 - exp\left(-\frac{t}{\lambda_e}\right)\right] \qquad (5.7)$$

Con λ_e igual a 11 días.

Se utiliza un método llamado de simulación secuencial gaussiana (*Sgsim*) del paquete *GSLIB* (Deutsch y Journel, 1998) para obtener las realizaciones aleatorias.

5.1.3 Vector espacio temporal de medias y matriz de covarianza

Utilizando datos existentes de la concentración del contaminante durante el periodo de dos años se pretende estimar la concentración del contaminante en los nodos de una malla que llamaremos la malla de *ESH*, que es una submalla de la malla de simulación estocástica, y consta de 5x5 elementos de igual tamaño. Para cada una de estas posiciones, las concentraciones se estiman seis veces en un periodo de dos años, equivalente a 121.7 días. Por lo tanto, es necesario calcular el vector de estimación inicial en 216 posiciones espaciotemporales y la matriz de covarianza espaciotemporal del error de estimación correspondiente de dimensión 216 X 216 = 46,656.

Las posiciones de muestreo son un subconjunto de las posiciones de estimación, por lo que el tamaño de la matriz de covarianza no se modifica por éstas.

5.1.4 Resultados

Se ejecutaron los códigos en el sistema HP Clúster Platform 3000SL "Miztli", el cual consta de 5,312 núcleos de procesamiento Intel E5-2670, con 16 tarjetas NVIDIA m2090, una memoria *RAM*

total de 15,000 Gbytes y una capacidad de procesamiento de 118 TFlop/s. El sistema tiene 750 TB de almacenamiento masivo.

El programa desarrollado en *Python* para realizar la paralelización se ejecutó para 1,000, 2,000 y 4,000 realizaciones con diferentes números de procesadores. En todos los casos reportados se tiene un equilibrio de carga óptimo ya que, por diseño del algoritmo, cada procesador funciona con el mismo número de realizaciones. En la tabla 5.1 se reportan tiempos de ejecución, aceleración (ecuación 3.1), eficiencia (ecuación 3.2) y ley de Amdahl (ecuación 3.3) obtenidos para los diferentes números de procesadores. El tiempo de ejecución va disminuyendo conforme va aumentando el número de procesadores. En la figura 5.2 se muestra la aceleración al aumentar el número de procesadores para los diferentes números de realizaciones, se puede observar que la aceleración se va incrementando cuando el número de procesadores aumenta. En la figura 5.3 vemos que la eficiencia es más estable para el caso de 4,000 realizaciones, ya que tiene menos oscilaciones que los casos para 1,000 y 2,000 realizaciones respectivamente. Para el caso de 1,000 realizaciones, se obtuvo una aceleración de 45.06 con 64 procesadores y una eficiencia correspondiente de 0.70; para el caso de 2,000 realizaciones, se obtuvo una aceleración de 48.64 con 64 procesadores y una eficiencia correspondiente de 0.76; para el caso de 4,000 realizaciones, se obtuvo una aceleración de 72.18 con 96 procesadores y una eficiencia correspondiente de 0.75.

Realiz. / Proc.	Tiempo (seg)			Sp			Ep			Ley de Amdahl		
	1,000	2,000	4,000	1,000	2,000	4,000	1,000	2,000	4,000	1,000	2,000	4,000
1	2606.93	4895.03	9186.88	1	1	1	1	1	1	1	1	1
2	1325.09	2601.75	4815.77	1.96	1.88	1.90	0.98	0.94	0.95	1.99	1.99	1.99
4	658.04	1409.16	2449.71	3.96	3.47	3.75	0.99	0.86	0.93	3.99	3.99	3.97
8	419.09	674.90	1288.65	6.22	7.25	7.12	0.77	0.90	0.89	7.97	7.98	7.88
12	229.28	494.53	891.92	11.36	9.89	10.30	0.94	0.82	0.85	11.94	11.96	11.73
16	181.82	362.38	735.51	14.33	13.50	12.49	0.89	0.84	0.78	15.90	15.93	15.52
24	121.89	283.03	474.89	21.38	17.29	19.34	0.89	0.72	0.80	23.77	23.85	22.92
32	97,.3	182.63	366.75	26.75	26.80	25.04	0.83	0.83	0.78	31.60	31.74	30.09
48	64.40	123.91	244.15	40.47	39.50	37.62	0.84	0.82	0.78	47.10	47.42	43.79
64	57.84	100.63	187.19	45.06	48.64	49.07	0.70	0.76	0.76	62.41	62.96	56.70
80	62.20	113.94	153.47	41.90	42.96	59.85	0.52	0.53	0.74	77.52	78.39	68.88
96	81.19	70.79	127.26	32.10	69.14	72.18	0.33	0.72	0.75	92.45	93.68	80.40

Tabla 5. 1 Datos de tiempo (seg), aceleración (Sp), eficiencia (Ep) y la ley de Amdahl con diferentes números de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso sencillo.

Los resultados que se obtuvieron en paralelo muestran que el rendimiento es más estable a medida que aumenta la carga de trabajo en cada procesador. En particular, se obtuvo una eficiencia muy buena, de 0.75, para 4,000 realizaciones con 96 procesadores. Durante el desarrollo de este trabajo, no hemos instalado ningún software complicado, sólo se han instalado las bibliotecas comunes en el clúster Miztli, además de haber realizado una modificación al código original en Fortran para calcular la matriz de covarianza.

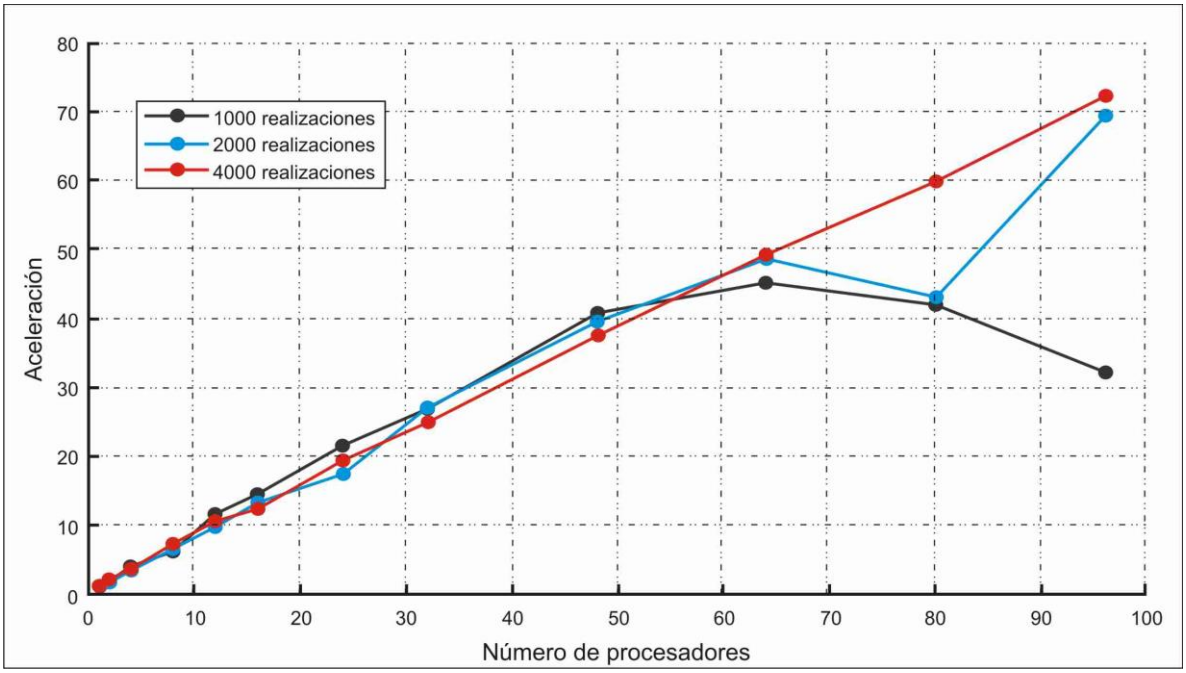


Figura 5. 2 Aceleración vs números de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso sencillo.

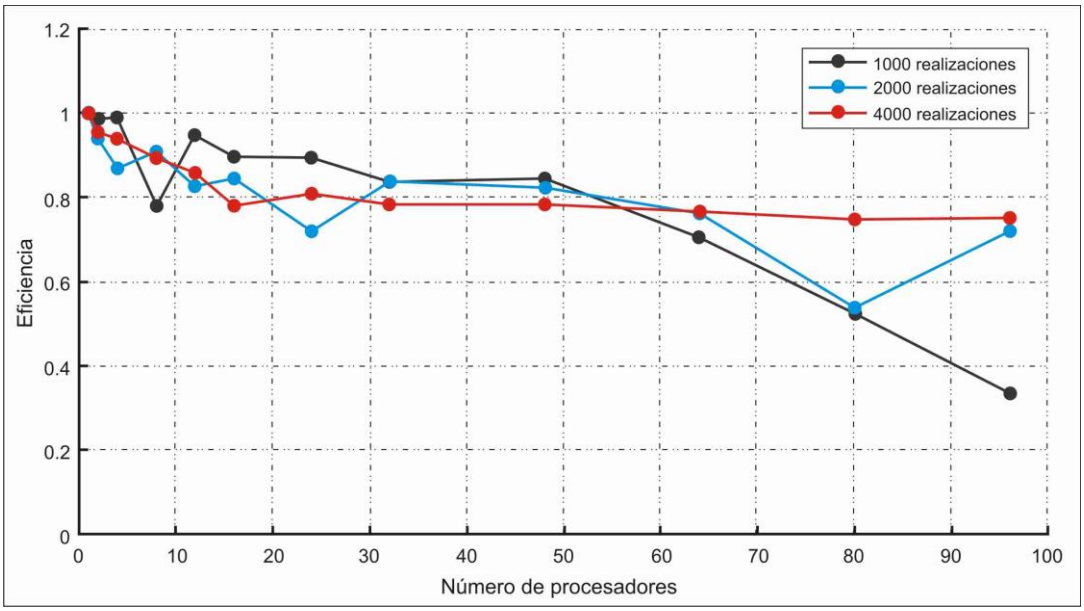


Figura 5. 3 Eficiencia vs números de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso sencillo.

En la tabla 5.2 se muestran los cálculos del costo (ecuación 3.4) para este caso. Se puede observar que este se incrementa conforme aumenta el número de procesadores, para 1,000, 2,000 y 4,000 realizaciones.

Realizaciones / Procesadores	Costo		
	1,000	**2,000**	**4,000**
1	2,606.93	2,666.93	159,083.15
2	2,650.18	2,890.18	161,900.98
4	2,632.16	3,592.16	161,521.76
8	3,352.79	7,192.79	208,360.31
12	2,751.42	11,391.42	176,476.62
16	2,909.23	18,269.23	192,823.15
24	2,925.43	37,485.43	213,011.35
32	3,118.01	64,558.01	251,638.97
48	3,091.39	141,331.39	326,814.91
64	3,702.01	249,462.01	471,582.97
80	4,976.48	388,976.48	687,565.28
96	7,794.72	560,754.72	1,028,437.92

Tabla 5. 2 Cálculos de los costos en diferentes números de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso sencillo.

En la figura 5.4 se muestra el comportamiento que tiene el costo, para 1,000, 2,000 y 4,000 realizaciones, conforme aumenta el número de procesadores. Se puede observar que para 1,000 realizaciones prácticamente el comportamiento es lineal, sin embargo, para 2,000 y 4,000 realizaciones el costo tiene un comportamiento exponencial. Para 2,000 realizaciones, se obtuvo una aceleración de 48.64 con 64 procesadores y un costo de 249,462.01; para 4,000 realizaciones se obtuvo una aceleración de 72.18 con 96 procesadores y un costo de 1,028,437.92.

Nótese que por un lado se obtiene una buena eficiencia para 4,000 realizaciones con 96 procesadores, sin embargo, a un costo elevado.

En la tabla 5.3 se muestran los cálculos de la memoria RAM por matriz que se utiliza dependiendo del número de procesadores.

