VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO DE CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"MODELO COMPUTACIONAL EN PARALELO DE FLUJO Y TRANSPORTE"

T E S I S
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIAS (COMPUTACIÓN)

PRESENTA:
AMED ANTONIO LEONES VILORIA

DIRECTOR DE TESIS: DR. LUIS MIGUEL DE LA CRUZ SALAS
POSGRADO DE CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

MÉXICO, D. F. MAYO 2013

Índice general

Índice de figuras

Índice de cuadros

1 Introducción

1.1. El contexto

La MMC (Modelación Matemática y Computacional) es una herramienta muy poderosa, comparable con la teoría y la experimentación, que nos proporciona respuestas rápidas y con buena precisión a problemas de gran reto. Por ejemplo, el estudio de explosión de estrellas supernovas, solo es posible mediante una simulación numérica en computadoras de alto desempeño. De igual manera, para entender los fenómenos que ocurren en procesos industriales, se puede recurrir a la MMC para obtener respuestas en tiempos cortos y con un presupuesto limitado.

En el caso del movimiento del agua subterránea así como el seguimiento de partículas de un contaminante en el subsuelo puede modelarse mediante un conjunto de ecuaciones diferenciales parciales. La búsqueda de soluciones a estas ecuaciones es solo posible mediante el uso de métodos numéricos, pues en la actualidad solo ha sido posible obtener soluciones analíticas para casos muy particulares y simplificados.

Los códigos computacionales fundamentados en el modelado matemático y numérico logran una aproximación eficiente a la solución buscada en especial para el problema del flujo y transporte multifásico de agua subterránea.

Los modelos de agua subterránea se emplean a menudo para predecir los efectos de cambios hidrológicos, sean naturales o artificiales, en el comportamiento de un acuífero. Se producen cambios artificiales, por ejemplo, por la extracción de agua para uso doméstico, industrial, y/o para regar. Estos modelos también son utilizados para predecir el efecto de la introducción del riego a base de un embalse o un bocatoma. Asimismo, los modelos se utilizan para evaluar los efectos de la contaminación química en la superficie del suelo que pueda infiltrar en el acuífero bajo la influencia de la

lluvia y/o lixiviación[1].

1.2. El problema

Actualmente existen varios softwares que permiten realizar simulaciones numéricas del agua subterránea. Ninguno de esto softwares hace uso del computo paralelo, y no es fácil implementar partes del código que aprovechen las últimas tecnologías de hardware, debido principalmente que se basan en lenguajes de programación de hace 30 años (Fortran 77). Particularmente, en el Instituto de Geofísica de la UNAM, el software MODFLOW continua en uso para generar modelos de flujo de agua subterránea y transporte de contaminantes. Debido a esto, no ha sido posible utilizar equipos de supercómputo para mejorar la precisión de las soluciones y reducir el tiempo de cálculo.

Como es bien conocido, en la actualidad el uso del cómputo paralelo es común debido a las nuevas arquitecturas de cómputo existentes en el mercado. Hoy en día, casi todas las computadoras, incluyendo laptops personales, tienen al menos 2 unidades de procesamiento. Esto hace que la comunidad científica se enfoque en el desarrollo de software que permita aprovechar este tipo de tecnologías. Por lo tanto, el desarrollo de un programa numérico, debería ser diseñado teniendo en cuenta que sería factible correrlo en paralelo. Además, las bibliotecas que se usan en la modelación matemática y computacional (MMC), contienen elementos para un fácil uso de arquitecturas multiprocesador. Por ejemplo, MPI[27] es una biblioteca de envío de mensajes que permite desarrollar códigos paralelos para sistemas de memoria distribuida. Los códigos así desarrollados podrán ser ejecutados en clústers con cientos de miles de procesadores. Por otro lado, OpenMP permite un desarrollo simple de aplicaciones paralelas, debido a que un código serial puede paralelizarse fácilmente con solo agregar directivas dentro del programa. Sin embargo, esto solo se puede correr en sistemas de memoria compartida, y actualmente las computadoras personales solo tienen hasta 6 unidades de procesamiento.

Otra tecnología que ha surgido impulsada por la industria de los videojuegos, son las tarjetas gráficas, las cuales llegan a tener hasta 448 unidades de procesamiento. Esto abre la puerta para desarrollar códigos masivamente paralelos. Sin embargo, existen otras limitantes como el ancho de banda entre el CPU y el GPU (Graphics Unit Processor), lo cual puede limitar el uso del GPU. Actualmente, muchos grupos de desarrollo están estudiando la posibilidad de usar GPUs en distintas áreas de la ciencia, con el objeto de reducir sus tiempos de ejecución.

1.3. El enfoque

La forma en que FEniCS paraleliza sus procesos, se basan principalmente en paralelizar la solución de los sistemas lineales. Para lograr esto, hace uso de PETSc, que

[1] Es un proceso en el que un disolvente líquido pasa a través de un sólido pulverizado para que se produzca la disolución de uno o más de los componentes solubles del sólido.

es un biblioteca que contiene varios algoritmos de solución basados en el subespacio de Krylov. Estos algoritmos son muy eficientes y en PETSc se paralelizan mediante la descomposición de datos, realizando operaciones de álgebra lineal parcialmente en varios procesadores. Aun cuando PETSc no es tan fácil de usar y optimizar como FEniCS, ha demostrado ser una de las bibliotecas más óptimas, llegando a resolver hasta 5×10^{11} incógnitas, y se ha ejectuado hasta en 224,000 procesadores. Dado que FEniCS se liga directamente con PETSc, haremos uso también de esta biblioteca en este trabajo.

El software para cómputo científico se ha desarrollado desde hace más de 40 años. Durante esta historia, se han generado bibliotecas que son de gran utilidad, de código abierto y libres para uso académico. Por ejemplo BLAS (Basic Linear Algebra Subprograms) [13], LAPACK[2] (Linear Algebra Package) [13], ATLAS (Automatically Tuned Linear Algebra Software) [13], estas tres bibliotecas son la base para el desarrollo de bibliotecas de más alto nivel, y se basan en lenguajes como Fortran 77 y C. Bibliotecas similares que usan paradigmas de programación actuales han aparecido en la última década. Por ejemplo uBLAS-Boost que provee la misma funcionalidad de BLAS usando C++ y templates [22]; MTL4 (Matrix Template Library) que provee herramientas para operaciones de álgebra lineal [22]; FLENS [13], Eigen [13] y Seldon [13], similares a MTL4, todas ellas escritas en C++ y que usan técnicas avanzadas basadas en templates para obtener alto rendimiento. Varias de estas bibliotecas se basan en BLAS, LAPACK y/o ATLAS.

La razón de la existencia de distintos tipos de bibliotecas de álgebra lineal, es que este tipo de operaciones aparecen en varias etapas en la solución de problemas científicos. Además, en la mayoría de los casos, estas operaciones son las que toman más tiempo durante la ejecución de un simulador. Por lo tanto, es necesario poner atención en esta parte para obtener códigos eficientes.

Uno de los objetivos en el desarrollo de software científico en los últimos años ha sido desarrollar software, que además de ser eficiente, sea fácil de usar y que represente fielmente las matemáticas que se implementan. Durante una búsqueda de software científico actual, se observó que hay varias bibliotecas que cumplen con estos requisitos. Sin embargo, dos de ellas llamaron nuestra atención que son FEniCS [22] y PETSc [2].

FEniCS permite desarrollar scripts en Python con los cuales se pueden resolver ecuaciones diferenciales parciales, mediante el uso del método de elemento finito (MEF). La novedad en esta biblioteca es que es posible implementar directamente las formulaciones débiles del problema, sin tener que pasar por definiciones complejas en el software. Esto permite al usuario, con conocimientos básicos en el MEF, concentrarse en las matemáticas y fenomenología de lo que quiere simular, sin distraerse en los detalles de la programación.

[2]La primera versión se llamó LINPACK y esta sigue usándose para clasificar las supercomputadoras más poderosas del mundo. Véase www.top500.com.

Desde nuestro punto de vista, esto es un gran paso en el cómputo científico. Durante este trabajo se presentarán ejemplos de FEniCS en donde se hará uso de esta característica, y además se resolverá un problema de contaminación de acuíferos. El mismo código puede ejecutarse en serial y en paralelo, por lo tanto en este trabajo se usará esto último para determinar la eficiencia y aceleración, cuando se ejecuta el software en un clúster de alto desempeño.

1.4. Objetivos, metas y relevancia

Tomando en cuenta lo descrito en las secciones anteriores, se plantean los siguientes objetivos y metas. Además se explica la relevancia del trabajo.

1.4.1. Objetivo

Desarrollar modelos matemáticos, numéricos y computacionales de flujo y transporte, para estudiar la dinámica del agua subterránea, así como del transporte de contaminantes en casos de estudio existentes.

1.4.2. Metas

Entre las metas que se plantean se tienen las siguientes:

1. Desarrollar modelos matemáticos y numéricos del flujo de agua subterránea y transporte de contaminantes en acuíferos.
2. Desarrollar códigos que se puedan entender, modificar y ejecutar en diferentes plataformas de cómputo.
3. Desarrollar códigos que hagan uso eficiente de arquitecturas paralelas.
4. Resolver problemas típicos de flujo de agua subterránea para calibrar los códigos.
5. Resolver un problema de transporte de contaminantes en acuíferos.
6. Realizar un estudio de aceleración y eficiencia de los problemas anteriores para determinar la viabilidad del uso de hardware multiprocesador.
7. Difundir el código desarrollado para que sea evaluado y en su caso, utilizado por investigadores de hidrología subterránea.

1.4.3. Relevancia y contribución del trabajo

La meta del trabajo de tesis es resolver un problema de flujo y transporte de contaminantes en agua subterránea por los métodos de elemento finito y volumen finito, además de paralelizar el código propio que se desarrolle.

Simular este tipo de procesos ayudaría a generar un conjunto de herramientas para afrontar iniciativas de calidad del agua, suministro de aguas subterráneas y protección

de aguas de manantial. Además, el resultado de esta tesis consistiría de rutinas de programación, con los cuales sería posible apoyar diferentes trabajos de investigación.

1.5. Estructura de la tesis

El contenido de esta tesis está dividido en cinco capítulos, además de esta introducción.

En el capítulo 3 se describen los modelos matemáticos que se usarían para simular el flujo de agua subterránea. Esta descripción se hace desde el punto de vista de la formulación axiomática, introducida por el Dr. Ismael Herrera en un par de libros, véase [15][1].

Debido a la complejidad de los modelos matemáticos y a la no existencia de soluciones analíticas en la mayoría de los casos, en el capítulo 2 se presentan los métodos numéricos que usaremos para encontrar soluciones aproximadas. Estos métodos son el de volumen finito (MVF) y el de elemento finito. En el primer caso se usará la biblioteca TUNAM [11][10] que implementa este método y está basada en el uso intensivo de templates de C++. En el segundo caso se usará FEniCS.

En el capítulo 2, además, se da una breve descripción del software actualmente usado en aguas subterráneas, y del software que usaremos así como los modelos de paralelismo que utilizan.

En el capítulo 4 se presentan ejemplos de solución de casos conocidos y de un caso de contaminación de un acuífero. En este último caso, se hace un estudio de aceleración y eficiencia para ejecuciones del código con varios procesadores.

En el capítulo 6 se dan las conclusiones de este trabajo.

2 Antecedentes

2.1. Modelo matemático

2.1.1. Formulación Axiomática

Para obtener el modelo de flujo y el modelo de transporte para el flujo de agua subterránea nos basaremos en la *formulación axiomática* la cual es introducida por Herrera et al. [1, 16] y que describiremos a continuación.

De manera general la formulación axiomática plantea, identificar las llamadas *propiedades intensivas y extensivas* y establecer una relación entre ellas a manera de *balance* dentro de un volumen $B(t)$, o cuerpo, dentro de un medio continuo.

Propiedades Intensivas

En los sistemas continuos, se considera que los *cuerpos* llenan completamente el espacio físico que ocupan. Un cuerpo $\mathcal{B}$ es un conjunto de partículas que ocupa un dominio $B(t)$ en el intervalo $t \in (-\infty, \infty)$ [1, 16].

Para sistemas dinámicos, las partículas cambian de posición conforme evoluciona el tiempo, por lo que es necesario identificar a la partícula para los diferentes lugares en los diferentes tiempos.

Para esto, definimos a $\underline{p}(\underline{X}, t)$ como la función de posición de la partícula X en un tiempo t, donde $\underline{X}$ es el vector de coordenadas referidas a las *coordenadas materiales* de la partícula. Usamos la notación $\underline{x}$ para referirnos a las coordenadas de la posición de la partícula en el espacio físico

$$\underline{x} = \underline{p}(\underline{X}, t)$$

Entonces, consideramos una propiedad intensiva a una propiedad descrita para cada partícula de un cuerpo para cada instante de tiempo t[1, 16]. Escribimos

$$\phi\left(\underline{X}, t\right)$$

la cual es la *Representación Lagrangiana* de la propiedad intensiva. También podemos escribir

$$\psi\left(\underline{x}, t\right)$$

la *Representación Euleriana* de la propiedad intensiva del punto $\underline{x}$ en el espacio físico para un tiempo t.

Ambas representaciones cumplen la siguiente propiedad

$$\phi\left(\underline{X}, t\right) \stackrel{def}{=} \psi\left(\underline{p}\left(\underline{X}, t\right), t\right) = \psi\left(\underline{x}, t\right) \tag{2.1}$$

o

$$\psi\left(\underline{x}, t\right) \stackrel{def}{=} \phi\left(\underline{p}^{-1}\left(\underline{x}, t\right), t\right) \tag{2.2}$$

La derivada con respecto al tiempo de la representación Lagrangiana se conoce como *derivada material* y viene dada como

$$D\psi/Dt = \partial\psi/\partial t + \underline{v} \cdot \nabla\psi \tag{2.3}$$

Propiedes extensivas

Consideramos una propiedad extensiva cuando una función o propiedad está definida para cada cuerpo del sistema continuo[1, 16].

Se dice que es extensiva cuando la propiedad se puede escribir como una integral de la propiedad intensiva en el cuerpo

$$E\left(\mathcal{B}, t\right) \stackrel{def}{=} \int_{B(t)} \psi\left(\underline{x}, t\right) d\underline{x} \tag{2.4}$$

Balance global de ecuaciones

Los modelos básicos de los sistemas continuos consisten en un balance de ecuaciones de sus propiedades intensivas y extensivas.

El cambio en la propiedad extensiva por unidad de tiempo lo podemos escribir de la siguiente manera

$$\frac{dE}{dt}\left(t\right) = \int_{B(t)} g\left(\underline{x}, t\right) d\underline{x} + \int_{\partial B(t)} q\left(\underline{x}, t\right) d\underline{x} \tag{2.5}$$

donde g representa la cantidad de la propiedad extensiva que entra al cuerpo por el punto $\underline{x}$ por unidad de tiempo y por unidad de volumen; q representa la cantidad de la propiedad extensiva que entra al cuerpo a través de la frontera en el punto $\underline{x}$ por unidad

de tiempo y por unidad de área.

En condiciones generales escribimos q como

$$q\left(\underline{x},t\right) \overset{def}{=} \underline{\tau}\left(\underline{x},t\right)\cdot\underline{n}\left(\underline{x},t\right) \tag{2.6}$$

donde $\underline{n}$ es el vector normal unitario en $\partial B\left(t\right)$ que apunta hacía afuera del cuerpo. El vector $\underline{\tau}$ representa el flujo de la propiedad extensiva.

Tomando en cuenta esta definición podemos escribir el cambio en la propiedad extensiva por unidad de tiempo como

$$\frac{dE}{dt}\left(t\right)=\int_{B(t)} g\left(\underline{x},t\right)d\underline{x}+\int_{\partial B(t)}\underline{\tau}\left(\underline{x},t\right)\cdot\underline{n}\left(\underline{x},t\right)d\underline{x} \tag{2.7}$$

esta última ecuación es la *Ecuación General de Balance Global*.

Ecuaciones de Balance Local

Denotamos ahora con $\Sigma\left(t\right)$ a las discontinuidades de salto, es decir, superficies donde las propiedades intensivas son discontinuas.