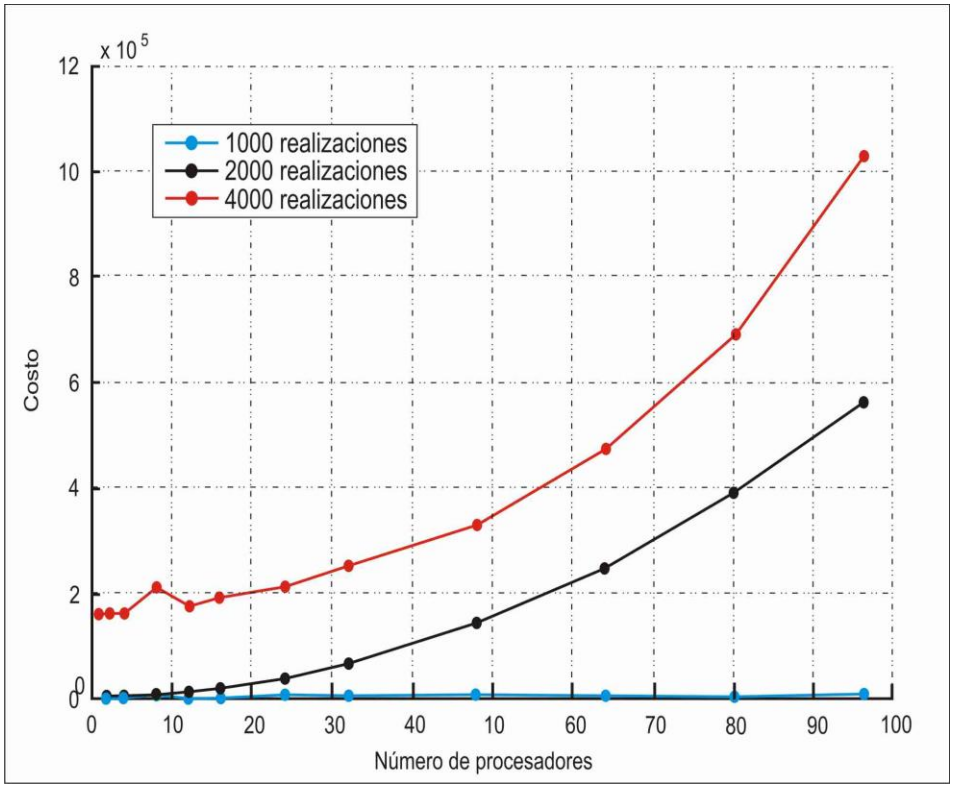


Figura 5. 4 Grafica del costo para 1,000, 2,000 y 4,000 realizaciones para el caso sencillo.

Procesadores	Memoria RAM (MB)
1	373.24
2	746.49
4	1,492.29
8	2,985.98
12	4,478.97
16	5,971.96
24	8,957.95
32	11,943.93
48	17,915.90
64	23,887.87
80	29,859.84
96	35,831.80

Tabla 5. 3 Cálculos de la memoria RAM por matriz a utilizar en diferentes números de procesadores para el caso sencillo.

El cálculo de la memoria por matriz se hizo con la siguiente fórmula:

$$n \times N_m \times 8 \qquad (5.8)$$

donde

n número de procesadores

N_m dimensión de la matriz

8 bytes.

En la figura 5.5 se muestra el comportamiento de la memoria dependiendo del número de procesadores, como se puede observar, hay un incremento lineal de esta al aumentar el número de procesadores. Se puede ver que, en este caso, con un procesador se usan 373.24 Mb y con 96 procesadores se utilizan 35,831.80 Mb, lo que es un gran incremento.

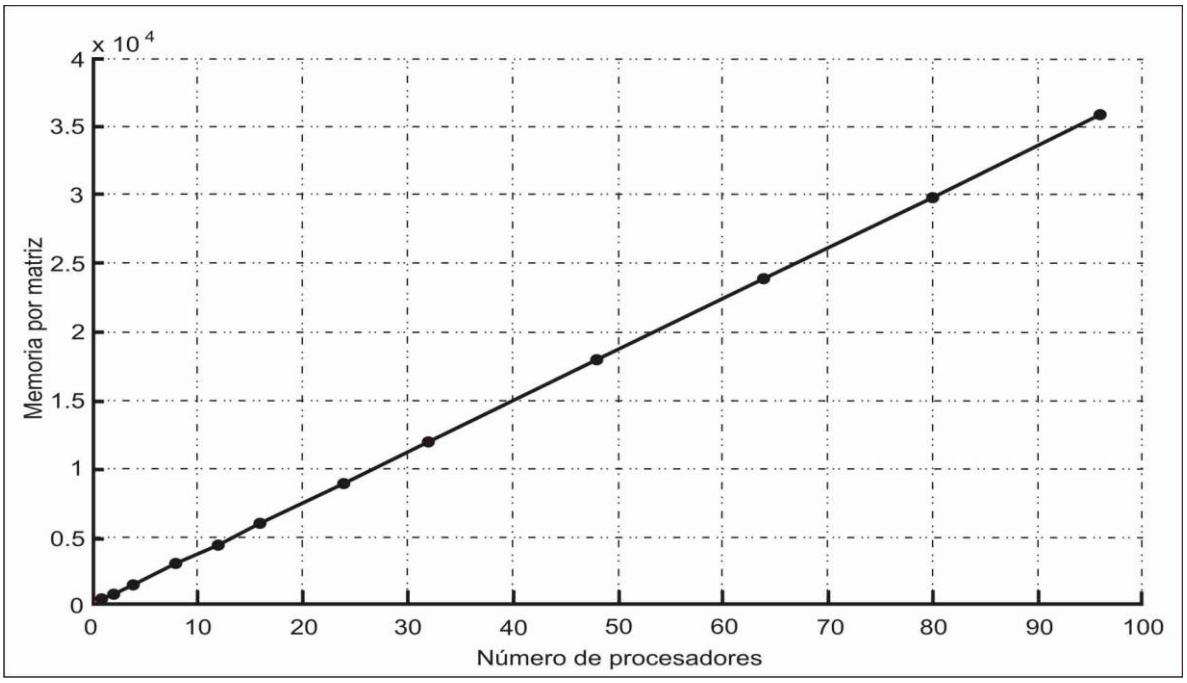


Figura 5. 5 Uso de la memoria RAM por matriz en diferentes números de procesadores para el caso sencillo.

5.1.5 Discusión

Tanto el tiempo de ejecución como la aceleración y la eficiencia están limitados por varios factores, entre ellos: la fracción en serie del código, las comunicaciones, las dependencias de datos, el balance de carga y la memoria. Para este caso, se tiene una parte mínima en serie, al principio del código, cuando se configura el problema en cada procesador y al final del código cuando se juntan los resultados de todos los procesadores para que se calcule la matriz de covarianza y el vector de la media. Las comunicaciones que pueden influir en el tiempo de ejecución del programa son dos: al inicio en la parte serial, copia archivos de entrada y programas a ejecutarse en cada uno de los procesadores, y al final cuando se copian los archivos auxiliares de cada procesador al disco y el procesador maestro los lee para calcular la matriz de covarianza, el vector de la media y el vector de la pluma media. Durante los cálculos no hay dependencia de datos, a excepción del inicio cuando se copian los archivos de entrada y programas a ejecutarse en cada uno de los procesadores y al final cuando se calcula la matriz de covarianza, el vector de la media y el vector de la pluma media. Se tiene un balance de carga óptimo debido a que por diseño cada procesador funciona con el mismo número de realizaciones.

En la figura 5.6 se muestra la comparación de las aceleraciones para 1,000, 2,000 y 4,000 realizaciones con la ley de Amdahl para 4,000 realizaciones, y se observa que los resultados de las

aceleraciones están muy acordes con las predicciones de esta ley. Además, las eficiencias que se obtuvieron son mayores que 0.70, por lo que se puede decir que los códigos paralelos son escalables (Ridgway et al., 2005).

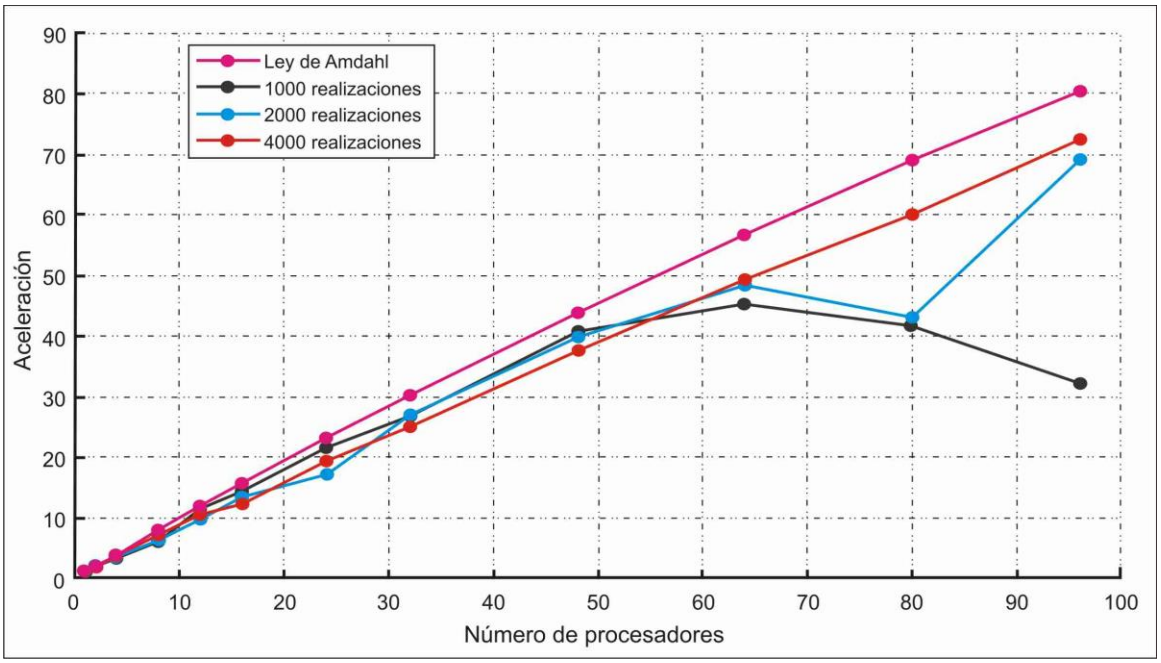


Figura 5. 6 La comparación entre la ley de Amdahl y la aceleración de 1,000, 2,000 y 4,000 realizaciones para el caso complejo.

Como se observa en la figura 5.4 el costo tiene un comportamiento exponencial, lo que indica que, al trabajar con más procesadores, el costo se incrementa de manera significativa. Lo que implica que para tener los beneficios de la paralelización implementada, es importante analizar con cuántos procesadores correr el código para que el costo que conlleva se pueda pagar o este pago valga la pena.

Como se comentó antes, el uso de memoria para almacenar las matrices de covarianza aumenta linealmente con el número de procesadores lo que claramente no es eficiente, además, las matrices auxiliares de covarianza de cada procesador se copian al disco duro y el procesador maestro las vuelve a leer, lo que conforme se aumenta el número de procesadores tardará más tiempo.

Como se explicó antes, el programa GWMC lee las realizaciones de concentraciones, las usa para calcular las matrices de covarianza auxiliares y las desecha, guardando en memoria las matrices calculadas. Otra posibilidad sería conservar en memoria las realizaciones y calcular la matriz de

covarianza cada vez que se requiere, en lugar de guardar en memoria las matrices de covarianza auxiliares. En la tabla 5.4 se muestran los cálculos de la memoria RAM que se utilizaría si se guardan solo las realizaciones.

Número de realizaciones	Memoria RAM (MB)
1,000	1,728
2,000	3,456
4,000	6,912

Tabla 5. 4 Calculo de la memoria RAM para diferentes números de realizaciones para el caso complejo.