Ahora bien la Ecuación de Balance Global podemos introducirla tomando en cuenta también las discontinuidades de salto[1, 16] de la siguiente manera

$$\frac{dE}{dt}\left(t\right)=\int_{B(t)} g\left(\underline{x},t\right)d\underline{x}+\int_{\partial B(t)}\underline{\tau}\left(\underline{x},t\right)\cdot\underline{n}\left(\underline{x},t\right)d\underline{x}+\int_{\Sigma(t)} g_{\Sigma}\left(\underline{x},t\right)d\underline{x} \tag{2.8}$$

donde g_{Σ} representa una fuente externa que está concentrada en Σ. Generalizado la aplicación del teorema de Gauss (véase Herrera [1, 16]), podemos escribir la ecuación de balance global como

$$\frac{dE\left(t\right)}{dt}=\int_{B(t)}\left\{g\left(\underline{x},t\right)+\nabla\cdot\underline{\tau}\left(\underline{x},t\right)\right\}d\underline{x}+\int_{\Sigma(t)}\left\{\left[\left[\underline{\tau}\right]\right]\cdot\underline{n}\left(\underline{x},t\right)+g_{\Sigma}\left(\underline{x},t\right)\right\}d\underline{x} \tag{2.9}$$

Sea $\psi\left(\underline{x},t\right)$, la propiedad intensiva que corresponde a la propiedad extensiva $E\left(t\right)$ y despreciando las discontinuidades por salto, podemos escribir la ecuación de balance local como

$$\frac{\partial\psi}{\partial t}+\nabla\cdot\left(\underline{v}\psi\right)=\nabla\cdot\underline{\tau}+g \tag{2.10}$$

Esta última ecuación puede todavía simplificarse más dependiendo de las condiciones propias del sistema que se está analizando.

Por ejemplo para un estado estacionario $\partial\psi/\partial t=0$

$$\nabla\cdot\left(\underline{v}\psi\right)=\nabla\cdot\underline{\tau}+g \tag{2.11}$$

y si $g = 0$ y no hay difusión $\underline{\tau} = 0$

$$\nabla \cdot (\underline{v}\psi) = 0 \tag{2.12}$$

Modelación Matemática de sistemas multifase

Para construir modelos de sistemas multifase[7] hay que modificar las ecuaciones globales y locales obtenidas anteriormente.

Los pasos son los siguientes:

1. Identificar la familia de propiedades extensivas.
2. Cada propiedad extensiva de la familia es asociada con una de las fases del sistema
3. Las condiciones de balance son aplicadas por cada propiedad extensiva de la familia, usando la velocidad de la partícula correspondiente a cada fase.

La aplicación de estos pasos, en el modelo matemático básico del sistema multifásico lleva al siguiente sistema de ecuaciones diferenciales

$$\frac{\partial \psi^{\alpha}}{\partial t} + \nabla \cdot (\underline{v}^{\alpha}\psi^{\alpha}) = \nabla \cdot \underline{\tau}^{\alpha} + g^{\alpha} \tag{2.13}$$

$\forall \alpha = 1, \ldots, N$

donde α representa una fase y N es el número total de fases del sistema.

Ahora se debe definir las ecuaciones constitutivas y auxiliares (frontera e inicial).

Esta información complementaria acerca de los sistemas, juega un papel muy importante, ya que es el medio por el que, el conocimiento científico y tecnológico, sobre una sistema particular se incorpora en el modelo.

2.1.2. Condiciones Iniciales y de frontera

Dado un problema [7] que involucre derivadas parciales sobre un dominio Ω, si la solución existe, esta no es única ya que generalmente este tiene un número infinito de soluciones. Para que el problema tenga una y sólo una solución es necesario imponer condiciones auxiliares, las llamadas condiciones iniciales y condiciones de frontera.

Condiciones Iniciales

Las condiciones iniciales expresan el valor de una función para un tiempo inicial, $t = 0$ (o se puede fijar a t en cualquier valor)

$$u\left(\underline{x}, 0\right) = \gamma\left(\underline{x}\right) \tag{2.14}$$

Condiciones de Frontera

Las condiciones de frontera indican los valores que la función puede tomar en la frontera $\partial\Omega$ y puede ser de tres tipos:

Dirichlet

El valor que la función $u\left(\underline{x},t\right)$ toma en la frontera $\partial\Omega$

$$u\left(\underline{x},t\right)=\gamma\left(\underline{x}\right) \tag{2.15}$$

En el caso de la formulación basada en la presión, la condición de primer tipo es una presión determinada, también llamada *Dirichlet*

$$p\left(\underline{x},t\right)=p_0\left(\underline{x},t\right) \qquad \underline{x}\in\partial\Omega_1 \tag{2.16}$$

Neumann

El segundo tipo, también llamado condición *Neumann* se conoce el valor de la derivada de la función $u\left(\underline{x},t\right)$ con respecto a la normal $\underline{n}$ a lo largo de la frontera $\partial\Omega$

$$\nabla u\left(\underline{x},t\right)\cdot\underline{n}=\gamma\left(\underline{x}\right) \tag{2.17}$$

es usualmente presentada como un flujo, es decir

$$\underline{q}_0\cdot\underline{n}=-\frac{\underline{\underline{k}}}{\mu}\cdot\left(\nabla p+\rho\hat{g}\nabla z\right)\cdot\underline{n} \tag{2.18}$$

o reordenando

$$\nabla p\left(\underline{x},p\right)\cdot\underline{n}=-\mu\underline{\underline{k}}^{-1}\underline{q}_0\left(\underline{x},t\right)\cdot\underline{n}-\rho\left(\underline{x},t\right)\hat{g}\nabla z\cdot\underline{n} \qquad \underline{x}\in\partial\Omega_2 \tag{2.19}$$

y para la condición común de ausencia de flujo a través de las fronteras

$$\nabla p\left(\underline{x},p\right)\cdot\underline{n}=-\rho\left(\underline{x},t\right)\hat{g}\nabla z\cdot\underline{n} \tag{2.20}$$

Carga Hidraúlica

La condición de tercer tipo es raramente usada en la formulación de la presión y no se presentará.

La formulación alternativa para las ecuaciones de flujo saturado utiliza el concepto de *carga hidráulica* h^w como la variable de estado desconocida para la cual se busca una solución.

El caso de la carga constante es una condición de primer tipo o condición de Dirichlet de la forma

$$h^w\left(\underline{x},t\right)=h_0^w\left(\underline{x},t\right) \tag{2.21}$$

2.2. Modelo Numérico

Una parte de este trabajo es mostrar un planteamiento de como transformar un sistema de ecuaciones diferenciales en un sistema de ecuaciones lineales. Un sistema lineal de ecuaciones tiene la forma

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n &= b_1 \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2n}x_n &= b_2 \\ &\vdots \\ A_{n1}x_1 + A_{n2}x_2 + \cdots + A_{nn}x_n &= b_n \end{aligned} \tag{2.22}$$

donde los coeficientes A_{ij} y las constantes b_j son conocidos y x_i representa las incógnitas [13]. En notación matricial las ecuaciones se pueden escribir como

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{2.23}$$

también es pertinente recordar[1] cuando una matriz es llamada definida positiva, ya que lo usaremos en la mayoría de los métodos.

Una matriz A de $n \times n$ elementos, se dice que es *definida positiva* si su parte simétrica

$$\left(A + A^T\right)/2 \tag{2.24}$$

es positiva definida. Esto es equivalente a la propiedad

$$x^T A x > 0 \tag{2.25}$$

para todo vector real no nulo, x. Una matriz es definida positiva si el determinante de todos sus menores son positivos [25]. Los menores principales de A son n matrices cuadradas. Se requiere que

$$A_{11} > 0, \qquad \begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix} > 0, \qquad \begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix} > 0, \ldots, |A| > 0 \tag{2.26}$$

2.2.1. Métodos de solución

Hay métodos para resolver sistemas de ecuaciones lineales: *métodos directos y métodos iterativos*[25].

La característica común de los *métodos directos* es que transforman las ecuaciones

[1] Usamos ^{T} para la transposición y $^{-1}$ para la matriz inversa.

originales en *ecuaciones equivalentes* (ecuaciones que tienen la misma solución) que pueden ser resueltas más fácilmente.

Los *métodos iterativos* o *métodos indirectos*, comienzan con una suposición de la solución x y luego repetidamente refinan la solución hasta que se alcanza cierto criterio de convergencia [13]. Éste tipo de métodos tiene importantes ventajas computacionales si la matriz de coeficientes es muy grande o es dispersa (la mayoría de los coeficientes son cero).

2.2.2. Visión general de los métodos

Presentaremos una descripción de cada uno de los métodos que se debatirán junto con algunas notas breves en términos de la clase de matrices para las que son más apropiados.

- Métodos estacionarios

  Jacobi — El método de Jacobi está basado en resolver para cada variable localmente con respecto a otras variables; una iteración del método corresponde a resolver para cada variable una vez. El método es fácil de entender y aplicar, pero la convergencia es lenta.

  Gauss-Seidel — El método de Gauss-Seidel es como el método de Jacobi, excepto que usa valores actualizados tan pronto estén disponibles. En general, si el método de Jacobi converge, el método de Gauss-Seidel convergerá más rápido que el método de Jacobi, aunque todavía es relativamente lento.

  SOR — Sobre relajación sucesiva (Successive Overrelaxation) puede ser derivado de el método de Gauss-Seidel al introducir un parámetro ω. Una elección óptima de ω, haría a SOR converger por un orden de magnitud más rápido que Gauss-Seidel.

  SSOR — Sobre relajación simétrica sucesiva (Symmetric Successive Overrelaxation) no tiene ninguna ventaja sobre SOR, sin embargo, es útil como un preacondicionador para métodos no estacionarios.

- Métodos no estacionarios

  CG — Gradiente conjugado (Conjugate Gradient) deriva su nombre del hecho de que genera una secuencia de vectores conjugados (u ortogonales). Estos vectores son los residuos de las iteraciones. También son los gradientes de una función escalar, la minimización que es equivalente a resolver un sistema lineal. CG es un método muy eficaz cuando la matriz de coeficientes es simétrica y definida positiva, ya que se requiere el almacenamiento el almacenamiento para un limitado número de vectores.

MINRES&SYMMMLQ Minimum Residual y Symmetric LQ son alternativas computacionales para CG para matrices que son simétricas pero que no son definidas positivas. SYMMLQ generará la misma solución iterando como CG si la matriz de coeficientes es simétrica definida positiva.

CGNE&CGNR Gradiente conjugado en ecuaciones normales, está basado en la aplicación de el método de CG a una de las dos formas de las *ecuaciones normales* para $Ax = b$. CGNE resuelve el sistema $\left(AA^T\right) y = b$ para y y entonces calcula la solución $x = A^T y$. CGNR resuelve $\left(A^T A\right) x = \overline{b}$ para el vector de solución x donde $\overline{b} = A^T b$. Cuando la matriz de coeficientes A es no simétrica y no singular, las matrices de las ecuaciones normales AA^T y $A^T A$ serán simétricas y definidas positivas y por lo tanto se podrá aplicar CG. La convergencia podría ser lenta, ya que la gama de matrices de ecuaciones normales será menos favorable que la gama de A.

GMRES El método generalizado de MINRES calcula una secuencia de vectores ortogonales (como MINRES) y combina éstos a través de resolver y actualizar mínimos cuadrados. A diferencia de MINRES (y CG) requiere almacenar toda la secuencia, de modo que una gran cantidad de almacenamiento es necesaria. Por esta razón, renovadas versiones de éste método son usadas. En versiones renovadas, los costos de cálculo y almacenamiento están limitadas por especificación a un número fijo de vectores que se generen. Este método es útil para matrices no simétricas.

BiCG El método de gradiente biconjugado genera dos secuencias de vectores como CG, una basada en un sistema con los coeficientes originales de la matriz A y otro con A^T. En lugar de ortogonalizar cada secuencia, se hace mutuamente ortogonales o "bi-ortogonales". Este método, como el de CG, usa almacenamiento limitado. Es útil cuando la matriz es no simétrica y no singular, sin embargo, la convergencia podría ser irregular y hay una posibilidad de que el método falle. BiCG requiere una multiplicación con la matriz de coeficientes y con su transpuesta en cada iteración.

QMR (Quasi-Minimal Residual). El método aplica mínimos cuadrados para resolver y actualizar a los residuos de BiCG y así suavizar el comportamiento irregular de convergencia de BiCG. QMR evita en gran medida las fallas que pueden ocurrir en BiCG. Por otra parte, no se realiza una verdadera minimización del error o el residuo y mientras converge suavemente, no es una mejora esencial en BiCG.

CGS (Conjugate Gradient Squared). Es una variante de BiCG que aplica la actualización de operaciones para la secuencia A y la secuencia A^T como a los mismos vectores. Idealmente, esto sería el doble de la tasa de convergencia, pero en la práctica la convergencia puede ser mucho más irregular que BiCG. Una ventaja práctica del método es

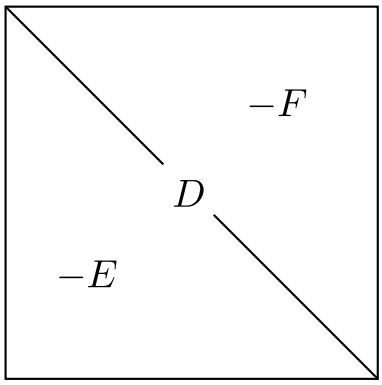


Figura 2.1: Particionamiento inicial de la matriz A

que no necesita las multiplicaciones con la transpuesta de la matriz de coeficientes.

Bi-CGSTAB El método de Gradiente Biconjugado Estabilizado es una variante de BiCG, pero usando diferentes actualizaciones en orden para la secuencia A^T para obtener una convergencia más suave que CGS.

Iteración Chebyshev. La iteración Chebyshev recursivamente determina polinomios con coeficientes escogidos para minimizar la norma de el residuo en un sentido min-max. La matriz de coeficientes debe ser definida positiva y es necesario el conocimiento de los valores propios. Este método tiene la ventaja que no requiere productos internos.

2.2.3. Jacobi, Gauss-Seidel, SOR y SSOR

A una matriz de coeficientes reales de $n \times n$ y un vector de n coeficientes reales, el problema considerado es encontrar x que pertenece a $\mathbb{R}^n$ tal que

$$Ax = b \tag{2.27}$$

Los métodos siguientes implican pasar de una iteración a la siguiente mediante la modificación de uno o unos pocos componentes de la solución a la vez[25]. Esto es natural, ya que hay criterios sencillos cuando se modifica un componente con el fin de mejorar una iteración. Un ejemplo es anular alguno componentes de el vector residual $b - Ax$. Comenzamos por descomponer A en

$$A = D - E - F \tag{2.28}$$

en donde D es la diagonal de A, $-E$ su estricta parte inferior, $-F$ su estricta parte superior como se ve en la figura 2.1, siempre asumiendo que las entradas de la diagonal sean siempre diferentes de cero. La iteración de Jacobi determina el i-ésimo componente de la siguiente aproximación y también anular el i-ésimo componente del vector

residual.

Con $\xi_i^{(k)}$ denotamos el i-ésimo componente de la iteración x_k y β_i el i-ésimo componente del vector del lado derecho, b. Por lo tanto, escribimos

$$(b - Ax_{k+1})_i = 0 \tag{2.29}$$

en el cual $(y)_i$ representa el i-ésimo componente del vector y, nos da

$$a_{ii}\xi_i^{(k+1)} = -\sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}\xi_j^{(k)} + \beta_i \tag{2.30}$$

o

$$\xi_i^{(k+1)} = \frac{1}{a_{ii}}\left(\beta_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}\xi_j^{(k)}\right) \qquad i = 1, \ldots, n \tag{2.31}$$

Todos los componentes de la siguiente iteración serán agrupados en el vector x_{k+1}. Podemos reescribir la notación anterior para la iteración de Jacobi de forma matricial como

$$x_{k+1} = D^{-1}(E + F)\, x_k + D^{-1}b \tag{2.32}$$

y en el algoritmo 2.1 se ilustra el método.

Algoritmo 2.1 Método de Jacobi

elegir $x^{(0)}$ #aproximación inicial a la solución x
para $k = 1, 2, \ldots$ **hacer**
  para $i = 1, 2, \ldots, n$ **hacer**
    $\overline{x}_i = 0$
    para $j = 1, 2, \ldots, i-1, i+1, \ldots, n$ **hacer**
      $\overline{x}_i = \overline{x}_i + a_{i,j}x_j^{(k-1)}$
    fin para
    $\overline{x}_i = (b_i - \overline{x}_i)/a_{i,i}$
  fin para
  $x^{(k)} = \overline{x}$ #revisar la convergencia, continuar de ser necesario
fin para

De manera similar, la iteración de Gauss-Seidel corrige el i-ésimo componente de la actual solución aproximada, en el orden $i = 1, 2, \ldots, n$, de nuevo anula el i-ésimo componente del vector residual [13]. Sin embargo, esta vez la solución aproximada se actualiza inmediatamente después de que el nuevo componente es determinado. Los nuevos componentes calculados $\xi_i^{(k)}$, $i = 1, 2, \ldots, n$ pueden ser modificados dentro de un vector de trabajo que es redefinido en cada paso de relajación. Por lo tanto, puesto

que el orden es $i = 1, 2, \ldots$ el resultado en el paso i-ésimo es

$$\beta_i - \sum_{j=1}^{i-1} a_{ij}\xi_j^{(k+1)} - a_{ii}\xi_i^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}\xi_j^{(k)} = 0 \tag{2.33}$$

lo que conduce a la iteración

$$\xi_i^{(k+1)} = \frac{1}{a_{ii}} \left(-\sum_{j=1}^{i-1} a_{ij}\xi_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}\xi_j^{(k)} + \beta_i \right), \qquad i = 1, \ldots, n \tag{2.34}$$

La ecuación (2.33) se puede escribir como

$$b + Ex_{k+1} - Dx_{k+1} + Fx_k = 0 \tag{2.35}$$

que conduce a la forma vectorial de la iteración de Gauss-Seidel

$$x_{k+1} = (D - E)^{-1} Fx_k + (D - E)^{-1} b \tag{2.36}$$

El cálculo de la nueva aproximación en la ecuación (2.32) requiere multiplicar por la matriz inversa de la matriz diagonal D. En la ecuación (2.36), un sistema triangular debe ser resuelto con $D - E$, la parte triangular inferior de A. Por lo tanto, el nuevo paso de aproximación de Gauss-Seidel puede ser determinado ya sea por la solución del sistema triangular con la matriz $D - E$ o desde la relación (2.34).

Una iteración de Gauss-Seidel hacía atrás puede ser definida como

$$(D - F)\, x_{k+1} = Ex_k + b \tag{2.37}$$

que es equivalente a hacer las correcciones de coordenadas en el orden $n, n-1, \ldots, 1$. Una iteración de Gauss-Seidel simétrica consiste de un barrido hacía adelante seguido de un barrido hacía atrás.

Las iteraciones de Jacobi y Gauss-Seidel ambas son de la forma

$$Mx_{k+1} = Nx_k + b = (M - A)\, x_k + b \tag{2.38}$$

en donde

$$A = M - N \tag{2.39}$$

es una descomposición de A, con $M = D$ para Jacobi, $M = D - E$ para Gauss-Seidel hacía adelante y $M = D - F$ para Gauss-Seidel hacía atrás. En el algoritmo 2.2 se ilustra el método de Gauss-Seidel.

Un método iterativo de la forma de la ecuación (2.38) puede ser definido para cualquier descomposición de la forma de la ecuación (2.39) donde M es no singular. La sobre relajación está basada en la descomposición

$$\omega A = (D - \omega E) - (\omega F + (1 - \omega)\, D) \tag{2.40}$$

Algoritmo 2.2 Método de Gauss-Seidel

elegir $x^{(0)}$ #aproximación inicial a la solución x
para $k = 1, 2, \ldots$ **hacer**
  para $i = 1, 2, \ldots, n$ **hacer**
    $\sigma = 0$
    para $j = 1, 2, \ldots, i-1$ **hacer**
      $\sigma = \sigma + a_{i,j} x_j^{(k)}$
    fin para
    para $j = i+1, \ldots, n$ **hacer**
      $\sigma = \sigma + a_{i,j} x_j^{(k-1)}$
    fin para
    $x_i^{(k)} = (b_i - \sigma) / a_{i,i}$
  fin para
  #revisar la convergencia, continuar de ser necesario
fin para

y el método de sobre relajación está basado en la recursión

$$(D - \omega E)\, x_{k+1} = [\omega F + (1 - \omega)\, D]\, x_k + \omega b \tag{2.41}$$

La iteración anterior corresponde a la secuencia de relajación

$$\xi_i^{(k+1)} = \omega \xi_i^{GS} + (1 - \omega)\, \xi_i^{(k)}, \qquad i = 1, 2, \ldots, n \tag{2.42}$$

donde ξ_i^{GS} está definida por la expresión en el lado derecho de la ecuación (2.34). Un barrido hacía atrás de SOR puede ser definido análogamente como el barrido hacía atrás de Gauss-Seidel en (2.37). En el algoritmo 2.3 mostramos el método de SOR.

Algoritmo 2.3 Método de SOR

elegir $x^{(0)}$ #aproximación inicial a la solución x
para $k = 1, 2, \ldots$ **hacer**
  para $i = 1, 2, \ldots, n$ **hacer**
    $\sigma = 0$
    para $j = 1, 2, \ldots, i-1$ **hacer**
      $\sigma = \sigma + a_{i,j} x_j^{(k)}$
    fin para
    para $j = i+1, \ldots, n$ **hacer**
      $\sigma = \sigma + a_{i,j} x_j^{(k-1)}$
    fin para
    $\sigma = (b_i - \sigma) / a_{i,i}$
    $x_i^{(k)} = x_i^{(k-1)} + \omega \left(\sigma - x_i^{(k-1)} \right)$
  fin para
  #revisar la convergencia, continuar de ser necesario
fin para

2.2.4. Gradiente conjugado, MINRES y SYMMLQ

El método de *Gradiente Conjugado* es un método efectivo para sistemas de matrices simétricamente definidas positivas [25]. El método pasa por la generación de secuencias del vector de iteración (es decir, aproximaciones sucesivas a la solución), residuos correspondientes a las iteraciones y direcciones de búsqueda, usadas para la actualización de las iteraciones y los residuos.

Aunque la longitud de estas secuencias puede ser muy grande, sólo un pequeño número de vectores se necesita que se mantengan en memoria.

En cada iteración del método, dos productos internos se realizan en orden para calcular la actualización de los escalares que están definidos para hacer que las secuencias satisfagan ciertas condiciones de ortogonalidad.

En un sistema simétricamente definido positivo, estas condiciones implican que la distancia a la verdadera solución se minimiza en alguna norma.

Las valor de $x^{(i)}$ son actualizados en cada iteración por un múltiplo (α_i) del vector de dirección de búsqueda $p^{(i)}$

$$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)} \tag{2.43}$$

en consecuencia el residual $r^{(i)} = r^{(i-1)} - Ax^{(i)}$ se actualiza como

$$r^{(i)} = r^{(i-1)} + \alpha_i p^{(i)} \tag{2.44}$$

donde $q^{(i)} = Ap^{(i)}$. La elección de $\alpha = \alpha_i = {r^{(i-1)}}^T r^{(i-1)} / {p^{(i)}}^T Ap^{(i)}$ minimiza ${r^{(i)}}^T A^{-1} r^{(i)}$ bajo todas las posibles elecciones para la ecuación (2.44).

Las direcciones de búsqueda son actualizada usando los residuales

$$p^{(i)} = r^{(i)} + \beta_{i-1} p^{(i-1)} \tag{2.45}$$

donde la elección $\beta_i = {r^{(i)}}^T r^{(i)} / {r^{(i-1)}}^T r^{(i-1)}$ asegura que $p^{(i)}$ y $Ap^{(i-1)}$ o equivalentemente $r^{(i)}$ y $r^{(i-1)}$ son ortogonales.

En el algoritmo 2.4 se muestra el método de Gradiente conjugado para un precondicionador M; para $M = I$ obtenemos la versión no precondicionada del Gradiente Conjugado.

El método de Gradiente conjugado puede ser visto como una variante especial del método de *Lanczos* para sistemas de matrices definidas positivas [25]. Los métodos de MINRES y SYMMLQ son variantes que pueden ser aplicadas a sistemas simétricamente indefinidos.

Cuando A no es definida positiva, pero simétrica, todavía podemos construir una base

Algoritmo 2.4 Gradiente Conjugado

elegir x_0 #podría ser cualquier vector pero uno que se acerque a la solución produce menos iteraciones
calcular $r^{(0)} = b - Ax^{(0)}$ **para** un valor inicial de $x^{(0)}$
para $i = 1, 2, \ldots$ **hacer**
  resolver $Mz^{(i-1)} = r^{(i-1)}$
  $\rho_{i-1} = {r^{(i-1)}}^T z^{(i-1)}$
  si $i = 1$ **entonces**
    $p^{(1)} = z^{(0)}$
  si no
    $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
    $p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$
  fin si
  $q^{(i)} = Ap^{(i)}$
  $\alpha_i = \rho_{i-1}/{p^{(i)}}^T q^{(i)}$
  $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)} = r^{(i-1)} - \alpha_i p^{(i)}$
  #revisar la convergencia, continuar de ser necesario
fin para

ortogonal en el subespacio de Krylov con tres relaciones de recurrencia. Eliminando las direcciones de búsqueda en las ecuaciones (2.44) y (2.45) obtenemos la siguiente recurrencia

$$Ar^{(i)} = r^{(i+1)} t_{i+1,i} + r^{(i)} t_{i,i} + r^{(i-1)} t_{i-1.i} \tag{2.46}$$

que puede ser escrita de forma matricial como

$$AR_i = R_{i+1} \overline{T}_i \tag{2.47}$$

donde $\overline{T}_i$ es una matriz tridiagonal de $(i+1) \times i$

$$\overline{T}_i = \begin{array}{c} \overset{\leftarrow i \rightarrow}{} \\ \begin{pmatrix} \ddots & \ddots & & \\ \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \ddots \\ & & & \ddots \end{pmatrix} \end{array} \begin{array}{c} \uparrow \\ i+1 \\ \downarrow \end{array}$$

2.2.5. GMRES

El método está diseñado para resolver sistemas lineales no simétricos. La forma más popular de GMRES se basa en la modificación Procedimiento de Gram-Schmidt que utiliza para controlar los requisitos de almacenamiento [25].

En el método de Gradiente Conjugado, los residuos forman una base ortogonal para el espacio $\left\{r^{(0)}, Ar^{(0)}, A^2r^{(0)}, \ldots\right\}$. En GMRES, esta base está formada explícitamente por:

$w^{(i)} = Av^{(i)}$
for $k = 1, \ldots, i$
$w^{(i)} = w^{(i)} - \left(w^{(i)}, v^{(k)}\right) v^{(k)}$
end
$v^{(i+1)} = w^{(i)} / \left\|w^{(i)}\right\|$

Aplicado a la secuencia de Krylov, esta ortogonalización es llamada método de Arnoldi. Los coeficientes de los productos internos $\left(w^{(i)}, v^{(k)}\right)$ y $\left\|w^{(i)}\right\|$ son almacenados en la parte superior de la matriz de Hassenberg. La iteración de GMRES es

$$x^{(i)} = x^{(0)} + y_1 v^1 + \cdots + y_i v^i \tag{2.48}$$

donde los coeficientes y_k han sido escogidos para minimizar la norma del residuo $\left\|b - Ax^{(i)}\right\|$. La norma del residuo puede ser calculada sin que se haya formado la iteración. Por lo tanto, la acción costosa de formar la iteración se puede posponer hasta que la norma del residuo se considere lo suficientemente pequeña.

A continuación se muestra el método en el algoritmo 2.5

2.3. Modelo Computacional

2.4. Software existente

Los métodos presentados en la sección anterior se han implementado en varios softwares especializados, libres y comerciales. A continuación se revisan algunos paquetes de software que se emplean para la modelación matemática de agua subterránea.

2.4.1. Modflow

Modflow es un programa de modelación en 3D para agua subterránea, trabaja bajo el método de diferencias finitas. Publicado por primera vez en 1984 (U.S. Geological Survey, Office of Groundwater, Enero 2011[2]).

Originalmente fue concebido para simulación de flujo de aguas subterráneas, pero paulatinamente se han ido incorporando *módulos* para aprovechar y potenciar sus usos.

La familia de componentes de Modflow incluye capacidades para simular aguas subterráneas, transporte de soluto, flujo de densidad variable y zona no saturada, sistema de acuífero de compactación y hundimiento del terreno.

[2]http://water.usgs.gov/nrp/gwsoftware/modflow-status-2011Jan.pdf

Algoritmo 2.5 GMRES

elegir $x^{(0)}$ #aproximación inicial a la solución x
para $j = 1, 2, \ldots$ **hacer**
  Resolver r **desde** $Mr = b - Ax^{(0)}$
  $v^{(1)} = r / \|r\|_2$
  $s := \|r\|_2 e_1$
  para $i = 1, 2, \ldots, m$ **hacer**
    Resolver w **desde** $Mw = Av^{(1)}$
    para $k = i, \ldots, i$ **hacer**
      $h_{k,i} = \left(w, v^{(k)}\right)$
      $w = w - h_{k,i} v^{(k)}$
    fin para
    $h_{i+1,i} = \|w\|_2$
    $v^{(i+1)} = w / h_{i+1,i}$
    aplicar $J_1, \ldots, J_{i-1}$ en $(h_{1,i}, \ldots, h_{i+1,i})$
    construir J_i, actuando sobre el i-ésimo y el i-ésimo más un componente de
    $h_{.,i}$, tal que el i-ésimo más un componente de $J_i h_{.,i}$ es 0
    $s := J_i s$
    si $s\,(i+1)$ es lo suficientemente pequeño **entonces**
      ACTUALIZAR$(\tilde{x}, i)$ y salir
    fin si
  fin para
  ACTUALIZAR$(\tilde{x}, i)$
fin para

En este esquema ACTUALIZAR$(\tilde{x}, i)$
reemplaza los siguientes cálculos:

Calcular y como una solución de $Hy = \tilde{s}$, en la cual
la parte superior de la matriz triangular de H, tiene como
sus elementos,
$\tilde{s}$ que representa el primer componente i-ésimo de s
$\tilde{x} = x^{(0)} + y_1 v^{(1)} + y_2 v^{(2)} + \ldots + y_i v^{(i)}$
$s^{(i+1)} = \|b - A\tilde{x}\|_2$
si $\tilde{x}$ es una aproximación bastante exacta entonces terminar
sino $x^{(0)} = \tilde{x}$

Es importante señalar que Modflow posee varias versiones de un kernel, ya que al ser software libre, es común que se generen varias ramas de un proyecto.

Otra característica importante es que el software puede hacer simulaciones tanto en estado transitorio, como en estado estacionario, con la facilidad de componer un acuífero confinados o semiconfinados con varias capas de distintos materiales.

Es posible incluir en las simulaciones fuentes o sumideros como: pozos de extracción o de inyección, áreas de recarga, flujos y drenes, así como otros parámetros, por ejemplo: coeficiente de almacenamiento, especificación de la carga hidráulica, condiciones de frontera, geometría del acuífero a analizar.

Cuando se comienza a realizar un modelo con Modflow, la información proporcionada para el modelo, debe ser cuidadosamente escogida, ya que datos como: el tamaño de la malla, el número de capas y la dimensión de la geometría afectan considerablemente a los tiempos y por tanto a la precisión y aplicabilidad del modelo.

Las ecuaciones diferenciales parciales son resueltas con el método de diferencias finitas, cada celda es considerada como un volumen unitario.

2.4.2. Feflow

Feflow es un software comercial para la simulación de aguas subterráneas, una herramienta para el análisis de problemas donde interviene la ecuación de flujo y transporte de contaminantes o calor en medio poroso.

Este software emplea la discretización de elementos finitos con refinamiento de malla local, que proporciona los medios para construir grandes modelos a escala regional sin perder precisión local donde sea necesaria.