El cálculo de la memoria para la matriz de realizaciones se realizó con la siguiente fórmula:

$$N_r \times N_v \times 8 \qquad (5.9)$$

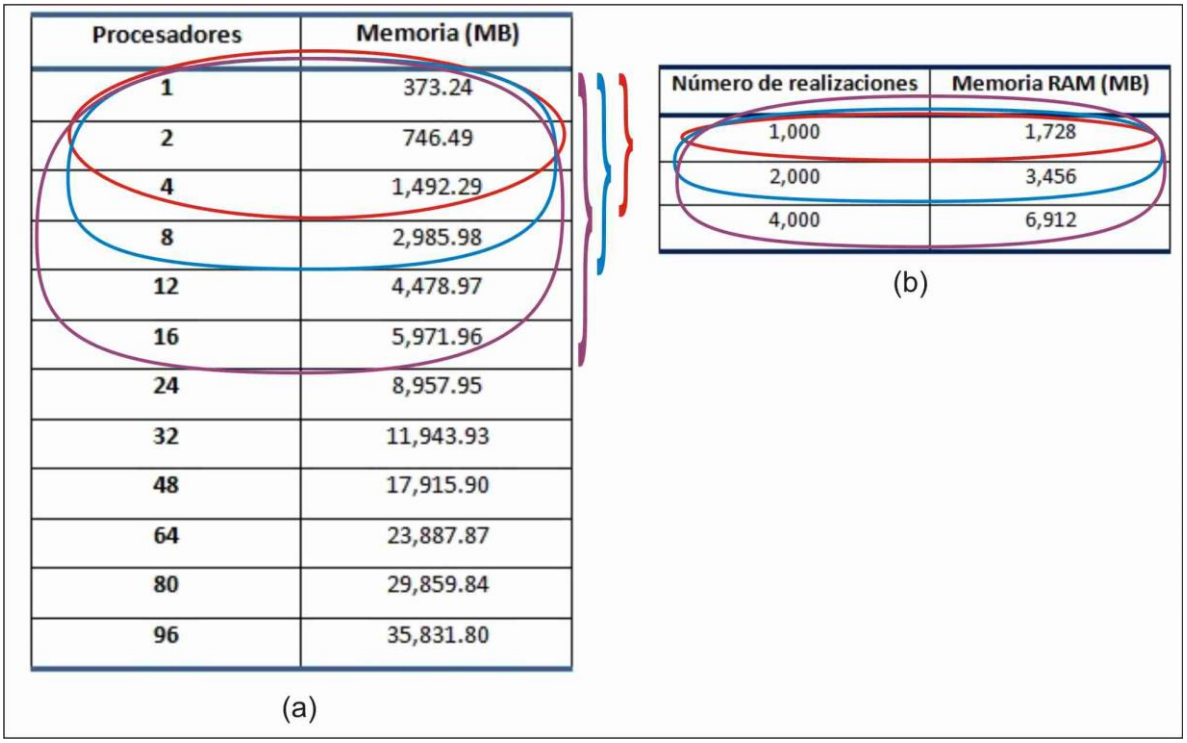
donde

N_r número de realizaciones

N_v dimensión del vector medio

8 bytes.

Para realizar la comparación de las tablas 5.3 y 5.4, nos apoyaremos con la figura 5.7, en la que se puede observar en óvalos de diferentes colores la utilización de diferentes números de procesadores en los cuales la memoria requerida al almacenar las matrices auxiliares de covarianza es menor que al almacenar las realizaciones. Los óvalos rojos indican el caso para 1,000 realizaciones, el azul para 2,000 y el violeta para 4,000. Y en lo que no está marcado con estos colores, la memoria requerida para almacenar las realizaciones es menor que al almacenar las matrices auxiliares de covarianza.



Procesadores	Memoria (MB)
1	373.24
2	746.49
4	1,492.29
8	2,985.98
12	4,478.97
16	5,971.96
24	8,957.95
32	11,943.93
48	17,915.90
64	23,887.87
80	29,859.84
96	35,831.80

(a)

Número de realizaciones	Memoria RAM (MB)
1,000	1,728
2,000	3,456
4,000	6,912

(b)

Figura 5. 7 Comparación del uso de memoria RAM al almacenar matrices auxiliares (a) y al almacenar realizaciones (b) para el caso sencillo.

5.2 Caso de estudio complejo (*GWMC-MFMT*)

Consideramos un acuífero piloto localizado en Nueva Jersey (McLane Environmental, 2010). El sitio tiene una superficie de aproximadamente 1,350 hectáreas y una altura aproximada de 50 pies (15.24 m) sobre el nivel medio del mar, limitada por áreas recreativas, colonias residenciales, complejos industriales ligeros, pequeños establecimientos comerciales y un río que está ubicado al este y que fluye de norte a sur (Figura 5.8).

En la figura 5.9 se puede observar que el acuífero está dividido en nueve unidades geológicas, representando 7 unidades hidrogeológicas, las unidades en color gris son acuitardos y las unidades en blanco son transmisoras de agua. Estas unidades en orden descendente son:

1. Miembro Cohansey Superior: está compuesta por arena.
2. Miembro Arcilla Amarilla de Cohansey: está compuesta por limo y arena.
3. Miembro Cohansey Primario: está compuesta por arena.
4. Miembro de Transición Cohansey/Kirkwood
5. Miembro Cohansey Inferior: está compuesta por arena fina, lentes de arena de color oscuro y camas de arcilla que van gradualmente a las arenas de manera lateral.

6. Miembro Kirkwood Superior: está compuesta por arena fina, lentes de arena de color oscuro y camas de arcilla que van gradualmente a las arenas de manera lateral, con baja conductividad.
7. Miembro Kirkwood No.1: está compuesta por arena fina, lentes de arena de color oscuro y camas de arcilla que van gradualmente a las arenas de manera lateral.
8. Unidad semiconfinante (Kirkwood Primario): consiste de unidades intercaladas de limo y arcilla con intercalaciones de arcilla arenosa y arcilla limosa.
9. Acuífero de Arena Inferior (Arena Kirkwood No.2): está compuesta por arenas de granos finos a medios, con contenido de arcilla o limo.

En algunas porciones de las unidades geológicas Cohansey Superior, Arcilla Amarilla de Cohansey, Cohansey Primario, Transición Cohansey/Kirkwood y Cohansey Inferior se han lixiviado algunos contaminantes, aunque bajo la unidad geológica Cohansey Inferior no se detectó contaminación.

En el área de estudio se detectaron contaminantes en el agua subterránea debido a las actividades asociadas con las operaciones históricas, los datos de muestreo indican veintiún compuestos que exceden los límites permitidos para la calidad del agua subterránea. Se han realizado trabajos de restauración en la fuente del contaminante a través de una combinación de tecnologías in situ, de biodegradación mejorada, y ex situ, de excavación y tratamiento in-situ, entre el 2001 y 2008, y en el acuífero, a través de un extenso sistema de extracción, tratamiento y recarga de agua subterránea. Por otro lado, se ha instalado un sistema de monitoreo para supervisar la remediación a través de una red de pozos de observación. Como un auxiliar en estos trabajos, se empleó el modelado matemático del flujo y transporte de un subconjunto de estos compuestos para predecir la persistencia, concentración y la distribución de los contaminantes en el agua subterránea. Con ayuda de este modelo se simula la evolución de la pluma del contaminante con el tiempo, así como la restauración general del acuífero mediante la reducción de la masa de contaminantes con el tiempo.

El objetivo de este caso de estudio es calcular el vector de medias espaciotemporal y la matriz de covarianza necesarios para utilizar el Ensamble Suavizado de Herrera (*ESH*) para estimar la concentración del contaminante en la zona del acuífero en la que la presencia de contaminantes es importante con los datos existentes para un período de 4 años.

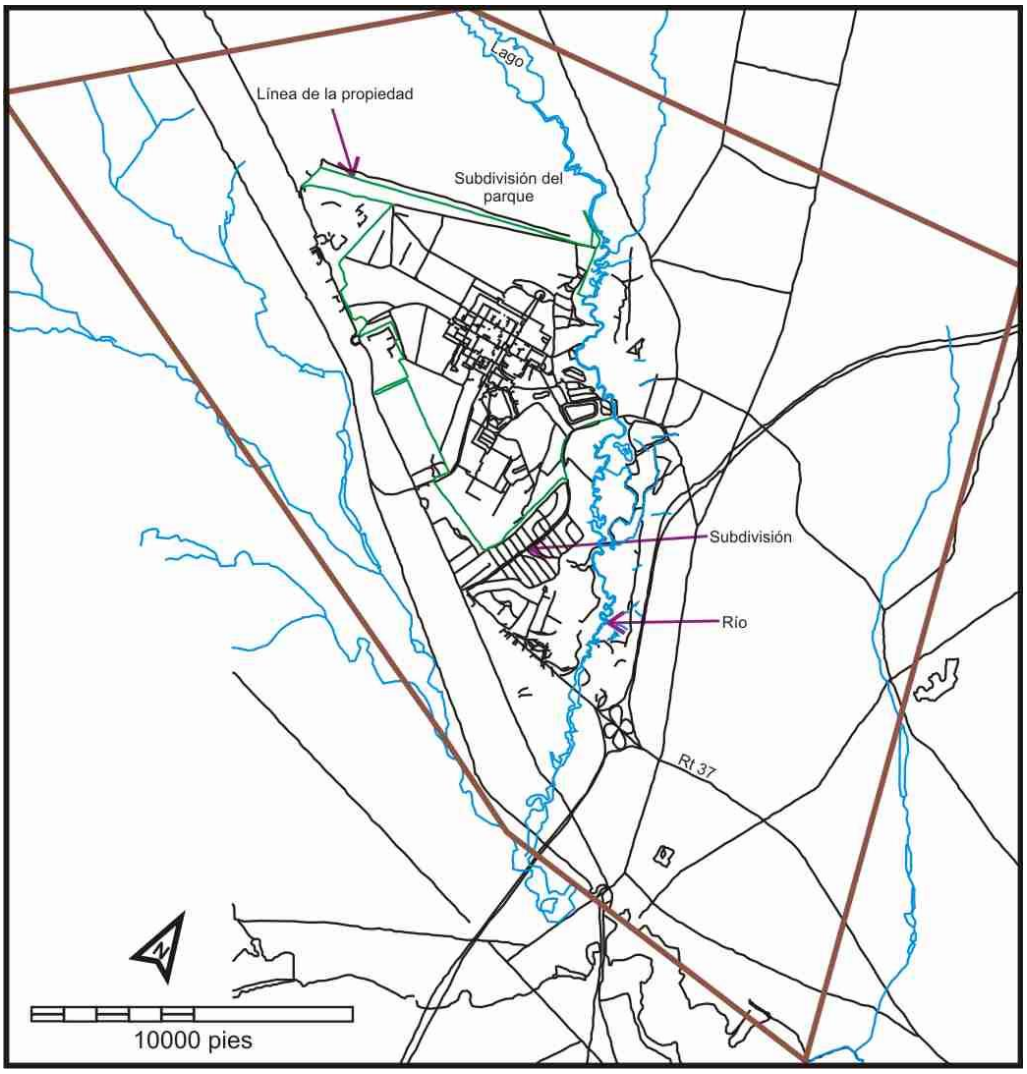


Figura 5. 8 Localización del área de estudio.

5.2.1 Modelo de flujo y transporte

Se cuenta con un modelo de flujo en *MODFLOW* y transporte en *MT3D* de nueve capas con espesor variable (figura 5.9), en tres dimensiones (McLane Environmental, 2010). Para describir la evolución de la pluma del contaminante se usan las ecuaciones de flujo y transporte acopladas por medio de la ley de Darcy.

El modelo de flujo es transitorio e incorpora todos los datos relacionados con el flujo del agua subterránea, como son: nivel del río, niveles de estratigrafía y el gasto del bombeo de los pozos. El dominio y la malla del modelo se muestran en la figura 5.10, esta última consta de 123 columnas y 123 filas, es decir, cuenta con 15,129 elementos de igual tamaño. En la figura 5.10, también se muestran las celdas de color azul, correspondientes a los límites de carga asignada; las celdas de

color verde son los límites dependientes de la carga, es decir, de los cuerpos de agua superficial, y los demás límites laterales sin flujo. En el modelo se consideró la infiltración con base en la topografía, uso de suelo y distribución del agua colgada.

La ecuación del flujo utilizando *MODFLOW* es:

$$\frac{\partial}{\partial x}\left(K_{xx}\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_{yy}\frac{\partial h}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_{zz}\frac{\partial h}{\partial z}\right)+W=S_s\frac{\partial h}{\partial t} \qquad (5.10)$$

Donde

x, y, z Son las coordenadas cartesianas, alineadas a lo largo de los ejes principales del tensor de la conductividad hidráulica

K_{xx}, K_{yy}, K_{zz} Las principales componentes del tensor de conductividad hidráulica

h Carga hidráulica

W Flujo volumétrico por unidad de volumen de las fuentes y/o sumideros del agua

S_s Almacenamiento específico de los acuíferos

t Tiempo

En la figura 5.11 se muestra la distribución de las conductividades hidráulicas dadas en las capas 1, 2, 3, 4, 5 y 6.

El modelo de transporte simula el clorobenceno de manera transitoria, se consideraron los términos de advección, dispersión, adsorción y biodegradación. Se supuso una porosidad efectiva de 25% para el proceso de advección. Los coeficientes de dispersión utilizados en el modelo son: longitudinales, transversales y vertical de 50 pies (15.24 m), 5 pies (1.53 m) y 1 pie (0.91 m) respectivamente. Para representar la cantidad de adsorción del clorobenceno, se usó el factor de retardación con un valor de 3.63 para este caso. Para la biodegradación se tomaron valores de bio-decaimiento de vida media de 3.1 años y el coeficiente de decaimiento de 0.11 años.

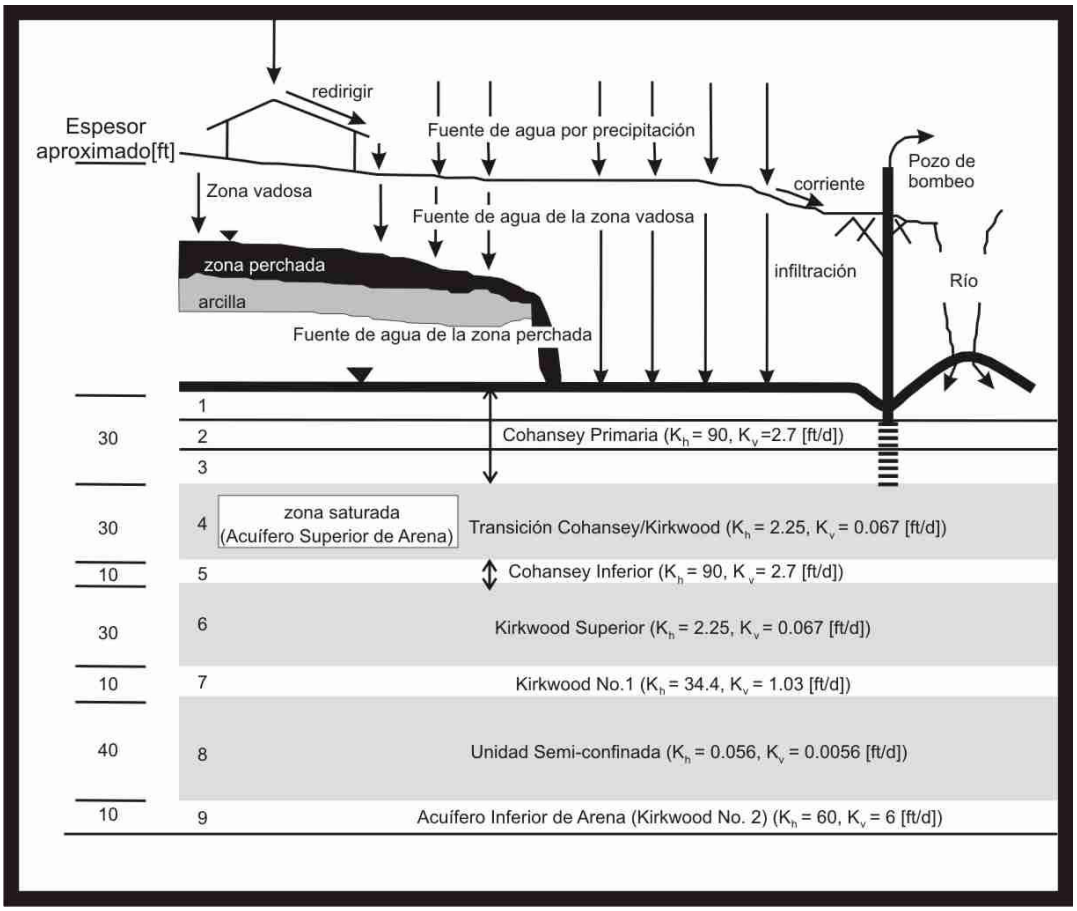


Figura 5. 9 Modelo conceptualizado, una visión vertical (modificada de McLane Environmental, 2010).

La ecuación de transporte utilizada en el simulador *MT3DMS* es:

$$\frac{\partial(\theta C^k)}{\partial t} = \frac{\partial}{\partial x_i}\left(\theta D_{ij}\frac{\partial C^k}{\partial x_j}\right) - \frac{\partial}{\partial x_i}(\theta v_i C^k) + q_s C_s^k + \sum R_n \qquad (5.11)$$

Donde

θ porosidad del medio

C^k concentración disuelta de la especie k

t tiempo

x_{ij} distancia a lo largo de los ejes cartesianos

D_{ij} componente del tensor de dispersión hidrodinámica

v_i componente de la velocidad de poro $v_i = \frac{q_i}{\theta}$

q_s tasa de flujo volumétrico por unidad de volumen del acuífero que representa fuentes (positivo) y sumideros (negativo)

C_s^k concentración del flujo de la fuente o sumidero para la especie k

$\sum R_n$ término de reacción química

Se utiliza el *MODFLOW* (Mc Donald y Harbaugh, 1998) para resolver la ecuación de flujo y *MT3D* (Zheng, 1992) para resolver la ecuación de transporte.

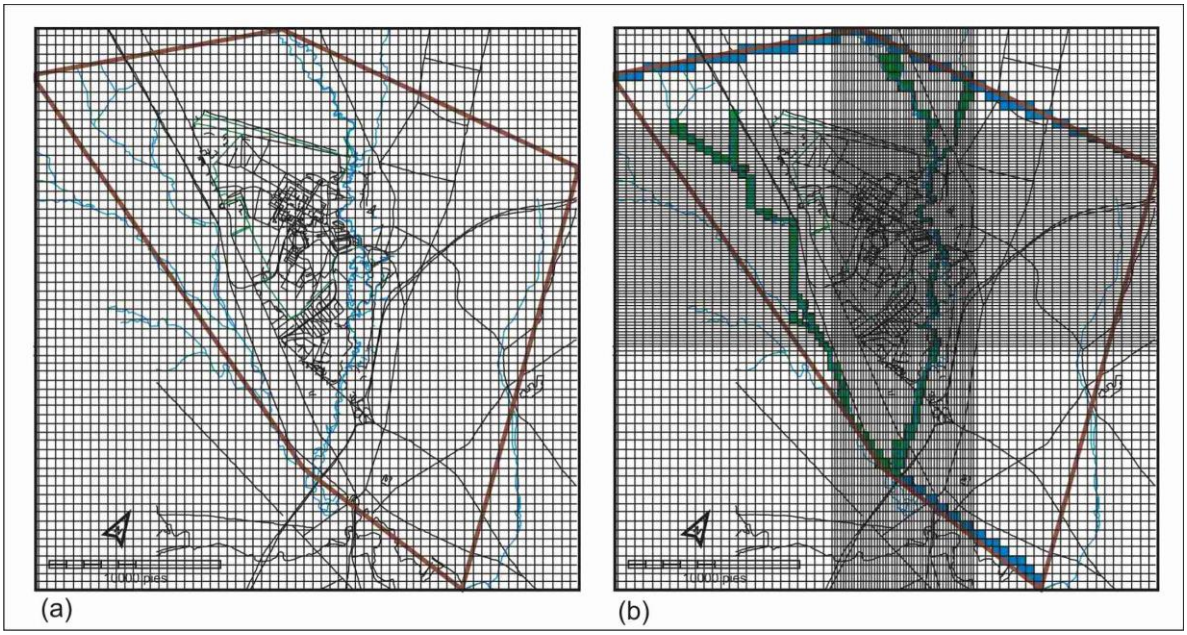


Figura 5. 10 a) Malla y dominio, b) submalla de simulación estocástica y las condiciones de frontera (celdas color azul) para el modelo de flujo en *MODFLOW* (modificada de McLane Environmental, 2010).

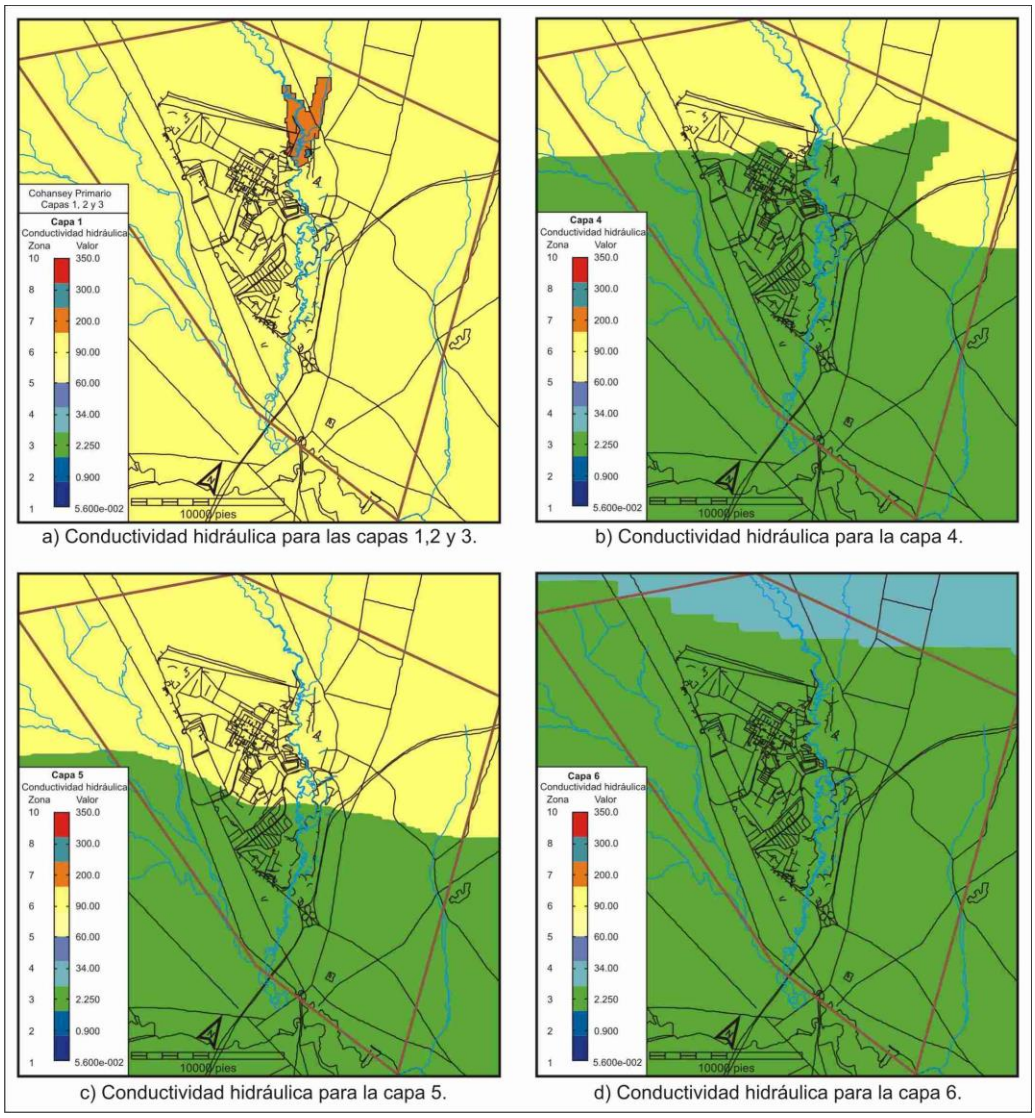


Figura 5. 11 Distribución de la conductividad hidráulica en las capas no homogéneas (modificada de McLane Environmental, 2010).