Feflow contiene la funcionalidad de procesamiento previo y posterior y un motor de simulación. Resuelve el balance de ecuaciones de advección-dispersión para contaminantes de masa (salinidad) y calor para 2D y 3D.

Permite el análisis de ambos acuíferos saturados (agua subterránea) y acuíferos no saturados. Para el caso de no saturado, resuelva la ecuación de Richards para modelos paramétricos opcionales como Van Genuchten-Mualem.

Posee la característica para emplearse en una o múltiples fases. Los efectos de adsorción pueden ser modelados vía isotermas de Henry, Freundlich.

Es posible emplear el software en condiciones transitorias o de flujo estacionario. Además de modelar una sola o múltiples especies.

En la solución de los sistemas de ecuaciones formados, el gradiente conjugado preacondicionado (PCG) se proporciona.

Se utiliza preacondicionador estándar tales como la factorización incompleta (IF) y, alternativamente, una técnica de factorización (MIF) basada en el algoritmo de Gustafsson.

Diferentes alternativas están disponibles para la solución de CG-como de las ecuaciones de transporte no simétricas: una ORTHOMIN renovadas (ortogonalización de minimización) método, un GMRES reiniciado (generalizada residual mínima) los métodos de la técnica y el tipo de Lanczos tales como CGS (gradiente conjugado cuadrado), BiCGSTAB (bi-estable gradiente conjugado) y BiCGSTABP (postconditioned bi-conjugado estable gradiente).

Para el acondicionamiento previo. El monitoreo de memoria del programa es completamente dinámico.

2.4.3. UTCHEM

Es un software libre desarrollado por la Universidad de Texas en Austin, por sus siglas en inglés es *Chemical Compositional Simulator* (Simulador Químico Composicional), y pose las siguientes características:

1. Se emplea en problemas desde una a tres dimensiones.
2. Es multifásico.
3. Es multicomponente.
4. Es composicional.
5. Admite variación en la temperatura.
6. Utiliza el método de diferencias finitas para la simulación numérica.

En aguas subterráneas el programa puede emplearse en:

1. La infiltración de NAPL (Non aquose phase liquids) en zonas saturadas y no saturadas.
2. Particionamiento y pruebas de trazado entre pozos (Pitts) en zonas saturados como insaturadas.
3. Remediación usando tensioactivos (SEAR) y/o polímeros, espuma de tensioactivo, o cosolventes.
4. Biorremediación.

También tiene aplicaciones en el campo del petróleo, tales como:

1. Pruebas de trazadores para caracterizar simple y doble porosidad yacimientos de petróleo.

2. Tensioactivo EOR incluyendo el uso de polímeros y espumas.
3. Inundaciones de polímeros para EOR.
4. Inundaciones de un alto PH químico para EOR.
5. EOR microbiana.
6. Perfil de control de pozos petroleros con geles de polímero Modelado de daño a la formación de pozos petroleros.

El simulador puede modelar presiones capilares, permeabilidades relativas de tres fases (agua / gas / fases orgánicas o agua / orgánicos / fases de microemulsión), la dispersión, difusión, adsorción, las reacciones químicas, de no equilibrio de transferencia de materia entre fases y relacionada con otras fenómenos.

El método de diferencias finitas que utiliza, tiene aproximaciones de segundo y tercer orden para el tiempo y derivadas espaciales y un limitador de flujo, el método de variación total decreciente (TVD).

Como se puede observar, ninguno de los paquetes antes revisados, incluye cómputo paralelo. En la sección que sigue explicamos el modelo de cómputo paralelo que se usa en este trabajo.

2.5. Modelo para cómputo en paralelo

El modelo de programación que utilizamos (generalmente) es el modelo de un programa único de datos múltiples (SPMD) de memoria distribuida (DM) [26].

El modelo SPMD DM ve la máquina paralela como p procesadores independientes cada uno con su propia memoria local.

Escribir un código de la aplicación paralela en el modelo SPDM DM, generalmente, consiste en escribir un programa de computadora que se ejecuta con secuencias de instrucciones (independientes) en cada procesador.

La comunicación entre los procesadores se logra mediante el uso del paso de mensajes.

2.5.1. Software para solución de sistemas lineales en paralelo (PETSc)

Algunos lenguajes como *Fortran 90, Pascal, Ada, C y C++*, tienen un número limitado de tipos de datos, pero poseen mecanismos adicionales que permiten al programador construir nuevos tipos de estructuras de datos de la combinación de los datos que ya vienen predefinidos.

En C se conoce como estructuras, en C++ como clases. Estas estructuras de datos

definidas por el programador pueden tratarse al igual que los tipos de datos estándar.

Esto permite a la persona que diseñe el código obtener un mayor nivel de abstracción, como por ejemplo: *matrices dispersas, mallas, discretizaciones, vectores,* sin preocuparse por los detalles de la implementación.

El software que se presenta aquí es un kit de herramientas, portables, extensibles para cómputo científico *(PETSc por sus siglas en inglés Portable, Extensible Toolkit for Scientific Computing)* escrito por William Gropp y Barry Smith [27].

El paquete *PETSc* es un gran conjunto de estructuras de datos y rutinas que están destinados como bloques de construcción para la realización aplicaciones portables, en paralelo y que sean eficientes en la solución de problemas en computación científica, especialmente la solución de ecuaciones diferenciales parciales [2].

Todas sus funciones se pueden utilizar directamente desde *Fortran 77* o lenguaje C, o *Python*.

El paquete PETSc se basa en dos principios fundamentales: *encapsulamiento de datos* y en *software por capas [2].*

En la ocultación de datos (encapsulamiento), la representación particular de datos está todo el código, excepto las rutinas que directamente deben manipular los datos, es decir, las *primitivas* para ese tipo de datos en específico.

Por ejemplo, una rutina de gradiente conjugado en paralelo no necesita saber el formato en que los vectores y los operadores lineales son almacenados; sólo las rutinas de manipulación de vector y la matriz necesita conocer sus detalles.

En el software por capas, las estructuras de datos y rutinas de mayor nivel son construidas mediante el uso de rutinas y estructuras de datos de un menor nivel [2], que a su vez pudieron haber sido construidas a a partir de estructuras de datos más simples y esto se presenta sucesivamente como se ve en la figura 2.2.

Por ejemplo, el software para resolver una ecuación no lineal podría construirse en la parte superior de un paquete para la solución de sistemas lineales el cual esta construido en el software para la manipulación de matrices ralas o no densas.

Una de las ventajas del encapsulamiento de datos y el software distribuido por capas, es que esto permite a los programadores desarrollar una aplicación sin que éste requiera entender la aplicación entera y los detalles fundamentales de la máquina para la cual el código ha sido desarrollado y así centrarse en los diferentes aspectos [19].

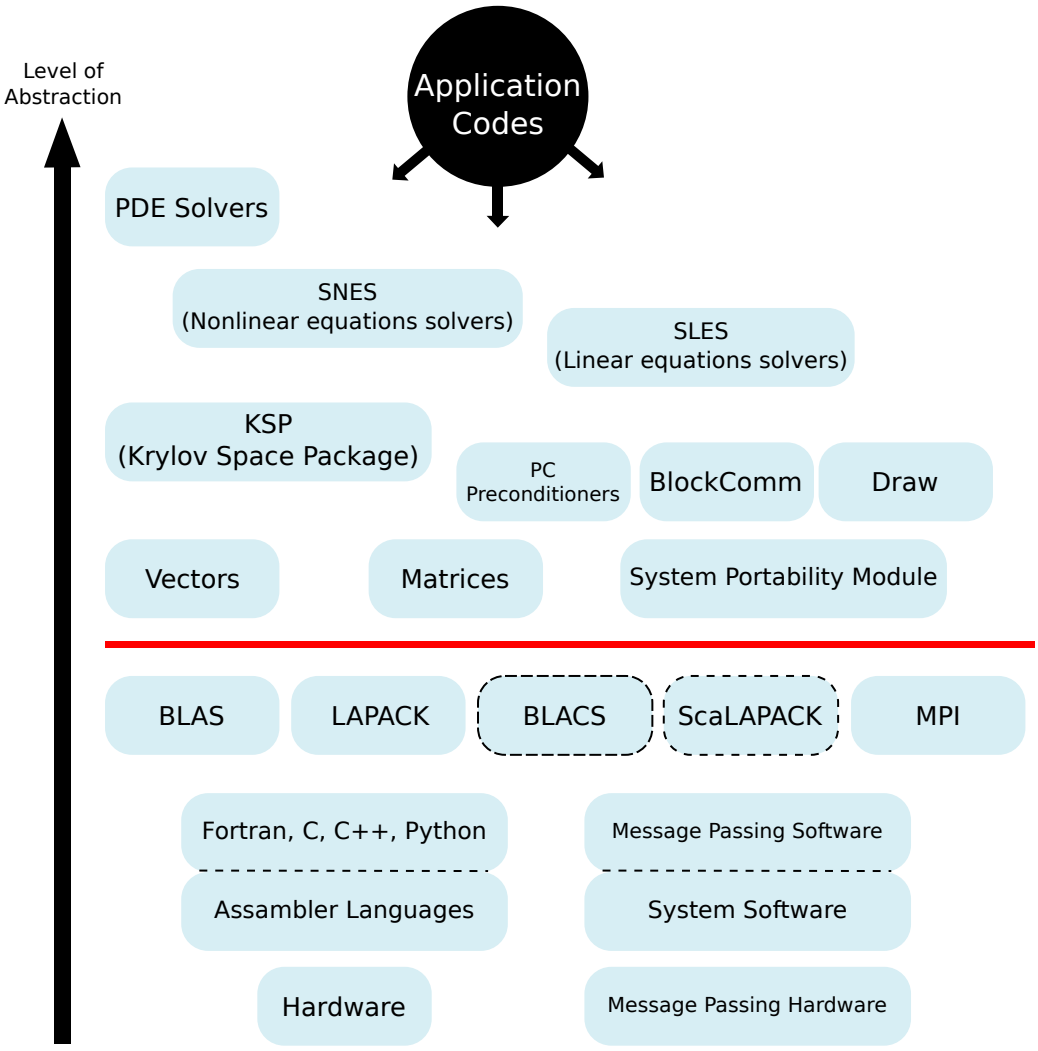


Figura 2.2: Organización de las bibliotecas de PETSc

2.5.1.1. Tipos de datos abstractos

Algunos tipos de datos los cuales tienen un uso extensivo en rutinas de alto nivel en PETSc [2].

2.5.1.2. Variables de contexto

Las rutinas que operan en complicadas estructuras de datos a menudo tienen muchos parámetros opcionales. Ya sea por que tienen secuencias muy largas de argumentos para las rutinas o el uso de bloques comunes. El problema con el enfoque anterior es que las rutinas dependen de las variables globales.

Para hacer frente a este problema, se ha optado por utilizar las variables de contexto. En su forma más simple, las variables de contexto son simplemente una manera de evitar largas secuencias. Se pueden utilizar para evitar todas las variables globales y por lo tanto permiten el uso flexible.

2.5.1.3. Vectores

Un vector es simplemente un arreglo de una dimensión de números almacenados en localidades consecutivas de memoria. En los equipos en paralelo hay muchos patrones de almacenamiento potenciales.

El objeto vector (llamado así desde el enfoque de programación orientada a objetos) contiene dos partes: una lista de todas las primitivas del vector y un contexto privado que permite agregar información necesaria para la implementación.

En forma secuencial, la implementación de un vector podría ser un entero que contiene el tamaño de un vector. La implementación podría ser una estructura en C, la cual contiene la longitud de la parte que reside en un procesador en particular y una lista de procesos que el vector comparte.

Los objetos vector no sólo contienen rutinas para la realización de las operaciones de tipo BLAS (Basic Linear Algebra Subprograms), sino que también se les agrega rutinas para generar más vectores. PETSc tienen ambas implementaciones: en forma secuencial o en paralelo de vectores.

2.5.1.4. Métodos del Subespacio de Krylov

PETSc tiene una *suite* de métodos del subespacio de Krylov para soluciones iterativas de sistemas lineales, llamada, KSP. Estas son programadas con secuencias comunes y con el uso de vectores (vistos anteriormente).

2.5.1.5. Matrices dispersas

Las matrices dispersas usan una forma de variables de contexto del tipo Mat. Para mantener el almacenamiento de datos de las matrices dispersas, accedemos a las

matrices dispersas sólo a través de un pequeño conjunto de rutinas bien definidas, las primitivas de la matriz. Estas rutinas nos permiten añadir valores a las matrices, realizar multiplicaciones por matrices, obtener factores de las matrices, eliminar filas y columnas de las matrices, etc.

2.5.1.6. Solucionadores de sistemas lineales y no lineales

PETSc es un conjunto jerárquico de bibliotecas que permite elegir una interfaz abstracta o simple para el software que se desea usar. El solucionador simplificado de ecuaciones lineales (SLES Simplified Linear Equation Solver) provee una para la solución de sistemas lineales.

PETSc también tiene una estructura de datos para la solución de sistemas de ecuaciones no lineales llamado SNES (Simplified Nonlinear Equation Solver).

El siguiente código de ejemplo, escrito en lenguaje C, se resuelve la ecuación de Laplace en una dimensión utilizando el método de diferencias finitas con las herramientas de PETSc.

```
static char help[] = "Solves a tridiagonal linear system with
    KSP.\n\n";
/*T
Concepts: KSP solving a system of linear equations
Processors: 1
T*/

#include "petscksp.h"

#undef __FUNCT__
#define __FUNCT__ "main"
int main(int argc,char **args)
{
        Vec x, b, u; /* approx solution, RHS, exact solution */
        Mat A; /* linear system matrix */
        KSP ksp; /* linear solver context */
        PC pc; /* preconditioner context */
        PetscReal norm; /* norm of solution error */
        PetscErrorCode ierr;
        PetscInt i,n = 10,col[3],its;
        PetscMPIInt size;
        PetscScalar neg_one = -1.0,one = 1.0,value[3];

        PetscInitialize(&argc,&args,(char *)0,help);
        ierr = MPI_Comm_size(PETSC_COMM_WORLD,&size);CHKERRQ(
            ierr);
        if (size != 1) SETERRQ(1,"This is a uniprocessor
            example        only!");
        ierr=PetscOptionsGetInt(PETSC_NULL,"-n",&n,PETSC_NULL);
```

```
            CHKERRQ(i        err);
27
28  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        - - - -
29  Compute the matrix and right-hand-side vector that define
30  the linear system, Ax = b.
31  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
         - - */
32
33  /*
34  Create vectors.
35  */
36          ierr = VecCreate(PETSC_COMM_WORLD,&x);CHKERRQ(ierr);
37          ierr = PetscObjectSetName((PetscObject) x,      "
                Solution");CHKERRQ(ierr);
38          ierr = VecSetSizes(x,PETSC_DECIDE,n);CHKERRQ(ierr);
39          ierr = VecSetFromOptions(x);CHKERRQ(ierr);
40          ierr = VecDuplicate(x,&b);CHKERRQ(ierr);
41          ierr = VecDuplicate(x,&u);CHKERRQ(ierr);
42
43  /*
44  Create matrix.
45  */
46          ierr = MatCreate(PETSC_COMM_WORLD,&A);CHKERRQ(ierr);
47          ierr=MatSetSizes(A,PETSC_DECIDE,PETSC_DECIDE,n,n);
                CHKERRQ(ierr);
48          ierr = MatSetFromOptions(A);CHKERRQ(ierr);
49  /*
50  Assemble matrix
51  */
52          value[0] = -1.0; value[1] = 2.0; value[2] = -1.0;
53          for (i=1; i<n-1; i++) {
54                  col[0] = i-1; col[1] = i; col[2] = i+1;
55                  ierr=MatSetValues(A,1,&i,3,col,value,
                        INSERT_VALUES);CHKERRQ(            ier);
56          }
57          i = n - 1; col[0] = n - 2; col[1] = n - 1;
58          ierr=MatSetValues(A,1,&i,2,col,value,INSERT_VALUES);
                CHKERRQ(ierr        );
59          i = 0; col[0] = 0; col[1] = 1; value[0] = 2.0; value[1]
                 = -1.0;
60          ierr=MatSetValues(A,1,&i,2,col,value,INSERT_VALUES);
                CHKERRQ(ierr        );
61          ierr = MatAssemblyBegin(A,MAT_FINAL_ASSEMBLY);CHKERRQ(
                ierr);
62          ierr = MatAssemblyEnd(A,MAT_FINAL_ASSEMBLY);CHKERRQ(
                ierr);
63
64  /*
65  Set exact solution; then compute right-hand-side vector.
```

```
66  */
67          ierr = VecSet(u,one);CHKERRQ(ierr);
68          ierr = MatMult(A,u,b);CHKERRQ(ierr);
69
70  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        - - - -
71  Create the linear solver and set various options
72  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        - - */
73
74          ierr = KSPCreate(PETSC_COMM_WORLD,&ksp);CHKERRQ(ierr);
75
76          ierr=KSPSetOperators(ksp,A,A,DIFFERENT_NONZERO_PATTERN)
                ;CHKERRQ(        ierr);
77
78          ierr = KSPGetPC(ksp,&pc);CHKERRQ(ierr);
79          ierr = PCSetType(pc,PCJACOBI);CHKERRQ(ierr);
80          ierr=KSPSetTolerances(ksp,1.e-7,PETSC_DEFAULT,
                PETSC_DEFAULT,PETS        C_DEFAULT);CHKERRQ(ierr);
81
82          ierr = KSPSetFromOptions(ksp);CHKERRQ(ierr);
83
84  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        - - - -
85  Solve the linear system
86  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        - - */
87          ierr = KSPSolve(ksp,b,x);CHKERRQ(ierr);
88          ierr = KSPView(ksp,PETSC_VIEWER_STDOUT_WORLD);CHKERRQ(
                ierr);
89
90          ierr = PetscFinalize();CHKERRQ(ierr);
91          return 0;
92  }
```

2.6. FEniCS

El Proyecto FEniCS es un proyecto de colaboración para el desarrollo de conceptos innovadores y herramientas de cálculo científico automatizado, prestando especial atención a la solución automatizada de ecuaciones diferenciales por el método de elemento finito[3].