5.2.2 Modelo estocástico

El modelo estocástico está basado en el modelo de flujo y transporte determinista. La conductividad hidráulica en este caso se representa como un campo aleatorio en las capas 1, 2, 3, 5, 7 y 9; en las capas restantes (4, 6 y 8) se consideraron deterministas e iguales a las de las capas originales. En cada capa con conductividad aleatoria el valor de la media de esta variable se tomó igual al valor correspondiente de la del modelo determinista y la desviación estándar es proporcional a la media con la constante de proporcionalidad igual a 1.3513. En consecuencia, el campo de velocidades, el campo de dispersión y el campo de concentración resultante también se vuelven campos aleatorios.

En este caso el campo de conductividad hidráulica tiene una distribución lognormal, la estructura espacial del $\ln(K)$ se describe por medio un semivariograma esférico, es decir:

$$\gamma_F(h) = \begin{cases} 1.5\frac{h}{a} - 0.5\left(\frac{h}{a}\right)^3, & si \quad h \leq a \\ 1, & en\ cualquier\ otro\ caso \end{cases} \qquad (5.12)$$

Donde $F = \ln(K(x))$ con parámetros $C_0 = 0.25$, $C = 0.75$, $a_x = a_y = 799.87$ pies y $a_z = 20.0$ pies.

Al igual que en el caso anterior se usó el método llamado de simulación secuencial gaussiana (*Sgsim*) del paquete *GSLIB* (Deutsch y Journel, 1998) para obtener las realizaciones aleatorias de la conductividad hidráulica.

El modelo tiene dos periodos de simulación para el flujo, el primer periodo es de 360 días con 12 periodos de esfuerzo (cada paso de tiempo es de 30 días), el segundo periodo es de 1080 días con 36 períodos de esfuerzo (cada paso de tiempo es de 30 días).

5.2.3 Vector espacio temporal de medias y matriz de covarianza

Se calcula el vector espaciotemporal de las medias y la matriz de covarianza para estimar con el *ESH* la concentración del contaminante en la submalla marcada con un rectángulo en la figura 5.13, con los datos existentes para el periodo de dos años (se tienen 491 datos de clorobenceno para 144 pozos de monitoreo, ver figura 5.12). Las estimaciones de concentración se obtienen en los nodos de esta submalla. Para cada una de estas posiciones, las concentraciones se estiman mensualmente durante un período de dos años (48 veces), esto es, se obtiene una estimación en cada posición cada 121.7 días. Entonces se tienen por capa 3,600 puntos espaciotemporales de estimación (15×5 posiciones en cada capa $\times\ 48$ tiempos) en 6 capas (1, 2, 3, 5, 7, y 9), haciendo un total de 21,600 puntos espaciotemporales de estimación. De estos, 12 coinciden con posiciones donde se tienen datos, por lo que la dimensión del vector estado, y por lo tanto del vector de la media, es de 22,079 ($21{,}600 + 491 - 12$) y la dimensión de la matriz de covarianza es de 487, 482,241 ($22{,}079 \times 22{,}079$).

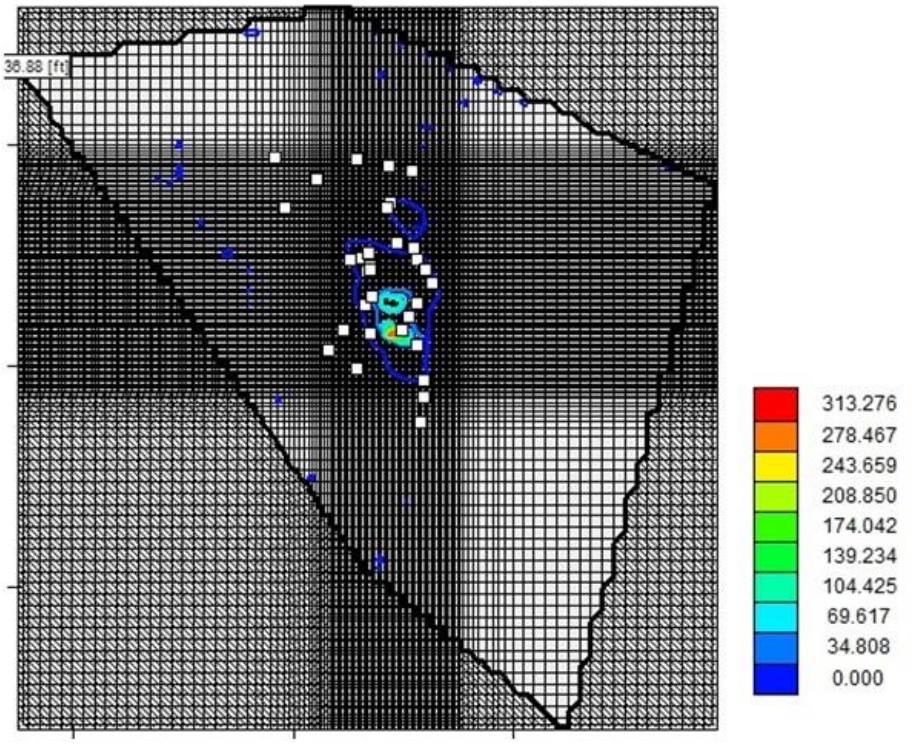


Figura 5. 12 Posiciones de observación.

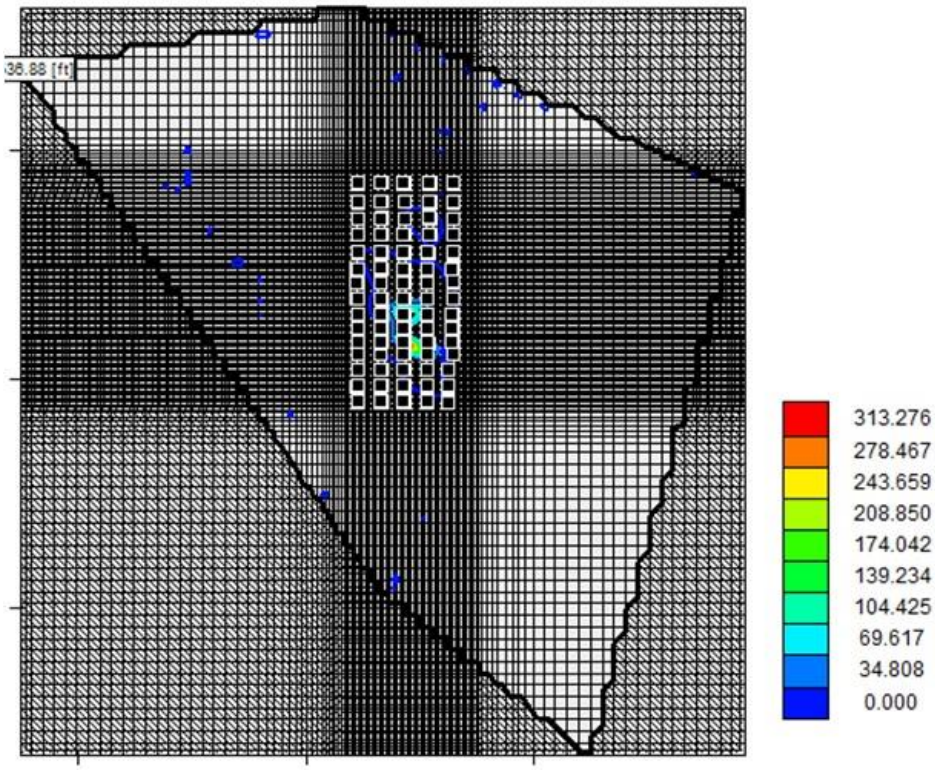


Figura 5. 13 Posiciones de estimación.

5.2.4 Resultados

Los códigos se ejecutaron en el mismo clúster que en el caso anterior. En la tabla 5.5 se pueden observar los resultados que se obtuvieron.

Realiz. / Proc.	Tiempo (seg)			Sp			Ep			Ley de Amdahl		
	1,000	2,000	4,000	1,000	2,000	4,000	1,000	2,000	4,000	1,000	2,000	4,000
1	68,914.74	117,053.35	250,129.74	1	1	1	1	1	1	1	1	1
2	45,969.19	65,714.61	125,990.42	1.49	1.78	1.98	0.74	0.89	0.99	1.87	1.92	1.94
4	30,267.46	39,481.68	69,839.64	2.27	2.96	3.58	0.56	0.74	0.89	3.12	3.31	3.46
8	20,715.26	27986.33	43,840.72	3.32	4.18	5.70	0.41	0.52	0.71	3.81	4.33	5.26
12	17,716.93	23,345.49	35,164.95	3.88	5.01	7.11	0.32	0.41	0.59	3.74	4.45	5.71
16	18,274.11	21,640.52	31,927.96	3.77	5.40	7.83	0.23	0.33	0.48	3.83	4.36	5.74
24	20,739.32	23,076.68	30,800.74	3.32	5.07	8.12	0.13	0.21	0.33	4.06	4.46	5.68
32	31,202.74	35,073.67	39,711.52	2.20	3.33	6.29	0.06	0.10	0.19	5.33	6.05	6.48

Tabla 5. 5 Datos de tiempo de ejecución (seg), aceleración (Sp), eficiencia (Ep) y ley de Amdahl con diferentes números de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso complejo.

En la figura 5.14 se observa la aceleración conforme aumenta el número de procesadores y en la figura 5.15 se muestra la eficiencia que se obtuvo para 1,000, 2,000 y 4,000 realizaciones en los diferentes números de procesadores. En el caso de 1,000 realizaciones la aceleración máxima que se obtuvo fue de 3.88 con 12 procesadores y la eficiencia correspondiente fue de 0.32; para el caso de 2,000 realizaciones la aceleración máxima que se obtuvo fue de 5.40 con 16 procesadores y la eficiencia correspondiente fue de 0.33; para el caso de 4,000 realizaciones se obtuvo una aceleración máxima de 8.12 con 24 procesadores y la eficiencia correspondiente fue de 0.33. Conforme aumenta el número de realizaciones, también aumenta el número de procesadores para la aceleración máxima, sin embargo, no tiene una buena eficiencia ya que no está arriba del 50%.

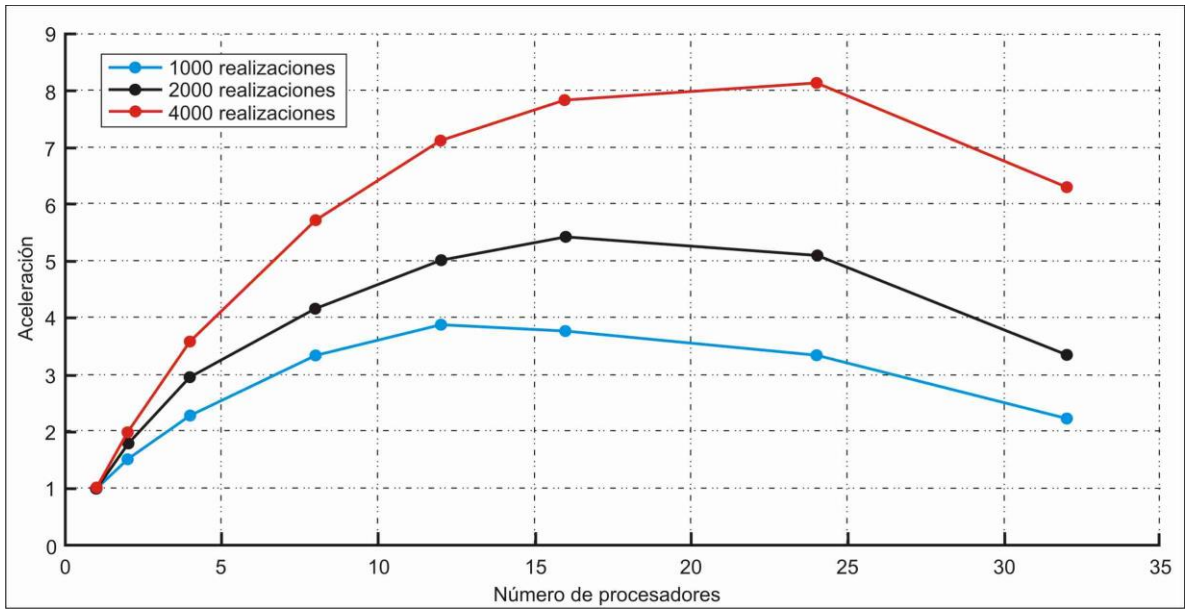


Figura 5. 14 Aceleración vs número de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso complejo.