Inicialmente el FEniCS consiste de dos bibliotecas: DOLFIN y FIAT (principalmente).

El software desarrollado por el Proyecto FENICS es gratuito (licenciado bajo la GNU

[3] http://fenicsproject.org/

(L) GPL).

2.7. Computo en Paralelo con FEniCS

FeniCS a través de su biblioteca DOLFIN, soporta cómputo paralelo [22].

Está diseñado de tal manera que los usuarios pueden realizar simulaciones en paralelo utilizando el mismo código que se usa para los cálculos de la forma secuencial.

Dos paradigmas para la simulación paralela son soportados. El primer paradigma es *multithreading* para máquinas de memoria compartida. El segundo paradigma es la paralelización totalmente distribuida para máquinas de memoria distribuida.

Para el cómputo en paralelo de memoria compartida es soportado por OpenMP, que es activado por la selección del número de *threads* a usar vía los parámetros del sistema.

En el cómputo en paralelo por memoria distribuida es soportado usando Message Passing Interface (MPI). Para llevar a cabo las simulaciones paralelas, DOLFIN debe ser compilado con MPI y un motor paralelo de álgebra lineal (puede ser PETSc o TRILINOS) habilitado [22].

Para ejecutar un simulación en paralelo, un programa debe ser lanzado usando mpirun (el nombre del programa que ejecutará Programas MPI pueden diferir en algunos computadoras).

Un programa en C++ utilizando 16 procesadores puede ser ejecutado usando:

```
mpirun -n 16 ./miprograma
```

y en Python:

```
mpirun -n 16 python miprograma.py
```

DOLFIN soporta completamente mallas en paralelo y distribuidas, es decir, significa que cada procesador tiene una copia de una única porción de la malla de la cual es responsable de calcular.

2.8. TUNAM

En este trabajo se utilizará el software TUNAM (Templates Units for Numerical Applications and Modeling) [11][12][10], el cual contiene herramientas especiales para el MVF y será adaptado para resolver los problemas planteados antes.

TUNAM es un software desarrollado originalmente para resolver problemas de convección natural en dominios rectangulares mediante el método de volumen finito. En

la construcción de TUNAM se aplicaron los paradigmas de programación orientada a objetos (POO) y genérica, de tal manera que es posible utilizar varios de sus conceptos y módulos para resolver problemas de otras áreas de estudio. Las componentes de este software están programadas en C++ y hace uso intensivo de *templates* los cuales proveen una herramienta efectiva para desarrollar programas genéricos. Además, se basa en la biblioteca Blitz++ [5] para manejo de arreglos multidimensionales. Esta última implementa las técnicas de *expression templates* [29], *meta programming* y *traits* [23], para obtener códigos claros y eficientes. De igual manera, se aplica el patrón de diseño CRTP (por su siglas en inglés Curiously Recurring Template Pattern) [3][9].

En TUNAM se aplica CRTP para realizar polimorfismo estático, y obtener construcciones genéricas que se optimizan en tiempo de compilación, véase [28]. Se utilizan las siguientes definiciones:

Definición 1 *Una **generalización** (Generalization) propone la existencia de un conjunto de elementos que comparten características comunes, entre ellas sus atributos y operaciones.*

Definición 2 *Una **especialización** (Specialization) es un caso particular de una entidad general que adiciona y/o redefine características (atributos y operaciones) especiales.*

Definición 3 *Un **adaptador** (Adaptor) es una implementación particular de un mismo concepto o algoritmo.*

Estas definiciones pueden ser explotadas mediante el uso de templates, permitiendo implementaciones alternativas de un mismo concepto y generando código optimizado. Tomando en cuenta los modelos conceptuales, matemáticos y numéricos mostrados en las secciones anteriores, se puede identificar lo siguiente:

- Generalizaciones:

  GeneralMesh : Representa cualquier tipo de malla, véase por ejemplo figura 3.1.

  GeneralEquation : Representa la ecuación general discreta (3.66), junto con sus coeficientes $(a_P, a_E, a_W, a_N, a_S, a_F, a_B, s)$.

  GeneralMatrix : Representa una matriz en general.

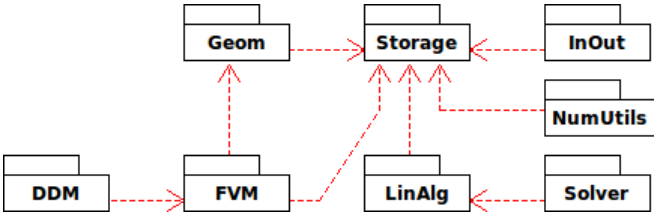
- Especializaciones :

  StructuredMesh : Representa mallas estructuradas como la mostrada en la figura 3.1.

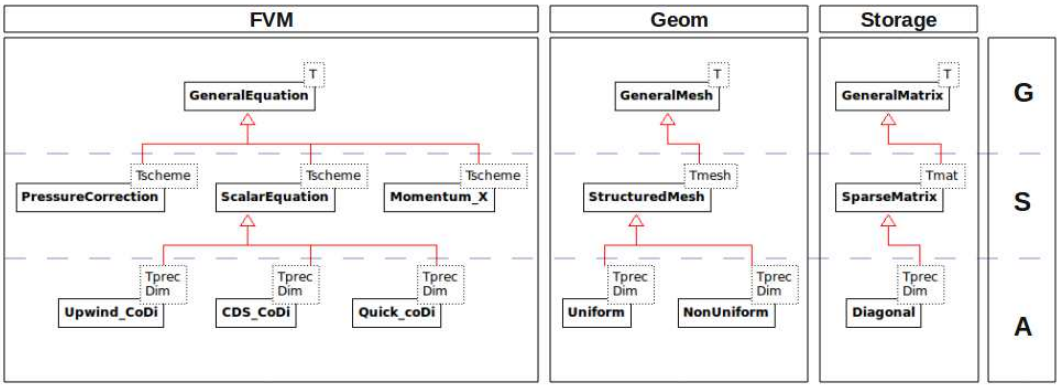
  TwoPhaseEquation : Ecuación diferencial particulares para flujo en dos fases.

  SparseMatrix : Representa matrices ralas.

- Adaptadores :



(a)



(b)

Figura 2.3: Arquitectura General de TUNAM
(a) Arquitectura General de TUNAM (b) Los paquetes **FVM** y **Geom** se muestran de manera esquemática: las letras G, S y A, a la derecha de la figura b, significan *Generalization, Specialization y Adaptor*, respectivamente.

Uniform, NonUniform : Definen mallas uniformes y no uniformes para dominios rectangulares.

CDS, Upwind, Quick, etc. : Definen los esquemas numéricos que se usarán para calcular los coeficientes de volumen finito.

Diagonal: Define matrices ralas en formato diagonal.

La anterior es una lista reducida de algunas herramientas implementadas en TUNAM. La manera en que se definen y crean los objetos que interactuarán para resolver un problema en TUNAM es la siguiente:

```
Specialization<Adaptor<prec t , dim> > object(arg1 , ... ,
argN);
```

Cuando un compilador de C++ analiza esta declaración, automáticamente genera código especializado y optimizado, de acuerdo con la implementación de la especialización y el adaptador correspondientes. En el código anterior `object` será un objeto perteneciente a la clase `Specialization<Adaptor<prec_t, dim> >` y `arg1`,

`..., argN` son los argumentos usados para la creación de `object`. Aquí, el parámetro `prec_t` define la precisión de los resultados `(float, double o long double)`, y el parámetro `dim` define la dimensión del problema. La figura 2.3 muestra un esquema general de TUNAM y algunas de sus generalizaciones, especializaciones y adaptadores. Para más detalles de TUNAM véase [11][12].

2.8.1. Mediciones en cómputo en paralelo

En cualquier área, es necesario disponer de métricas que ayuden a evaluar objetivamente, el comportamiento óptimo en el desarrollo de alguna actividad.

Así, en cómputo paralelo existen mediciones, las principales son dos, el *speedup* y la *eficiencia*.

El **speedup** [4] es la relación en tiempo entre una aplicación corriendo secuencialmente y la misma aplicación corriendo en múltiples procesadores; indica la ganancia lograda al aumentar el número de procesadores. Es decir:

$$S_p \overset{def}{=} \frac{T_1}{T_p} \tag{2.49}$$

donde p es el número de procesadores, T_1 es el tiempo de ejecución de el algoritmo secuencial y T_p es el tiempo de ejecución del algoritmo en paralelo con p procesadores.

La siguiente métrica, la **eficiencia** [4], se determina como el speed-up entre el número de procesadores para ese *speedup*; indica cuan eficientemente se están ocupando los recursos. La siguiente ecuación lo define

$$E_p \overset{def}{=} \frac{S_p}{p} = \frac{T_1}{pT_p} \tag{2.50}$$

2.9. Resumen

En este capítulo se presenta un acercamiento a la formulación axiomática desarrollada por el Dr. Ismael Herrera y otros.

Se basa en una analogía similar a los balances en contabilidad financiera, es decir, aplicar los conceptos de patrimonio neto, activo y pasivo, donde el patrimonio neto es igual a los activos menos los pasivos; de manera parecida identificamos las llamadas propiedades extensivas que intervienen en el modelo y las propiedades intensivas, pasando después a la aplicación de la condición de balance a lo largo del tiempo. Y se extiende ésta idea para la MMC de sistemas multifase.

También se muestran los métodos de solución para sistemas de ecuaciones lineales. En específico se abordan los métodos iterativos como Jacobi, SOR, GMRES, CG, etc.

Una vez establecidos los métodos de solución, nos enfocamos en la descripción de las bibliotecas científicas de alto desempeño, así como el modelo para cómputo en paralelo y el tratamiento de los tipos de datos.

Finalizamos el capítulo con un vistazo general al software comercial y no comercial para la modelación matemática de problemas en aguas subterraneas en 3D como FEFLOW, MODFLOW, UTCHEM; además de los proyectos de software, FEniCS (para casos secuenciales y su paralelización) y TUNAM , donde se desarrolla el código para la solución de los casos de estudio y del problema de contaminantes a analizar. Además de una descripción de las métricas para cálculo de la eficiencia en implementaciones de rutinas en paralelo.

3

Modelo de Flujo y Transporte

Aplicando los conceptos de la formulación axiomática, expuestos en el capítulo anterior, obtendremos los modelos de flujo y transporte que usaremos.

3.1. Modelo de flujo de fluidos a través de medios porosos

El modelo se basa únicamente en una propiedad extensiva: la masa del fluido, que escribimos como:

$$M_f(t) \stackrel{def}{=} \int_{B(t)} \phi(\underline{x}, t)\, \rho(\underline{x}, t)\, d\underline{x} \tag{3.1}$$

Donde ϕ es la porosidad definida por

$$\phi(\underline{x}, t) \stackrel{def}{=} \frac{\text{Volumen de Poros}}{\text{Volumen Total}} \tag{3.2}$$

y ρ la densidad. Ahora sí, podemos escribir la ecuación de balance global después de haber hallado la única propiedad extensiva del modelo, M_f y la intensiva, $\phi(\underline{x}, t)\, \rho(\underline{x}, t)$; entonces escribimos la ecuación de *flujo de fluido a través de un medio poroso* así

$$\frac{\partial}{\partial t}(\phi\rho) + \nabla \cdot (\phi\rho\underline{v}) = g + \nabla \cdot \underline{\tau} \tag{3.3}$$

en algunos casos, no existe la *difusión*, entonces podemos asumir que $\underline{\tau} = 0$ y además $g = 0$ si no hay fuentes, entonces

$$\frac{\partial}{\partial t}(\phi\rho) + \nabla \cdot (\phi\rho\underline{v}) = 0 \tag{3.4}$$

En estudios de agua subterránea regional, se incorpora frecuentemente la inyección de agua por pozos, como una fuente de alimentación externa distribuida, por lo que $g \neq 0$ y la ecuación es

$$\frac{\partial}{\partial t}(\phi\rho) + \nabla \cdot (\phi\rho\underline{v}) = g \tag{3.5}$$

Tomando el término de la derivada temporal de $\phi\rho$ y descomponiéndolo mediante la fórmula de la derivada de un producto, tenemos

$$\frac{\partial \phi\rho}{\partial t} = \phi\frac{\partial \rho}{\partial t} + \rho\frac{\partial \phi}{\partial t} \tag{3.6}$$

aquí el término $\frac{\partial \rho}{\partial t}$ implica una contribución con la compresibilidad del fluido y $\frac{\partial \phi}{\partial t}$ a la elasticidad de la matriz de sólidos.

Ahora asumimos que el fluido satisface una ecuación de estado, la cual permite expresar la densidad como una función de la presión exclusivamente, $\rho = \rho(p)$, la presión como una función de la posición $\underline{x}$ y el tiempo t, $\rho = \rho(p(\underline{x}, t))$ y también a la porosidad ϕ como una función de la presión tenemos $\phi = \phi(p)$. El primer término de ecuación (3.5) queda

$$\frac{\partial \phi\rho}{\partial t} = \phi\frac{d\rho}{dp}\frac{\partial p}{\partial t} + \rho\frac{d\phi}{dp}\frac{\partial p}{\partial t} \tag{3.7}$$

Si definimos γ y β como

$$\gamma \stackrel{def}{=} \frac{d\phi}{dp} \tag{3.8}$$

$$\beta \stackrel{def}{=} \frac{1}{\rho}\frac{d\rho}{dp} \tag{3.9}$$

y las sustituimos ambas en la ecuación (3.7), tenemos

$$\begin{aligned} \frac{\partial \phi\rho}{\partial t} &= \beta\phi\rho\frac{\partial p}{\partial t} + \gamma\rho\frac{\partial p}{\partial t} \\ \frac{\partial \phi\rho}{\partial t} &= (\beta\phi + \gamma)\,\rho\frac{\partial p}{\partial t} \end{aligned} \tag{3.10}$$

y considerando al coeficiente de almacenamiento específico como

$$S_s = \rho\hat{g}(\gamma + \beta\phi) \tag{3.11}$$

substituyendo en la ecuación (3.10)

$$\frac{\partial \phi\rho}{\partial t} = \frac{1}{\hat{g}}S_s\frac{\partial p}{\partial t} \tag{3.12}$$

3.1.1. Uso de la Ley de Darcy

Una diferencia importante entre los modelos de flujo de fluido y los de transporte de soluto, es que la velocidad de la partícula no está dada como un dato en la ecuación de flujo de fluido, pero puede ser derivada usando la sustitución muy común de la *Ley de Darcy* [7] que relaciona la velocidad del fluido con el espacio de distribución de la presión del fluido (3.5)

$$\underline{U} \stackrel{def}{=} \phi\underline{v} = -\frac{1}{\mu}\underline{\underline{k}} \cdot \left(\nabla p - \rho\underline{\hat{g}}\right) \tag{3.13}$$

Donde $\underline{\hat{g}}$ es el vector de la aceleración debida a la gravedad, μ es la viscosidad dinámica del fluido, $\underline{\underline{k}}$ es el tensor de permeabilidad intrínseca, p es la presión del fluido y $\underline{U}$ es la velocidad de Darcy.