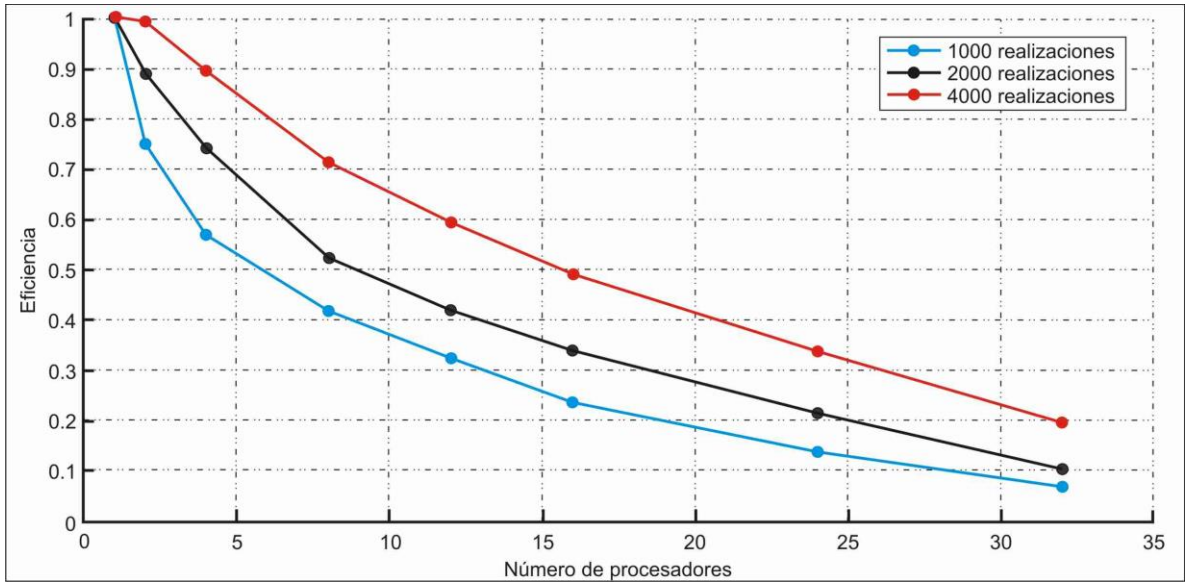


Figura 5. 15 Eficiencia vs número de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso complejo.

En la tabla 5.6 se muestran los cálculos del costo para este caso.

Realizaciones / Procesadores	Costo		
	1,000	**2,000**	**4,000**
1	68,914.74	117,053.35	250,129.74
2	91,938.39	131,429.22	251,980.85
4	121,069.86	15,7926.73	279,358.59
8	165,722.12	223,890.66	350,725.77
12	212,603.20	280,145.88	421,979.40
16	292,385.80	346,248.32	510,847.48
24	497,743.84	553,840.53	739,217.78
32	998,487.87	112,2357.6	1,270,768.74

Tabla 5. 6 Cálculos de los costos en diferentes números de procesadores para 1,000, 2,000 y 4,000 realizaciones para el caso complejo.

En la figura 5.16 se muestra el comportamiento del costo, para 1,000, 2,000 y 4,000 realizaciones, conforme aumenta el número de procesadores. Se puede observar que el comportamiento del costo es exponencial. Para 1,000 realizaciones se obtuvo una aceleración de 3.88 con 12 procesadores y un costo de 212,604.20; para 2,000 realizaciones se obtuvo una aceleración máxima de 5.4 con 16 procesadores y un costo de 346,248.32; para 4,000 realizaciones se obtuvo una aceleración máxima de 8.12 con 24 procesadores y un costo de 739,217.78.

Sin embargo, a pesar de obtener la aceleración máxima para 1,000, 2,000 y 4,000 realizaciones no se tiene la eficiencia esperada. La mejor eficiencia fue de 0.74 para 1,000 realizaciones con 2 procesadores y un costo de 91,938.39; para 2,000 realizaciones con 4 procesadores y un costo de 157,926.73, para 4,000 realizaciones con 8 procesadores se obtuvo una eficiencia de 0.71 y un costo de 350,725.77.

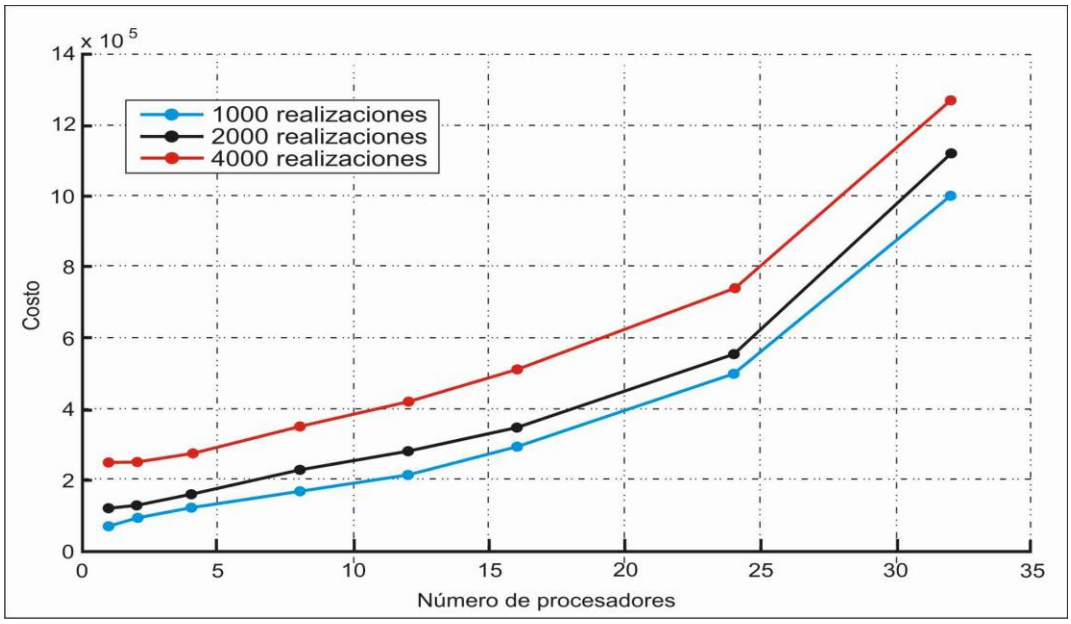


Figura 5. 16 Gráfica del costo para 1,000, 2,000 y 4,000 realizaciones para el caso complejo.

En la tabla 5.7 se muestran los cálculos de la memoria que se utiliza de acuerdo al número de procesadores.

Procesadores	Memoria RAM (GB)
1	3,899.85
2	7,799.71
4	15,599.43
8	31,198.86
12	46,798.29
16	62,397.72
24	93,596.59
32	124,795.45

Tabla 5. 7 Cálculos de la memoria RAM por matriz a utilizar en diferentes números de procesadores para el caso complejo.

En la figura 5.17 se muestra el comportamiento de la memoria RAM (Ec. 5.8) en diferentes números de procesadores.

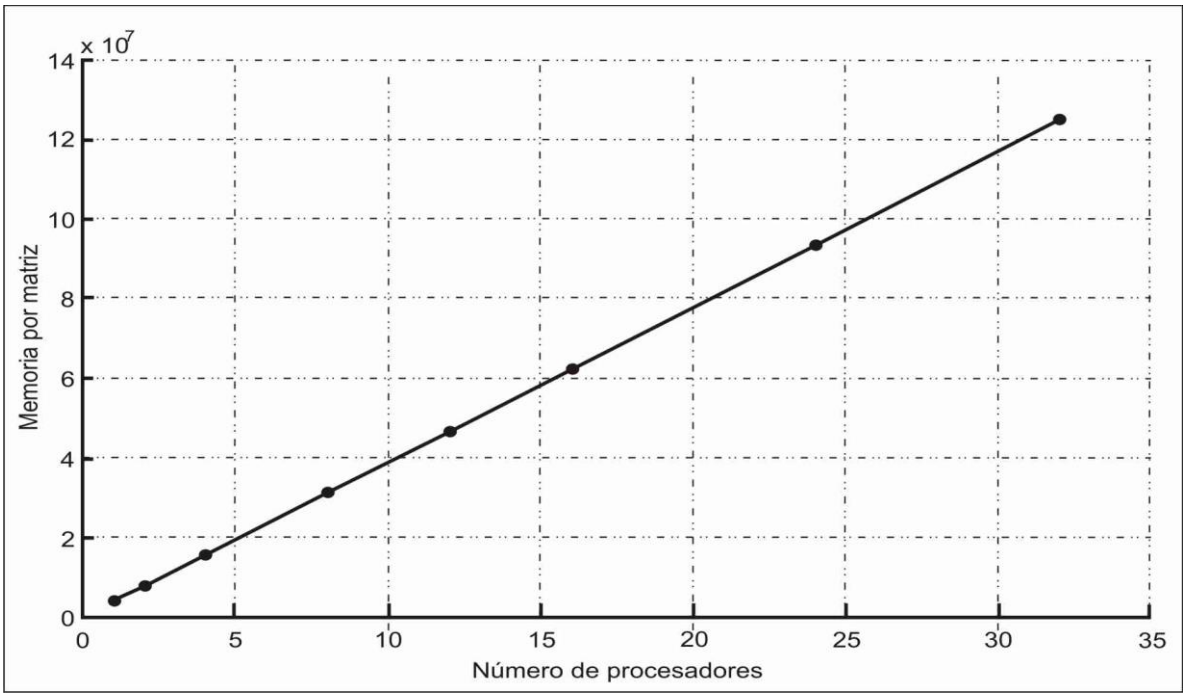


Figura 5. 17 Uso de memoria RAM por matriz en diferentes números de procesadores para el caso complejo.

5.2.5 Discusión

En la figura 5.18 se muestra la comparación de las aceleraciones para 1,000, 2,000 y 4,000 realizaciones con la ley de Amdahl para 4,000 realizaciones, y se observa que los resultados de las aceleraciones van acordes a las predicciones de esta ley hasta 12 procesadores, porque al aumentar los procesadores la aceleración ya no aumenta como se esperaría.

La fracción en serie que se utiliza para la ley de Amdahl se mide en unidades de tiempo, por lo que se esperaría que al aumentar el número de realizaciones los procesadores tuvieran que trabajar más en paralelo para reducir la fracción en serie como consecuencia, sin embargo, en este caso en específico no se logró ese objetivo.

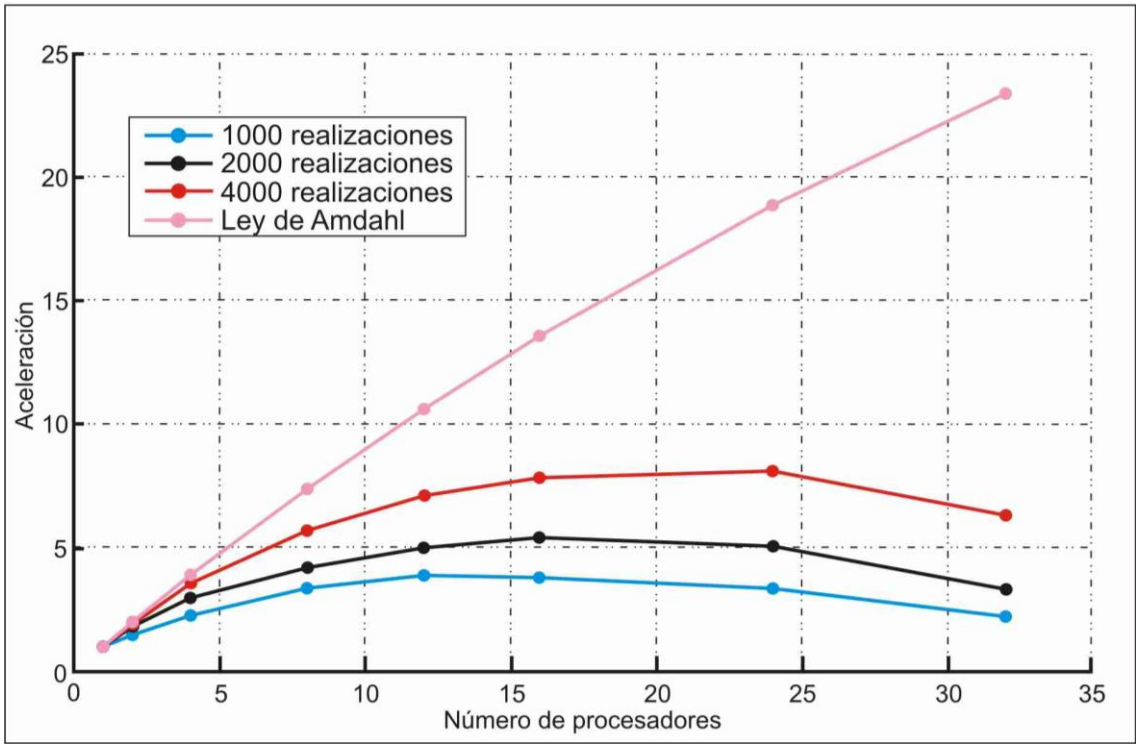


Figura 5. 18 La comparación entre la aceleración de 1,000, 2,000 y 4,000 realizaciones con la Ley de Amdahl para 4000 realizaciones para el caso complejo.