Sustituyendo esta velocidad en la ecuación (3.5)

$$\frac{\partial}{\partial t}(\phi\rho) + \nabla \cdot (\rho \underline{U}) = g \tag{3.14}$$

Tomando a z como la altura con respecto al nivel de referencia y $\hat{g}$ el módulo del vector aceleración de la gravedad, podemos reescribir a $\underline{\hat{g}}$ como

$$\underline{\hat{g}} = -\hat{g}\nabla z \tag{3.15}$$

entonces la ley de Darcy la podemos escribir también como

$$\underline{U} = -\frac{1}{\mu}\underline{\underline{k}} \cdot (\nabla p + \rho \hat{g} \nabla z) \tag{3.16}$$

3.1.2. Aplicación del Nivel piezométrico.

Para cualquier tiempo t y cualquier punto $\underline{x}$ de un medio poroso saturado, el *nivel piezométrico* [7] se define como

$$h(\underline{x}, t) \overset{def}{=} \frac{1}{\hat{g}} \int_{p_0}^{p(\underline{x},t)} \frac{d\xi}{\rho(\xi)} + z(\underline{x}) \tag{3.17}$$

obteniendo el gradiente de la función anterior y considerando que el fluido es incompresible

$$\nabla h = \frac{1}{\hat{g}\rho}\nabla p + \nabla z \tag{3.18}$$

con lo anterior podemos escribir la ley de Darcy en términos del nivel piezométrico

$$\underline{U} = -\frac{\hat{g}\rho}{\mu}\underline{\underline{k}} \cdot (\nabla h) = -\underline{\underline{K}} \cdot \nabla h \tag{3.19}$$

con $\underline{\underline{K}}$ es el tensor de conductividad hidráulica. Y también notamos que de la ecuación (3.17) obtenemos

$$\frac{\partial h}{\partial t} = \frac{1}{\hat{g}\rho}\frac{\partial p}{\partial t} \tag{3.20}$$

o bien

$$\frac{\partial p}{\partial t} = \hat{g}\rho\frac{\partial h}{\partial t} \tag{3.21}$$

aplicando el resultado anterior a la ecuación (3.12)

$$\frac{\partial \phi\rho}{\partial t} = \rho S_s \frac{\partial h}{\partial t} \tag{3.22}$$

y esta última en la ecuación (3.14)

$$\rho S_s \frac{\partial h}{\partial t} + \nabla \cdot (\rho \underline{U}) = g \tag{3.23}$$

aplicamos propiedades de la divergencia

$$\rho S_s \frac{\partial h}{\partial t} + \underline{U} \cdot \nabla \rho + \rho \nabla \cdot \underline{U} \quad = \quad g \tag{3.24}$$

dividimos por ρ lo anterior

$$S_s \frac{\partial h}{\partial t} + \underline{U} \cdot \nabla \ln \rho + \nabla \cdot \underline{U} = \frac{g}{\rho} \tag{3.25}$$

puede despreciarse el término involucrado con $\nabla \ln \rho$, entonces la ecuación nos queda

$$S_s \frac{\partial h}{\partial t} + \nabla \cdot \underline{U} = \frac{g}{\rho} \tag{3.26}$$

substituyendo la ecuación (3.19) en la anterior, tendremos la *ecuación general de flujo*

$$S_s \frac{\partial h}{\partial t} - \nabla \cdot \left(\underline{\underline{K}} \cdot \nabla h \right) = \frac{g}{\rho} \tag{3.27}$$

y si estamos considerando un problema donde no hay fuentes, $g = 0$, la ecuación de flujo se transforma en

$$S_s \frac{\partial h}{\partial t} = \nabla \cdot \left(\underline{\underline{K}} \cdot \nabla h \right) \tag{3.28}$$

y si es isotrópico, es decir, $\underline{\underline{K}} = K \underline{\underline{I}}$

$$S_s \frac{\partial h}{\partial t} = K \nabla^2 h \tag{3.29}$$

El movimiento tridimensional del agua subterránea de densidad constante a través de un medio poroso, heterogéneo y anisotrópico es descrito por la siguiente ecuación diferencial parcial

$$\nabla \cdot \left(\underline{\underline{K}} \cdot \nabla h \right) = S_s \frac{\partial h}{\partial t} - \frac{g}{\rho} \tag{3.30}$$

y para un caso en particular (nuestros modelos a analizar)

$$\frac{\partial}{\partial x} \left(K_x \frac{\partial h}{\partial x} \right) + \frac{\partial}{\partial y} \left(K_y \frac{\partial h}{\partial y} \right) + \frac{\partial}{\partial z} \left(K_z \frac{\partial h}{\partial z} \right) = S_s \frac{\partial h}{\partial t} - R \tag{3.31}$$

donde K_x, K_y y K_z son los valores de la conductividad hidráulica a lo largo de los ejes coordenados x, y y z, los cuales se asumen que son paralelos a los ejes principales de la conductividad hidráulica, h es la carga piezométrica.

R es el flujo volumétrico por unidad de volumen y representa las fuentes y/o sumideros de agua, S_s es el coeficiente de almacenamiento específico del material poroso y t es el tiempo.

3.2. Modelo de transporte

Para este tipo de modelo nuestra propiedad extensiva asociada es la masa del soluto, $M_s(t)$ contenida en un cuerpo de un medio poroso, viene dada por la integral

$$M_s(t) = \int_{B(t)} \phi(\underline{x}, t)\, c(\underline{x}, t)\, d\underline{x} \tag{3.32}$$

aquí $c(\underline{x}, t)$ es la concentración del soluto, es decir, la masa de soluto por unidad de volumen del fluido y $\varepsilon(\underline{x}, t)$ es la porosidad. La ecuación de balance global se puede escribir

$$\frac{dM_s}{dt}(t) = \int_{B(t)} g_s(\underline{x}, t)\, d\underline{x} + \int_{\partial B(t)} \underline{\tau}_s(\underline{x}, t) \cdot \underline{n}(\underline{x}, t)\, d\underline{x} \tag{3.33}$$

Aplicando la formulación axiomática, la ecuación gobernante es

$$\frac{\partial \phi c}{\partial t} + \nabla \cdot (\phi c \underline{v}) = g_s + \nabla \cdot \underline{\tau}_s \tag{3.34}$$

3.2.1. Análisis de dispersión-difusión

Los procesos difusivos para el transporte de soluto en medios porosos son modelados por la *Ley de Fick* [7, 8] como:

$$\underline{\tau}_s(\underline{x}, t) = \phi \underline{\underline{D}} \nabla c \tag{3.35}$$

sustituyendo en la ecuación (3.34) tenemos

$$\frac{\partial \phi c}{\partial t} + \nabla \cdot (\phi c \underline{v}) = g_s + \nabla \cdot \left(\phi \underline{\underline{D}} \nabla c\right) \tag{3.36}$$

usando la definición de la derivada material

$$\frac{D(*)}{Dt} \overset{def}{=} \frac{\partial (*)}{\partial t} + \underline{v} \cdot \nabla (*) \tag{3.37}$$

de la ecuación (3.36) se obtiene

$$\begin{aligned}
\frac{\partial \phi c}{\partial t} + \nabla \cdot (\phi c \underline{v}) &= g_s + \nabla \cdot \left(\phi \underline{\underline{D}} \nabla c\right) \\
\frac{\partial \phi c}{\partial t} + \underline{v} \cdot \nabla \phi c + \phi c \nabla \cdot \underline{v} &= \text{"} \\
\frac{D \phi c}{Dt} + \phi c \nabla \cdot \underline{v} &= \text{"} \\
\phi c \left(\frac{1}{\phi c} \frac{D \phi c}{Dt} + \nabla \cdot \underline{v} \right) &= \text{"} \\
\phi c \left(\frac{1}{c} \frac{Dc}{Dt} + \frac{1}{\phi} \frac{D\phi}{Dt} + \nabla \cdot \underline{v} \right) &= \text{"} \\
\phi c \left(\frac{D \ln c}{Dt} + \frac{D \ln \phi}{Dt} + \nabla \cdot \underline{v} \right) &= g_s + \nabla \cdot \left(\phi \underline{\underline{D}} \nabla c\right)
\end{aligned} \tag{3.38}$$

a su vez la última ecuación se puede transformar en

$$\begin{aligned}
\frac{1}{\phi}\left[\phi c\left(\frac{D\ln c}{Dt}+\frac{D\ln\phi}{Dt}+\nabla\cdot\underline{v}\right)\right] &= \frac{1}{\phi}\left[g_s+\nabla\cdot\left(\phi\underline{\underline{D}}\nabla c\right)\right]\\
c\left(\frac{D\ln c}{Dt}+\frac{D\ln\phi}{Dt}+\nabla\cdot\underline{v}\right) &= \frac{1}{\phi}g_s+\frac{1}{\phi}\left(\nabla\cdot\left(\phi\underline{\underline{D}}\nabla c\right)\right)\\
c\frac{D\ln c}{Dt}+c\frac{D\ln\phi}{Dt}+c\nabla\cdot\underline{v} &= \phi^{-1}g_s+\phi^{-1}\left(\nabla\cdot\left(\phi\underline{\underline{D}}\nabla c\right)\right)\\
c\frac{1}{c}\frac{Dc}{Dt}+c\frac{D\ln\phi}{Dt}+c\nabla\cdot\underline{v} &= \phi^{-1}g_s+\phi^{-1}\left(\underline{\underline{D}}\nabla c\cdot\nabla\phi+\phi\nabla\cdot\left(\underline{\underline{D}}\nabla c\right)\right)\\
\frac{Dc}{Dt}+c\frac{D\ln\phi}{Dt}+c\nabla\cdot\underline{v} &= \phi^{-1}g_s+\underline{\underline{D}}\nabla c\cdot(\nabla\ln\phi)+\nabla\cdot\left(\underline{\underline{D}}\nabla c\right)\\
\frac{Dc}{Dt}+c\left(\frac{D\ln\phi}{Dt}+\nabla\cdot\underline{v}\right) &= \phi^{-1}g_s+(\nabla\ln\phi)\cdot\underline{\underline{D}}\nabla c+\nabla\cdot\left(\underline{\underline{D}}\nabla c\right)
\end{aligned}$$

que es el resultado más general de *la ecuación gobernante para transporte de solutos en medios porosos [16].* A continuación derivamos algunos casos particulares:

- Cuando el fluido es incompresible, se cumple

$$\frac{\partial\phi}{\partial t}+\nabla\cdot(\phi\underline{v})=0 \tag{3.39}$$

  o equivalentemente para nuestro caso

$$\frac{D\ln\phi}{Dt}+\nabla\cdot\underline{v}=0 \tag{3.40}$$

  por lo tanto, la ecuación gobernante se reduce a

$$\frac{Dc}{Dt}=\phi^{-1}g_s+(\nabla\ln\phi)\cdot\underline{\underline{D}}\nabla c+\nabla\cdot\left(\underline{\underline{D}}\nabla c\right) \tag{3.41}$$

- Si además, sabemos que la matriz de sólidos es homogénea, entonces el término $\nabla\ln\phi$ puede ser despreciado, pasando a

$$\frac{Dc}{Dt}=\phi^{-1}g_s+\nabla\cdot\left(\underline{\underline{D}}\nabla c\right) \tag{3.42}$$

  o bien

$$\frac{\partial c}{\partial t}+\underline{v}\cdot\nabla c=\phi^{-1}g_s+\nabla\cdot\left(\underline{\underline{D}}\nabla c\right) \tag{3.43}$$

- Añadiendo, si el fluido esta en reposo ($\underline{v}=0$), reducimos aún más en

$$\frac{\partial c}{\partial t}=\phi^{-1}g_s+\nabla\cdot\left(\underline{\underline{D}}\nabla c\right) \tag{3.44}$$

- Para transporte no difusivo pasamos de

$$\frac{\partial c}{\partial t} + \underline{v} \cdot \nabla c = \phi^{-1} g_s + \nabla \cdot \left(\underline{\underline{D}} \nabla c \right) \tag{3.45}$$

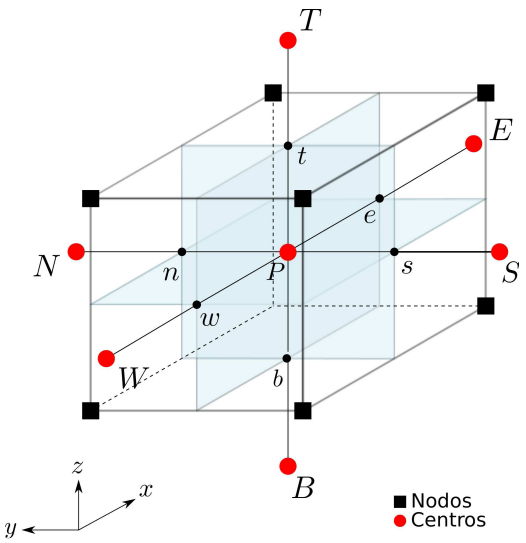
a

$$\frac{\partial c}{\partial t} + \underline{v} \cdot \nabla c = \phi^{-1} g_s \tag{3.46}$$

Todas estas ecuaciones diferenciales gobernantes, son ampliamente utilizados en la modelación de aguas subterráneas con contaminantes. Muestran de forma explícita como en el transporte de fluidos libres y en el transporte de solutos por un fluido en medios porosos, la velocidad de advección es también la velocidad de la partícula del fluido.

3.3. Discretización usando el método de volumen finito

El paso clave de el método de volumen finito es la integración de la ecuación(o ecuaciones) diferencial gobernante bajo un volumen de control, para obtener una ecuación discretizada en su punto nodal P. Véase en la figura 3.1.



con $\Delta V = \Delta x \Delta y \Delta z$

Figura 3.1: Una celda en tres dimensiones y sus nodos vecinos

En la figura vemos una celda que contiene al nodo P y a sus seis nodos vecinos más cercanos identificados por: oeste, este, sur, norte, los nodos superiores e inferiores (W, E, S, N, T, B).

Como antes, la notación, w, e, s, n, t y b son usados para referirse a las caras oeste, este, sur, norte, las superiores y las inferiores, respectivamente.

Haciendo la integral de la ecuación (3.31) en el volumen de control, tenemos

$$\int_{\Delta V} \frac{\partial}{\partial x}\left(K_x \frac{\partial h}{\partial x}\right) dV + \int_{\Delta V} \frac{\partial}{\partial y}\left(K_y \frac{\partial h}{\partial y}\right) dV + \int_{\Delta V} \frac{\partial}{\partial z}\left(K_z \frac{\partial h}{\partial z}\right) dV = \int_{\Delta V} S_s \frac{\partial h}{\partial t} dV - \int_{\Delta V} R dV \tag{3.47}$$

Reescribimos la primera integral de la ecuación anterior como

$$\int_{\Delta V} \frac{\partial}{\partial x}\left(K_x \frac{\partial h}{\partial x}\right) dV = \int_f^b \int_s^n \int_w^e \frac{\partial}{\partial x}\left(K_x \frac{\partial h}{\partial x}\right) dx dy dz \tag{3.48}$$

asumiendo a K_x constante dentro del volumen de control e integrando sobre x tenemos

$$\int_f^b \int_s^n K_x \frac{\partial h}{\partial x}\bigg|_w^e dy dz \tag{3.49}$$

si evaluamos en las caras w y e

$$\int_f^b \int_s^n \left[K_e \left(\frac{\partial h}{\partial x}\right)_e - K_w \left(\frac{\partial h}{\partial x}\right)_w\right] dy dz \tag{3.50}$$

separando la integral en dos

$$\int_f^b \int_s^n K_e \left(\frac{\partial h}{\partial x}\right)_e dy dz - \int_f^b \int_s^n K_w \left(\frac{\partial h}{\partial x}\right)_w dy dz \tag{3.51}$$

o

$$A_e K_e \left(\frac{\partial h}{\partial x}\right)_e - A_w K_w \left(\frac{\partial h}{\partial x}\right)_w \tag{3.52}$$

donde A_e y A_w son las áreas de las caras e y w, respectivamente. Así de manera análoga integramos para las otras dos dimensiones tomando en cuenta las caras $\{(n, s), (t, b)\}$, entonces la ecuación (3.47) queda

$$\begin{aligned} &\left[A_e K_e \left(\frac{\partial h}{\partial x}\right)_e - A_w K_w \left(\frac{\partial h}{\partial x}\right)_w\right] \\ &+ \left[A_n K_n \left(\frac{\partial h}{\partial x}\right)_n - A_s K_s \left(\frac{\partial h}{\partial x}\right)_s\right] \\ &+ \left[A_t K_t \left(\frac{\partial h}{\partial x}\right)_t - A_b K_b \left(\frac{\partial h}{\partial x}\right)_b\right] \\ &= \int_{\Delta V} S_s \frac{\partial h}{\partial t} dV - \int_{\Delta V} R dV \end{aligned} \tag{3.53}$$

Del lado derecho de la ecuación anterior, expresamos el segundo término como

$$\int_{\Delta V} R dV = \int_f^b \int_s^n \int_w^e R dx dy dx \tag{3.54}$$

o bien, aproximando

$$\int_f^b \int_s^n \int_w^e R dx dy dx \approx \int_f^b \int_s^n \int_w^e \overline{R} dx dy dx \tag{3.55}$$

en esta aproximación estamos considerando a $\overline{R}$ como un valor representativo de cada volumen de control y además constante, por lo que el término resulta

$$\int_{\Delta V} R dV \approx \overline{R} \int_f^b \int_s^n \int_w^e dx dy dx = \overline{R} \Delta V = \overline{R} \Delta x \Delta y \Delta z \tag{3.56}$$

Ahora nos queda discretizar el término de la derivada con respecto a t y las derivadas de h con respecto a cada una de las variables espaciales.

Para esto usaremos un esquema de diferencias finitas hacia adelante, así pues usando esquema θ implícito tenemos

$$\left(\frac{\partial h^{n+1}}{\partial x}\right)_e \approx \frac{h_E^{n+1} - h_P^{n+1}}{\Delta x_{PE}} \tag{3.57}$$

$$\left(\frac{\partial h^{n+1}}{\partial x}\right)_w \approx \frac{h_P^{n+1} - h_W^{n+1}}{\Delta x_{WP}} \tag{3.58}$$

$$\left(\frac{\partial h^{n+1}}{\partial y}\right)_n \approx \frac{h_N^{n+1} - h_P^{n+1}}{\Delta y_{PN}} \tag{3.59}$$

$$\left(\frac{\partial h^{n+1}}{\partial y}\right)_s \approx \frac{h_P^{n+1} - h_S^{n+1}}{\Delta y_{SP}} \tag{3.60}$$

$$\left(\frac{\partial h^{n+1}}{\partial z}\right)_t \approx \frac{h_T^{n+1} - h_P^{n+1}}{\Delta z_{PT}} \tag{3.61}$$

$$\left(\frac{\partial h^{n+1}}{\partial z}\right)_b \approx \frac{h_P^{n+1} - h_B^{n+1}}{\Delta z_{BP}} \tag{3.62}$$

y de manera similar hacemos la consideración de que la derivada temporal también la podemos proponer como una diferencia finita hacia adelante, por lo tanto

$$\int_{\Delta V} S_s \frac{\partial h}{\partial t} dV = \int_f^b \int_s^n \int_w^e S_s \frac{h_P^{n+1} - h_P^n}{\Delta t} dx dy dx = S_s \frac{h_P^{n+1} - h_P^n}{\Delta t} \Delta x \Delta y \Delta z \tag{3.63}$$

Si sustituimos las ecuaciones: (3.57)-(3.62), (3.63) y (3.56) en la ecuación (3.53), lle-

gamos finalmente a lo siguiente

$$\begin{aligned}
&\left[K_e\frac{\left(h_E^{n+1}-h_P^{n+1}\right)A_e}{\Delta x_{PE}}-K_w\frac{\left(h_P^{n+1}-h_W^{n+1}\right)A_w}{\Delta x_{WP}}\right]\\
&+\left[K_n\frac{\left(h_N^{n+1}-h_P^{n+1}\right)A_n}{\Delta y_{PN}}-K_s\frac{\left(h_P^{n+1}-h_S^{n+1}\right)A_s}{\Delta y_{SP}}\right]\\
&+\left[K_t\frac{\left(h_T^{n+1}-h_P^{n+1}\right)A_t}{\Delta z_{PT}}-K_b\frac{\left(h_P^{n+1}-h_B^{n+1}\right)A_b}{\Delta z_{BP}}\right]\\
&=S_s\frac{h_P^{n+1}+h_P^n}{\Delta t}\Delta x\Delta y\Delta z-\overline{R}\Delta x\Delta y\Delta z \qquad (3.64)
\end{aligned}$$

Esta ecuación se puede reordenar de la siguiente manera

$$\begin{aligned}
&\left(\frac{K_eA_e}{\Delta x_{PE}}+\frac{K_wA_w}{\Delta x_{WP}}+\frac{K_nA_n}{\Delta y_{PN}}+\frac{K_sA_s}{\Delta y_{SP}}+\frac{K_tA_t}{\Delta z_{PT}}+\frac{K_bA_b}{\Delta z_{BP}}-\frac{S_s\Delta V}{\Delta t}\right)h_P^{n+1}\\
&=\left(\frac{K_eA_e}{\Delta x_{PE}}\right)h_E^{n+1}+\left(\frac{K_wA_w}{\Delta x_{WP}}\right)h_W^{n+1}+\left(\frac{K_nA_n}{\Delta y_{PN}}\right)h_N^{n+1}+\left(\frac{K_sA_s}{\Delta y_{SP}}\right)h_S^{n+1}\\
&\quad+\left(\frac{K_tA_t}{\Delta z_{PT}}\right)h_T^{n+1}+\left(\frac{K_bA_b}{\Delta z_{BP}}\right)h_B^{n+1}+\left(S_s\frac{h_P^n}{\Delta t}-\overline{R}\right)\Delta V \qquad (3.65)
\end{aligned}$$

La ecuación (3.65) arroja la ecuación de discretización general para los nodos interiores:

$$\begin{aligned}
a_P^{n+1}h_P^{n+1} &= a_E^{n+1}h_E^{n+1}+a_W^{n+1}h_W^{n+1}+a_N^{n+1}h_N^{n+1}+a_S^{n+1}h_S^{n+1}\\
&\quad+a_T^{n+1}h_T^{n+1}+a_B^{n+1}h_B^{n+1}+\left(S_s\frac{h_P^n}{\Delta t}-\overline{R}\right)\Delta V \qquad (3.66)
\end{aligned}$$

donde los coeficiente están definidos como sigue

$$a_E^{n+1} = \frac{K_eA_e}{\Delta x_{PE}} \qquad (3.67)$$

$$a_W^{n+1} = \frac{K_wA_w}{\Delta x_{WP}} \qquad (3.68)$$

$$a_N^{n+1} = \frac{K_nA_n}{\Delta y_{PN}} \qquad (3.69)$$

$$a_S^{n+1} = \frac{K_sA_s}{\Delta y_{SP}} \qquad (3.70)$$

$$a_T^{n+1} = \frac{K_tA_t}{\Delta z_{PT}} \qquad (3.71)$$

$$a_B^{n+1} = \frac{K_bA_b}{\Delta z_{BP}} \qquad (3.72)$$

$$a_P^{n+1} = a_E^{n+1}+a_W^{n+1}+a_N^{n+1}+a_S^{n+1}+a_T^{n+1}-\frac{S_s\Delta V}{\Delta t} \qquad (3.73)$$

3.4. Discretización usando el método de elemento finito

Primero necesitamos reescribir el problema en forma variacional para esto se define el producto escalar de funciones por pedazos u y v

$$(u, v) = \int u(x)\, v(x)\, dx \tag{3.74}$$

Introducimos también, el siguiente espacio vectorial de funciones:

$$\begin{aligned} V &= \{v | v \text{ es una función continua,} \\ &\quad v' \text{ es continua a pedazos y acotada}\} \end{aligned}$$

Tomamos $v \in V$ y multiplicamos por la ecuación (3.31) e integramos

$$\int_\Omega v\frac{\partial}{\partial x}\left(K_x\frac{\partial h}{\partial x}\right)d\Omega + \int_\Omega v\frac{\partial}{\partial y}\left(K_y\frac{\partial h}{\partial y}\right)d\Omega + \int_\Omega v\frac{\partial}{\partial z}\left(K_z\frac{\partial h}{\partial z}\right)d\Omega = \int_\Omega S_s\frac{\partial h}{\partial t}vd\Omega - \int_\Omega Rvd\Omega \tag{3.75}$$

de otra forma

$$\int_\Omega v\left[\frac{\partial}{\partial x}\left(K_x\frac{\partial h}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y\frac{\partial h}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z\frac{\partial h}{\partial z}\right)\right]d\Omega = \int_\Omega S_s\frac{\partial h}{\partial t}vd\Omega - \int_\Omega Rvd\Omega \tag{3.76}$$

asumiendo que los valores de la conductividad hidráulica en los tres ejes coordenados son constantes para cada elemento, tenemos

$$\int_\Omega v\left[K_x\frac{\partial^2 h}{\partial x^2} + K_y\frac{\partial^2 h}{\partial y^2} + K_z\frac{\partial^2 h}{\partial z^2}\right]d\Omega = \int_\Omega S_s\frac{\partial h}{\partial t}vd\Omega - \int_\Omega Rvd\Omega \tag{3.77}$$

usando el operador ∇, escribimos

$$\frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2} + \frac{\partial^2 h}{\partial z^2} = \nabla\cdot(\nabla h) \tag{3.78}$$

sustituyendo lo anterior en la ecuación (3.77), tenemos

$$\int_\Omega Kv\nabla\cdot(\nabla h)\, d\Omega = \int_\Omega S_s\frac{\partial h}{\partial t}vd\Omega - \int_\Omega Rvd\Omega \tag{3.79}$$

o bien

$$\int_\Omega v\left(K\nabla^2 h\right)d\Omega = \int_\Omega S_s\frac{\partial h}{\partial t}vd\Omega - \int_\Omega Rvd\Omega \tag{3.80}$$

$$K\int_\Omega v\nabla^2 h d\Omega = \int_\Omega S_s\frac{\partial h}{\partial t}vd\Omega - \int_\Omega Rvd\Omega \tag{3.81}$$

luego el **teorema de Green**[1] nos dice

$$\int_\Omega v\nabla^2 h d\Omega = -\int_\Omega(\nabla v)\cdot(\nabla h)\, d\Omega + \int_{\partial\Omega} v\nabla h\cdot\underline{n}\partial\Omega \tag{3.82}$$

[1] En un dominio Ω, consideramos la frontera $\partial\Omega$ y el vector normal $\underline{n}$ al dominio Ω.

directamente aplicamos este teorema en la ecuación (3.80)

$$-K\int_{\Omega}(\nabla v)\cdot(\nabla h)\,d\Omega+K\int_{\partial\Omega}v\nabla h\cdot\underline{n}\partial\Omega=\int_{\Omega}S_s\frac{\partial h}{\partial t}vd\Omega-\int_{\Omega}Rvd\Omega \quad (3.83)$$

por lo que nuestra ecuación se reduce a

$$-K\int_{\Omega}(\nabla v)\cdot(\nabla h)\,d\Omega=\int_{\Omega}S_s\frac{\partial h}{\partial t}vd\Omega-\int_{\Omega}Rvd\Omega \quad (3.84)$$

Si tomamos la aproximación de la primera derivada temporal como la diferencia hacia adelante, podemos escribirla de la siguiente manera

$$\frac{\partial h}{\partial t}\approx\frac{h^{n+1}-h^{n}}{\Delta t} \quad (3.85)$$

usando esto en la ecuación (3.84), consideramos a S_s como un valor constante, así

$$-K\int_{\Omega}\nabla v\cdot\nabla h^{n+1}d\Omega=S_s\int_{\Omega}\frac{h^{n+1}-h^{n}}{\Delta t}vd\Omega-\int_{\Omega}Rvd\Omega \quad (3.86)$$

o bien

$$S_s\int_{\Omega}\frac{h^{n+1}}{\Delta t}vd\Omega+K\int_{\Omega}\nabla v\cdot\nabla h^{n+1}d\Omega-R\int_{\Omega}vd\Omega=\int_{\Omega}\frac{h}{\Delta t}vd\Omega \quad (3.87)$$

Definimos una aproximación de h en el método de elemento finito de la forma

$$h^{n}(x,y,z)=\sum_{i=1}^{M}h_i^{n}\varphi_i(x,y,z) \quad (3.88)$$

y de la misma forma para h^{n+1}

$$h^{n+1}(x,y,z)=\sum_{i=1}^{M}h_i^{n+1}\varphi_i(x,y,z) \quad (3.89)$$

reemplazando las ecuaciones (3.88) y(3.89) en la ecuación (3.87)

$$S_s\int_{\Omega}\frac{1}{\Delta t}\left(\sum_{i=1}^{M}h_i^{n+1}\varphi_i\right)vd\Omega+K\int_{\Omega}\nabla v\cdot\nabla\left(\sum_{i=1}^{M}h_i^{n+1}\varphi_i\right)d\Omega-\int_{\Omega}Rvd\Omega \quad = \quad \int_{\Omega}\frac{1}{\Delta t}\left(\sum_{i=1}^{M}h_i^{n}\varphi_i\right)vd\Omega \quad (3.90)$$

podemos intercambiar la suma y la integral de la siguiente manera

$$S_s\frac{1}{\Delta t}\sum_{i=1}^{M}\int_{\Omega}\left(h_i^{n+1}\varphi_i\right)vd\Omega+K\sum_{i=1}^{M}\int_{\Omega}\nabla v\cdot\nabla\left(h_i^{n+1}\varphi_i\right)d\Omega-\int_{\Omega}Rvd\Omega \quad = \quad \frac{1}{\Delta t}\sum_{i=1}^{M}\int_{\Omega}\left(h_i^{n}\varphi_i\right)vd\Omega \quad (3.91)$$

escribimos a v en su forma vectorial y sacamos a h como una constante de las integrales y tenemos

$$S_s \frac{1}{\Delta t} \sum_{i=1}^{M} h_i^{n+1} \int_\Omega \varphi_i \underline{v} d\Omega + K \sum_{i=1}^{M} h_i^{n+1} \int_\Omega \nabla \underline{v} \cdot \nabla \varphi_i d\Omega - \int_\Omega R \underline{v} d\Omega = \frac{1}{\Delta t} \sum_{i=1}^{M} h_i^n \int_\Omega \varphi_i \underline{v} d\Omega \quad (3.92)$$

Usando la aproximación de Galerkin, podemos expresar a v como

$$v = \sum_{j=1}^{M} v_j \varphi_j \quad (3.93)$$

entonces la ecuación (3.92) se escribe

$$S_s \frac{1}{\Delta t} \sum_{i=1}^{M} h_i^{n+1} \int_\Omega \varphi_i \varphi_j d\Omega + K \sum_{i=1}^{M} h_i^{n+1} \int_\Omega \nabla \varphi_j \cdot \nabla \varphi_i d\Omega - \int_\Omega R \underline{v} d\Omega = \frac{1}{\Delta t} \sum_{i=1}^{M} h_i^n \int_\Omega \varphi_i \varphi_j d\Omega \quad (3.94)$$

ahora introducimos las siguientes definiciones en forma matricial, donde ^{T} denota la transposición

$$A_{ij} = \int_\Omega \varphi_i \varphi_j d\Omega \quad (3.95)$$

$$B_{ij} = \int_\Omega \nabla \varphi_i \cdot \nabla \varphi_j d\Omega \quad (3.96)$$

$$\underline{h}^n = [h_1^n, h_2^n, ..., h_M^n]^T \quad (3.97)$$

$$\underline{h}^{n+1} = \left[h_1^{n+1}, h_2^{n+1}, ..., h_M^{n+1}\right]^T \quad (3.98)$$

$$\eta = \int_\Omega R \underline{v} d\Omega \quad (3.99)$$

la última ecuación diferencial se puede escribir en forma matricial como

$$S_s \frac{1}{\Delta t} \underline{h}^{n+1} A + K \underline{h}^{n+1} B - \eta = \frac{1}{\Delta t} \underline{h}^n A \quad (3.100)$$

agrupando

$$\left(S_s \frac{1}{\Delta t} A + KB \right) \underline{h}^{n+1} - \eta = \frac{1}{\Delta t} \underline{h}^n A \quad (3.101)$$

haciendo

$$C = S_s \frac{1}{\Delta t} A + KB \quad (3.102)$$

y

$$b^n = \frac{1}{\Delta t}\underline{h}^n A + \eta \tag{3.103}$$

llegamos al siguiente sistema de ecuaciones

$$C\underline{h}^{n+1} = b^n \tag{3.104}$$

3.5. Resumen

Este capítulo presenta el desarrollo para la obtención del modelo matemático del flujo de fluidos a través de medios porosos y transporte.