En la figura 5.17 se puede observar el comportamiento lineal que tiene la memoria RAM por matriz con diferentes números de procesadores, en este caso para un procesador la memoria RAM es de 3,899.85 GB y para 32 procesadores es de 124,795.45 GB.

Los comentarios respecto a la fracción en serie del código, las comunicaciones, las dependencias de datos también aplican para el *GWMC-MFMT*. Al igual que en el caso anterior, el uso de la memoria RAM crece linealmente con el número de procesadores, sin embargo, en este caso la matriz de covarianza es de dimensiones mayores por lo que el uso de la memoria RAM se incrementa aún más. Por otro lado, debido a que la matriz auxiliar de covarianza, el vector auxiliar de la media y el vector auxiliar de la pluma media se guardan en el disco duro, al aumentar el número de procesadores, el espacio de disco requerido y el tiempo de escritura son mucho mayores que en el caso anterior. Inicialmente se ejecutó el programa en nodos con 16 cores y 64 GB de RAM, sin embargo, el programa abortaba o en ocasiones en algunos cores se quedaba pasmado, debido a estas incidencias, se utilizaron los nodos g3_a ó g3_b con 32 cores y 256 GB de RAM, con los que se obtuvieron los datos que se presentan. El gran incremento en el uso de

memoria RAM (de un máximo de 36 GB en el caso sencillo a cerca de 125,000 GB en el caso complejo) pudiera explicar este comportamiento.

Número de realizaciones	Memoria RAM (GB)
1,000	176.63
2,000	353.26
4,000	706.52

Tabla 5. 8 Calculo de la memoria RAM para las realizaciones, para el caso complejo.

Al comparar las tablas 5.7 y 5.8, se puede observar que, en este caso, debido a la dimensión de las matrices de covarianza, la memoria requerida al almacenar 1,000, 2,000 y 4,000 realizaciones es menor que al almacenar las matrices auxiliares de covarianza para los diferentes números de procesadores.

6. Conclusiones

En el análisis del estado del arte, se encontró que van Leeuwen y Evensen (1996) desarrollaron un método de asimilación de datos muy similar al desarrollado por Herrera (1998) llamado ensamble suavizado. Por ese motivo, en esta tesis se diferenciaron como ensamble suavizado de van Leeuwen y Evensen (*ESLE*) y ensamble suavizado de Herrera (*ESH*). Una diferencia que existe entre estos métodos de asimilación de datos es que el *ESH* calcula y almacena la matriz de covarianza y deshecha las realizaciones una vez usadas, el *ESLE* almacena las realizaciones en memoria y con ellas calcula de manera indirecta la matriz de covarianza que no guarda.

De los trabajos revisados, el más relevante es el de Dong et al. (2012) que utilizan una metodología de paralelización similar a la que se presenta en este trabajo de tesis. Ellos aplican dos niveles de paralelización de grano grueso en el cálculo de MODFLOW y grano fino en la solución de PCG (Preconditioned Conjugate Gradient), en el sistema de procesamiento paralelo de Java (JPPF), logrando reducir el tiempo de ejecución en un orden de dos, sin embargo, esta técnica vale la pena sólo para cuadrículas muy grandes, más de 10^6 puntos. En nuestro trabajo solo se aplica una metodología de paralelización de grano grueso en los cálculos del GWMC, reduciendo el tiempo de ejecución, en el caso sencillo no requerimos de mallas tan grandes, sin embargo, en el caso complejo que es de mayores dimensiones, se podría tomar en cuenta la paralelización de grano fino, para la resolución de las ecuaciones de flujo y transporte, y conseguir mejores resultados de los que se obtuvieron.

Los resultados obtenidos en paralelo, en este trabajo de tesis son:

- ✓ Para el caso sencillo la aceleración crece a medida que aumenta la carga de trabajo para cada procesador. Obtuvimos una muy buena eficiencia de 0.63 para 4,000 realizaciones con 96 procesadores. El costo computacional presenta un comportamiento exponencial conforme incrementan los diferentes números de procesadores. Se utilizaron aproximadamente 36 GB de memoria RAM al ejecutarse el GWMC para 4,000 realizaciones en 96 procesadores, utilizando nodos con 16 cores y 64 GB de memoria RAM, sin ninguna complicación computacional.
- ✓ Para el caso complejo se utilizó la misma estrategia que en el caso de estudio sencillo, aunque por ser un problema de mayores dimensiones, se encontraron algunas dificultades. Debido a que se almacena en memoria la matriz auxiliar de covarianza correspondiente a cada

procesador y el tamaño de esta se mantiene constante, el uso de la memoria RAM crece linealmente con el número de procesadores y a que estas son mucho mayores que en el primer caso, las pruebas se tuvieron que ejecutar en nodos con mayor capacidad de memoria RAM (g3_a ó g3_b con 32 cores y 256 GB de *RAM*). Obtuvimos una buena eficiencia de 0.71 para 4,000 realizaciones con 8 procesadores. De lo que se concluye que la eficiencia es buena con un número muy pequeño de procesadores, aun así, este desempeño puede ser útil pues la ejecución en serie del *GWMC* para 4,000 realizaciones en un procesador toma 2 días con 21 horas, 28 minutos y 49.8 segundos y al ejecutarse el script paralelo de Python para 4,000 realizaciones en 8 procesadores, tarda 12 horas con 10 minutos y 40.7 segundos, reduciendo en aproximadamente 1/6 el tiempo de ejecución.

Hay al menos dos puntos que se pueden mejorar en el programa para que aumente la aceleración, y por ende la eficiencia. Cuando se utilizan varios procesadores para realizar los cálculos, en cada procesador se crea una matriz auxiliar del mismo tamaño de la matriz de covarianza, por ende, al incrementar el número de procesadores, aumentará la memoria por matriz de forma lineal, como se puede observar en las figuras 5.5 (caso sencillo) y 5.17 (caso complejo). Como se mencionó en la discusión del caso sencillo, al comparar las tablas 5.3 y 5.4, cuando el número de procesadores es mayor a 16 y el número de realizaciones es mayor a 4,000 convendrá almacenar las realizaciones en lugar de almacenar la matriz auxiliar, pues se ahorraría memoria RAM, sin embargo, para problemas de mayores dimensiones, como en el caso complejo, se requiere de menor memoria RAM al cambiar la estrategia y almacenar las realizaciones en vez de las matrices de covarianza, para todos los números de procesadores probados. Por esto se recomienda realizar algunas modificaciones al código del programa GWMC, para tener la opción de almacenar las realizaciones en memoria RAM, como lo hace Evensen (2003). Además, se recomienda modificar la parte en la que cada procesador realiza una copia de la matriz auxiliar en disco para que luego las lea el procesador maestro y construya la matriz de covarianza final (que es un proceso secuencial que aumenta el tiempo de ejecución de manera importante), para que esto se lleve a cabo en memoria RAM.

Referencias

Ababou, R. (1988). *Three-dimensional flow in random porous media, Ph. D. thesis.* Massachusetts: Massachusetts Institute of Technology, Department of Civil Engineering, Cambridge.

Aguilar, J., & Leiss, E. (2004). *Introducción a la computación paralela* (Primera ed.). Mérida - Venezuela: Universidad de los Andes.

Alzraiee, A. H., Gates, T. K., & Garcia, L. A. (2013). Modeling subsurface heterogeneity of irrigated and drained fields. II: Multivariate stochastic analysis of root-zone hydrosalinity and crop yield. *Journal of Irrigation and Engineering*, 809-820.

b_diaz. (s.f.). *Tipos de memoria RAM*. Recuperado el Agosto de 2018, de Tipos de memoria RAM: http://monografias.com/trabajos3/tiporam/tiposram.shtml

Babu, D., Pinder, G., Niemi, A., Ahlfed, D., & Sothoff, S. (1993). *Chemical transport by three-dimensional groundwater flows.* Princeton, N.J.: Tech. Rep. 84-WR-3, Dep. of Civ. Eng., Princeton Univ.

Bailey, R., & Baù, D. (2010). Ensemble smoother assimilation of hydraulic head and return flow data to estimate hydraulic conductivity distribution. *Water Resources Research, 46*, 1-19.

Bauser, G., Hendricks Franssen, H.-J., Kaiser, H.-P., Fritz Stauffer, U. K., & Kinzelbach, W. (2010). Real-time management of an urban groundwater well field threatened by pollution. *Environmental Science & Technology, 44*(17), 6802-6807.

Briseño-Ruíz, J. V. (2012). *Método para la calibración de modelos estocásticos de flujo y transporte en aguas subterráneas, para el diseño de redes de monitoreo de calidad del agua. Tesis doctoral.* México: Universidad Nacional Autónoma de México.

Briseño-Ruiz, J. V., Herrera-Zamarrón, G. d., & Júnez-Ferreira, H. E. (2011). Método para el diseño óptimo de redes de monitoreo de los niveles del agua subterránea. *Tecnología y Ciencias del Agua, II*(4), 77-96.

Burgers, G., van Leeuwen, P., & Evensen, G. (1998). Analysis scheme in the ensenble Kalman filter. *Monthly Weather Review*, 1719-1724.

Caché. (s.f.). *WIKIPEDIA*. Recuperado el Agosto de 2018, de WIKIPEDIA: https://es.wikipedia.org/wiki/Caché_(informática)

Castañeda Cárdenas, J. A., Nieto Arias, M. A., & Ortiz Bravo, V. A. (2013). Análisis y aplicación del filtro de Kalman en una señal con ruido aleatorio. *Scientia et Technica, 18*(1), 267-274.

Chen, Y., & Zhang, D. (2006). Data assimilation for transient flow in geologic formations via ensemble Kalman filter. *Advances in Water Resources, 29*, 1107-1122.

Crestani, E., Camporese, M., Baù, D., & Salandin, P. (2013). Ensemble Kalman filter versus ensemble smoother for assesing hydraulic conductivity via tracer test data assimilation. *Hydrology and Earth System Sciences, 17*, 1517-1531.

Dalcin, L. (2012). *MPI for Python, release 1.3*. Recuperado el Enero de 2014, de http://mpi4py.scipy.org/

Deutsch, C., & Journel, A. (1998). *Geostatistical software library and user's guide* (second edition ed.). New York: Oxford University Press.

Dong, Y., & Li, G. (2009). A parallel PCG solver for MODFLOW. *Ground Water, 47*(6), 845-850.

Dong, Y., Guomin, L., & Xu, H. (2012). Distributed parallel computing in stochastic modeling of groundwater systems. *Ground Water*, 1-5.

Evensen, G. (1994). Sequential data assimilation with a nonlinear quasigostrophic model using Monte Carlo methods to forecast error statistics. *J. Geophys. Res.*(99), 10143-10162.

Evensen, G. (2003). The ensemble Kalman filter: theoretical formulation and practical implementation. *Ocean Dynamics*, 343 - 367. doi:I0.1007/s10236-0003-0036-9

Evensen, G. (2009). *Data assimilation - The ensemble Kalman filter* (Second Edition ed.). New York: Springer - Verlag.

Evensen, G., & van Leeuwen, P. J. (2000). An ensemble Kalman smoother for nonlinear dynamic. *Monthly Weather Review*, 1852-1867.

Foster, I., Kesselman, C., & Steve, T. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *The International Journal of High Performance Computing Applications, 15*(3), 220-222.

García Carrasco, J. M. (s/a). *Herramientas para programación paralela.* Universidad de Castilla-La Mancha, Departamento de informática.

García Regis, O., & Cruz Martínez, E. (2003). *Paralelización en la supercomputadora "Cray Origin 2000".*

Giménez, D., Boratto, M., & Coelho, L. (s/a). *Capítulo 1. Programación paralela y distribuida.* Notas de clase.

Gropp, W., Lusk, E., & Skjellum, A. (1999b). *Using MPI-2: Advanced features of the message passing interface.* MIT Press. ISBN 0-262-57133-1.

Gunarathne, T., Wu, T.-L., Qiu, J., & Fox, G. (2011). Cloud computing paradigns for pleasingly parallel biomedical applications. *Concurrency and Computation: Practice and Experience, 23*, 2338-2354.

Herrera Olivares, G. (1998). *Cost effective groundwater quality sampling network design. Ph.D.* Vermont: Universidad de Vermont.

Herrera, G., & Pinder, G. (2005). Space-time optimization of groundwater quality sampling networks. *Water Resources Research, 41*(W12407, doi:10.1029/2004WR003626), 1-15.

Hidalgo Pérez, J., & Cervigón Rückauer, C. (2004). Una revisión de los algoritmos evolutivos y sus aplicaciones. *ResearchGate*, 1-16.

Jazwinski, A. (1970). *Stochastic processes and filtering theory.* New York: Academic Press.

Jiménez Castilla, J., & Pino Telleria, I. V. (s/a). *Lección 1. Introducción y clasificación.* Universidad José Carlos Mariátegui.

Jiménez-González, D. (s/a). *Introducción a las arquitecturas paralelas.* Universidad Oberta de Catalunya.

Jones, E., Oliphant, T., & Peterson, P. a. (2001). *SciPy: Open source scientific tools for Python.* Recuperado el Enero de 2014, de http://www.scipy.org

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering, 82*, 35 - 94.

Karniadakis, G., & Kirby II, R. (2003). *Parallel scientific computing in C++ and MPI. A seamless aproach to parallel algorithms and their impementation.* USA: Cambridge University Press.

Keppenne, C. (2000). Data Assimilation into a primitive-equation model with a parallel ensemble Kalman filter. *American Meteorological Society, 128*, 1971-1981.

Keppenne, C. L., & Rienecker, M. M. (2002). Initial testing of a massively parallel ensemble filter Kalman with Poseidon isopycnal ocean general circulation model. *American Meteorological Society, 130*, 2951-2964.

Keppenne, C., Rienecker, M., Kurkowski, N., & Adamec, D. A. (2005). Ensemble Kalman filter assimilation of temperature and altimeter data with bias correction and application to seasonal prediction. *Nonlinear Processes in Geophysics, 12*, 491-503.

Kikut V., A. (2003). *Uso del filtro de Kalman para estimar la tendencia de una serie.* Informe técnico, Departamento de División Económica.

Kirk, D. B., & Hwu, W. W. (2010). *Programming massively parallel processors: A hands-on approach.* (1rs ed.). CA, USA: Morgan Kaufmann Publishers Inc.

Kollat, J. B., Reed, P. M., & Maxwell, R. M. (2011). Many-objective groundwater monitoring network design using bias-aware ensemble Kalman filtering, evolutionary optimization, and visual analytics. *Water Resources Research, 47*, 1-18.

Kollat, J., Reed, P., & Rizzo, D. (2008). Addressing model bias and uncertainty in three dimensional groundwater transport forecast for a physical aquifer experiment. *Geophysical Research Letters*, 1-5.

Kurtz, W., Franssen, H.-J. H., Kaiser, H.-P., & Vereecken, H. (2014). Joint assimilation of piezometric heads and groundwater temperatures for improved modeling of river-aquifer interactions. *Water Resources Research, 50*, 1665-1688.

Leyva Suárez, E. (2010). *Acuíferos semiconfinados y su modelación: Aplicaciones al acuífero de la zona metropolitana de la ciudad de México. Tesis.*

Leyva-Suárez, E., Herrera, G. S., & de la Cruz, L. M. (2015). A parallel computing strategy for Monte Carlo simulation using groundwater models. *Geofísica Internacional, 54*(3), 245-254.

Liu, G., Chen, Y., & Zhang, D. (2008). Investigation of flow and transport processes at the MADE site using ensemble Kalman filter. *Advances in Water Resources, 31*, 975-986.

Lynn W., G. (1993). *Stochastic subsurface hydrology.* USA: PRETINCE HALL.

Madrid, U. E. (s/a). *Unidad de control segmentada. Procesamiento paralelo.* LAUREATE INTERNATIONAL UNIVERSITIES.

Mc Donald, M., & Harbaugh, A. (1998). *MODFLOW, a modular 3d finite difference ground-water flow model.* US Geological Survey.

McLane Environmental, L. (2010). *Classification exception area for Ciba-Geigy superfund site dover township.* New Jersey.

Montenegro García, A. (2005). *Introducción al filtro de Kalman.* Bogotá: Pontificia Universidad Javeriana. Facultad de Ciencias Económicas y Administrativas.

Muller Santa Cruz, H. (2007). *Programando en Fortran. Apuntes.*

Nowak, W., de Barros, F., & Rubin, Y. (2010). Bayesian geostatistical desing: Task-driven optimal site investigation when the geostatistical model is uncertain. *Water Resources Research, 46*, 1-17.

Olivares, J.-L. (2001). *Argus ONE-PTC interface, v. 2.2 User´s guide.* Burlington, Vermont: University of Vermont.

Olivares-Vázquez, J. (2002). *Groundwater quality monitor GUI. User's guide.*

Osses, A. (2008). Asimilación de datos y aplicaciones en ciencias atmosféricas. *Primer Encuentro de Radio-Astronomía y Meteorología en Valparaíso*.

RAM, M. (30 de Septiembre de 2014). *Significados.com*. Recuperado el Agosto de 2018, de Significados.com: https://www.significados.com/memoria-ram/

Renard, P. (2007). Stochastic hydrogeology: What professionals really need? *Groundwater*(5), 531-541.

Ridgway, S., Terry, C., & Babak, B. (2005). *Scientific parallel computing.* New Jersey: Princeton University Press.

Riechle, R., & Koster, R. D. (2003). Assessing the impact of horizontal error correlations in background fields on soil moisture estimation. *Journal of Hydrometeorology, 4*, 1229-1242.

Rostami, M. A., Bucker, H., Vogt, C., Seidler, R., Neuhäuser, D., & Rath, V. (2014). A distributed-memory parallelization of a shared-memory parallel ensemble Kalman filter. *16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, (págs. 455-462).

Solera Ramírez, A. (2003). *El filtro de Kalman.* Nota técnica, Banco Central de Costa Rica, Departamento de Inverstigaciones Económicas.

stothoff, S. (1996). *Princeton Transport Code Tutorial.* Burlington, Vermont: University of Vermont.

Stutz, M. (2001). *The Linux cookbook.* San Francisco: LINUX, JOURNAL PRESS.

Swap. (24 de Enero de 2018). *Significados.com*. Recuperado el Agosto de 2018, de Significados.com: http://www.significados.com/swap/

Tandeo, P. (2011). *Introducción a la asimilación de datos.*

van Hees, F., van der Steen, A. J., & van Leeuwen, P. J. (2003). A parallel data assimilation model for oceanographic observations. *Concurrency and Computation: Practice and experience, 15*, 1191-1204.

van Leeuwen, P. J., & Evensen, G. (1996). Data assimilation and inverse methods in terms of a probabilistic formulation. *Monthly Weather Review*, 2898-2913.

Welch, G., & Bishop, G. (2002). *An introduction to the Kalman filtro.* NC: University of North Carolina, Department of Computer Science. Chapel Hill.

Xu, T., Gómez-Hernández, J., Li, L., & Zhou, H. (2013). Parallelized ensemble Kalman filter for hydraulic conductivity characterization. *Computers & Geosciences, 52*, 42-49.

Zhang, Y., Pinder, G. F., & Herrera, G. S. (2005). Least cost design of groundwater quality monitoriong networks. *Water Resources Research, 41*, 1-12.

Zheng, C. (1992). *MT3D: A modular three-dimensional transport model for simulation of advection, dispersion and chemical reactions of contaminants in groundwater system.* SS Papadopulos & Associates.

Anexo I

A continuación, se describen partes del script en Python que se utilizó para el caso de estudio sencillo (*GWMC_PTC*) para paralelizar el proceso, descrito en la sección 4.2. Algo similar se hace para el GWMFMT3D.

1) Un primer paso es crear un directorio por cada procesador para facilitar la paralelización y el almacenamiento de la información. Se usó el rango del procesador para definir el nombre de cada directorio.

```
if os.path.isdir('%s' % t + '%d' % rank):
    shutil.rmtree('%s' % t + '%d' % rank)
os.mkdir('%s' % t + '%d' % rank)

os.chdir('%s' % t + '%d' % rank)
```

2) Se ponen en el directorio de cada procesador los cuatro archivos de entrada que se utilizarían para una corrida en serie, estos son: *gwmc.par*, *sgsim.par*, *nrm2log.par* y *randTS2.par*. Cada archivo contiene entradas para los códigos *rndcsim*, *sgsim*, *nrm2log* y *ranTS2*, respectivamente. Sin embargo, es necesario modificarlos para que cada procesador genere el mismo número de realizaciones. La modificación que se le hace al archivo *sgsim.par* es cambiar el número de realizaciones, como se muestra a continuación:

```
ofile = open("sgsim.par", 'w')
i=0
for line in lines:
    i+=1
    if i==21:
        ofile.write('%d \n' % local_realizaciones)
    elif i==25:
        j+=2
        ofile.write('%d \n' % j)
    else:
        ofile.write('%s' % line)


ofile.close()
```

3) Todos los archivos de entrada se copian en cada nodo del clúster para ejecutar los programas: *sgsim*, *nrm2log*, *randTS2* y *GWMC*. Una vez que realizada la copia de los archivos de entrada, se ejecuta cada uno de estos programas. El código ejecutable está escrito en *Fortran* y se ejecuta utilizando una llamada al sistema desde el script de *Python*.

```
os.system('./sgsim < sgsim.par > sgsim.OUTPY')
os.system('./nrm2log > nrm2log.OUTPY')
os.system('./randts2 > randts2.OUTPY')
os.system('./gwmc > gwmc.OUTPY')
```

4) Para el cálculo de la matriz de covarianza, utilizamos una barrera para asegurar que la información de los diferentes procesadores haya llegado a la memoria del procesador maestro que construye la matriz de covarianza. El cálculo de la matriz de covarianza se lleva a cabo después de recopilar esta información.

```
MPI.COMM_WORLD.Barrier()
if (rank==0):
   def checkfile(archivo):
     import os.path
     if os.path.isfile(archivo):
       os.system('rm %s'%archivo)

   os.chdir('/home/estherl/NV3-MPI/')
   checkfile('covarianza.out')
   checkfile('media.out')
   print("tiempo sgsim = %e " % (t2-t1))
   print("tiempo nrmlog = %e " % (t3-t2))
   print("tiempo randts2 = %e " % (t4-t3))
   print("tiempo gwmc = %e " % (t5-t4))
   os.system('./matriz %d' %size)
   os.system('mv meanvect*.out basura/')
   os.system('mv meanplume*.out basura/')

   os.system('mv covmatrx*.out basura/')
```

Observe que en el código de Python para saber en qué procesador se está trabajando, se le asigna la etiqueta “rank”, y al asingar “rank==0”, nos referimos al procesador maestro. Utilizando esta clasificación todos los procesadores en paralelo ejecutan los pasos 1 a 3. El paso 4 se lleva a cabo en el procesador maestro, donde se copian todos los archivos del vector auxiliar de la pluma media, el vector auxiliar medio y la matriz auxiliar de covarianza con la etiqueta del procesador

que procede para que el procesador maestro pueda construir la matriz de covarianza, el vector de la media y el vector de la pluma media. Después de realizar estos cálculos, el procesador maestro envía todos los archivos auxiliares a un directorio que se llama basura.