El objetivo del capítulo es desarrollar las ecuaciones que resolveremos más adelante en los problemas específicos.

También, como objetivo, presentar el desarrollo para la discretización de las ecuaciones gobernantes a través de los métodos de volumen finito y elemento finito.

4

Casos de Estudio

La hidrología subterránea abarca el estudio del subsuelo y de toda la ciencia del movimiento del agua en su interior [8]. El agua que se almacena en las rocas porosas del subsuelo, representa una fracción importante de la masa total de agua presente en el planeta tierra. Este volumen de agua es mucho más importante que el que se encuentra en lagos y ríos alrededor del mundo. Sin embargo, su volumen es menor al de los mayores glaciares, pues las masas más extensas pueden alcanzar millones de kilómetros cuadrados. La razón principal por la que el agua del subsuelo es un recurso importante, es por que de ahí se abastece a una tercera parte de la población mundial. Como es de suponerse, la gestión de este recurso es difícil debido a su sensibilidad a la contaminación y a la sobreexplotación.

El agua subterránea puede encontrarse ocupando los intersticios (poros y grietas) del suelo, del sustrato rocoso o del sedimento sin consolidar, los cuales la contienen como una esponja. A estas formaciones se les denomina acuíferos. Existen también casos en los que el agua excava simas[1], cavernas y otras vías de circulación, en las rocas solubles como las calizas y los yesos, susceptibles de sufrir el proceso llamado karstificación [8].

En hidrología se denomina un *acuífero* a aquel estrato o formación geológica permeable que permite la circulación y el almacenamiento del agua subterránea por sus poros o grietas [8] (ver figura 4.1). Dentro de estas formaciones podemos encontrarnos con materiales muy variados como gravas de río, limo, calizas muy agrietadas, areniscas porosas poco cementadas, arenas de playa, algunas formaciones volcánicas, depósitos de dunas e incluso ciertos tipos de arcilla. El nivel superior del agua subterránea se denomina tabla de agua, y en el caso de un acuífero libre, corresponde al nivel freático.

La circulación subterránea tiende a depurar el agua de partículas y microorganismos contaminantes, por lo que en general el agua subterránea tiende a ser dulce y pota-

[1]Cavidad que se abre al exterior mediante un pozo o conducto vertical o en pendiente pronunciada, originada por un proceso erosivo kárstico en la roca calcárea o derrumbe del techo de una cavidad por el que el agua se filtra a niveles inferiores.

ble. Desafortunadamente, en muchos casos los acuíferos pueden contaminarse debido la actividad humana, como la construcción de fosas sépticas o la agricultura. Existen otros factores naturales de contaminación, por ejemplo si los acuíferos son demasiado ricos en sales disueltas o por la erosión natural de ciertas formaciones rocosas. La contaminación del agua subterránea puede permanecer por largos períodos de tiempo. Esto se debe a la baja tasa de renovación y largo tiempo de residencia, pues debido al difícil acceso a las zonas de almacenamiento del agua subterránea, no es fácil aplicar procesos artificiales de depuración como los que se pueden aplicar a los depósitos superficiales. En caso de zonas locales de contaminación se puede realizar remediación de acuíferos mediante la técnica de bombeo y tratamiento, que consiste en extraer agua del acuífero, tratarla químicamente, e inyectarla de vuelta al acuífero.

Entre las causas de contaminación, originadas por los seres humanos (antropogénicas) están la infiltración de nitratos y otros abonos químicos muy solubles usados en la agricultura. Estos suelen ser una causa grave de contaminación de los suministros en llanuras de elevada productividad agrícola y densa población. Otras fuentes de contaminantes son las descargas de fábricas, los productos agrícolas y los químicos utilizados por las personas en sus hogares y patios. Los contaminantes también pueden provenir de tanques de almacenamiento de agua, pozos sépticos, lugares con desperdicios peligrosos y vertederos. Actualmente, los contaminantes del agua subterránea que más preocupan son los compuestos orgánicos industriales, como disolventes, pesticidas, pinturas, barnices, o los combustibles como la gasolina.

En cuanto a los abonos químicos minerales, los nitratos son los que generan mayor preocupación. Estos se originan de diferentes fuentes: la aplicación de fertilizantes, los pozos sépticos que no están funcionando bien, las lagunas de retención de desperdicios sólidos no impermeabilizadas por debajo y la infiltración de aguas residuales o tratadas. El envenenamiento con nitrato es peligroso en los niños. En altos niveles pueden limitar la capacidad de la sangre para transportar oxígeno, causando asfixia en bebés. En el tubo digestivo el nitrato se reduce produciendo nitritos, que son cancerígenos.

El agua subterránea en áreas costeras puede contaminarse por intrusiones de agua de mar (intrusión salina) cuando la tasa de extracción es muy alta. Esto provoca que el agua del mar penetre en los acuíferos de agua dulce. Este problema puede ser tratado con cambios en la ubicación de los pozos o excavando otros que mantengan el agua salada lejos del acuífero de agua dulce. En todo caso, mientras la extracción supere a la recarga por agua dulce, la contaminación con agua salada sigue siendo una posibilidad.

El estudio de la contaminación de acuíferos es de vital importancia para la humanidad. Actualmente existen varias tecnologías para estudiar y entender la dinámica del flujo de agua subterránea. Es posible, mediante la aplicación de leyes físicas, químicas y biológicas, generar modelos matemáticos que gobiernan las fenómenos que ocurren en el transporte de contaminantes en acuíferos. La solución adecuada de estos modelos matemáticos pueden proporcionar reglas adecuadas para el monitoreo, explotación y mantenimiento de los acuíferos, que actualmente son la base del abastecimiento de agua potable para un alto porcentaje de la población mundial.

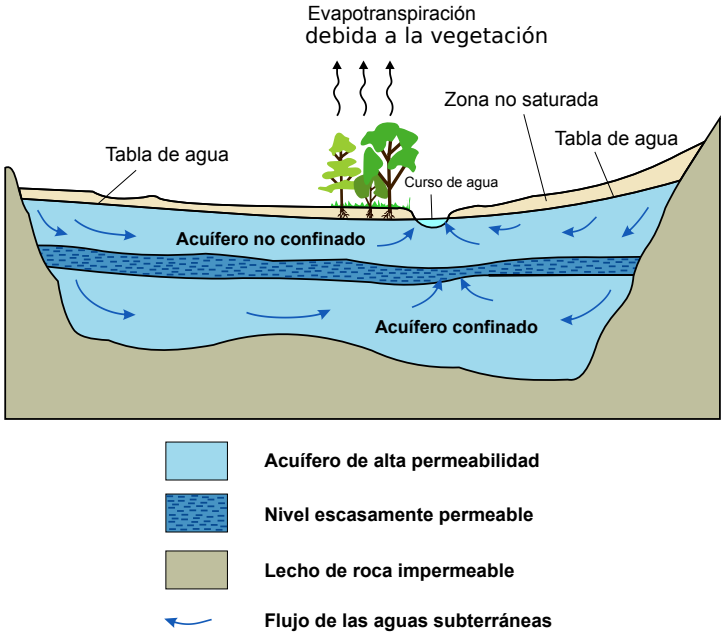


Figura 4.1: Esquema de un acuífero. Tomado de wikipedia.org

4.1. Problemas sin contaminantes

4.1.1. Problema 1

El sistema de flujo regional de agua subterránea descrito por Toth (1962) [30] es un ejemplo de un aplicación de la ecuación de Laplace.

Las fronteras izquierda y derecha del acuífero son aguas subterráneas, representada matemáticamente como impermeables, fronteras sin flujo de agua.

La frontera inferior es también de no flujo debido a la base de roca impermeable (una barrera física al flujo) el límite superior del modelo matemático describe una variación lineal en el gradiente hidráulico (igual a la altura del nivel freático).

La frontera superior se encuentra en $z = z_0$ para x que van de 0 a s. La distribución de gradiente hidráulico a lo largo de esta frontera es lineal, y a lo largo de la frontera superior hace que sea un Condición de contorno Dirichlet.

Los otras tres son fronteras de no-flujo: la especificación de flujo (incluso flujo cero) a través de estos límites hace las condiciones de contorno Neumann.

La ecuación de Laplace (véase en la figura 4.2) simula el flujo de las aguas subterráneas en un acuífero homogéneo, isotrópico sin acumulación o pérdida de agua en el

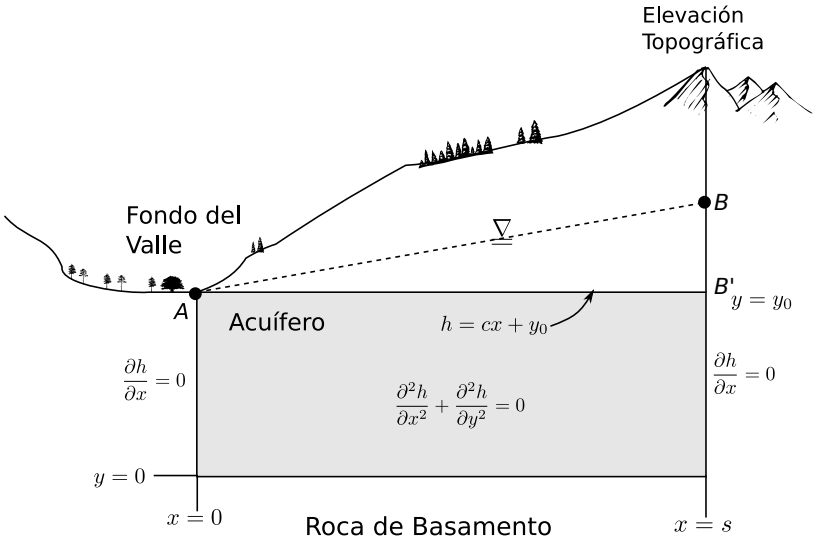


Figura 4.2: Modelo regional de un sistema de flujo de agua subterránea

sistema. Las condiciones son:

$$\begin{aligned} h(0, y_0) &= y_0 && x = 0 \\ h(x, y_0) &= cx + y_0 && 0 \leq x < s \end{aligned}$$

Solución numérica

Resolver un problema físico con FEniCS consta de los siguientes pasos:

1. Identificar las EDP[2] y sus condiciones de contorno.
2. Reformular el problema como un problema variacional.
3. Hacer un programa *Python* en que las fórmulas en el problema variacional están codificadas, junto con las definiciones de los datos de entrada, tales como f, u_0 y una malla para el dominio espacial.
4. Añadir declaraciones en el programa para resolver el problema variacional, calculando las magnitudes derivadas, tales como ∇u y la visualizar los resultados.

Formulación Variacional

Para pasar de nuestra EDP a la formulación variacional, multiplicamos la EDP por una función, integramos la ecuación resultante y realizamos la integración por partes de los términos con derivadas de segundo orden.

[2]Utilizo EDP para abreviar *Ecuaciones Diferenciales Parciales*

La función que multiplica a la EDP dentro del método de elemento finito es llamada *función de prueba*.

Multiplicamos nuestra ecuación por la función de prueba v e integramos

Se puede escribir

$$\frac{\partial^2 s}{\partial x_1^2}+\frac{\partial^2 s}{\partial x_2^2}+\frac{\partial^2 s}{\partial x_3^2}=\begin{cases}\nabla\cdot(\nabla s)=\nabla^2 s & \text{(notación directa)}\\ \dfrac{\partial^2 s}{\partial x_i \partial x_i}\end{cases} \tag{4.1}$$

$$\frac{\partial^2 h}{\partial x^2}+\frac{\partial^2 h}{\partial y^2}=\nabla\cdot(\nabla h)=\nabla^2 h$$

luego nuestra ecuación es

$$\nabla^2 h=0$$

entonces

$$\begin{aligned}\int_\Omega \left(\nabla^2 h\right) v dx &= 0\\ \int_\Omega \left(\nabla^2 h\right) v dx &= \int_\Omega \nabla h\cdot\nabla v dx-\int_{\partial\Omega}\frac{\partial h}{\partial \mathrm{n}} v ds\end{aligned}$$

donde $\frac{\partial h}{\partial n}$ es la derivada de h en la dirección normal exterior a la frontera.

El segundo término del lado derecho de la última ecuación se elimina. De esto se deduce que

$$\int_\Omega \nabla h\cdot\nabla v dx=0,\qquad \forall v\in\hat{V}$$

Los espacios *trial* y *test* están definidos como:

$$\begin{aligned}\hat{V} &= \left\{v\in H^1(\Omega): v=0 \text{ en } \partial\Omega\right\}\\ V &= \left\{v\in H^1(\Omega): v=u_0 \text{ en } \partial\Omega\right\}\end{aligned}$$

Adoptamos la siguiente notación para la formulación débil

$$a(u,v)=L(v)$$

y para nuestro problema

$$a(u,v)=\int_\Omega \nabla h\cdot\nabla v dx$$

y

$$L(v)=\int_\Omega f v dx$$

con $f=0$

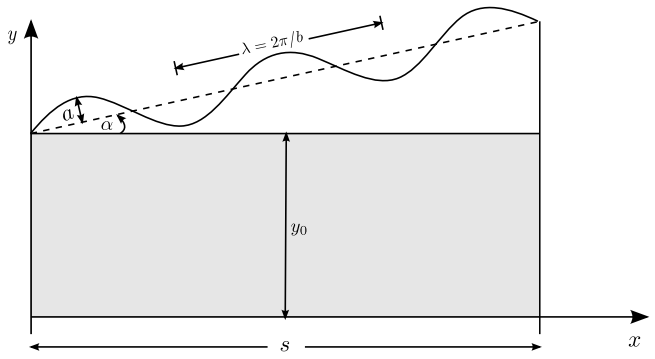


Figura 4.3: Sistema de flujo regional de agua subterránea con nivel freático senoidal

Solución analítica

Toth (1962) presentó una solución analítica al modelo del sistema de flujo regional de agua subterránea

$$h\left(x,y\right)=y_0+\frac{cs}{2}-\frac{4cs}{\pi^2}\sum_{m=0}^{\infty}\frac{\cos\left[\left(2m+1\right)\pi x/s\right]\cosh\left[\left(2m+1\right)\pi y/s\right]}{\left(2m+1\right)^2\cosh\left[\left(2m+1\right)\pi y_0/s\right]}$$

para un acuífero homogéneo e isotrópico con una configuración lineal del nivel freático.

4.1.2. Problema 2

$$h\left(x,y_0\right)=y_0+x\tan\alpha+a\left\{\left[\sin\left(bx/\cos\alpha\right)\right]/\cos\alpha\right\}$$

donde

$$\alpha=1.1^{\text{o}}\ (\text{ó}\ 0.019199\,\text{rad})$$

y

$$b=\frac{2\pi}{\lambda}$$

donde $\lambda = 80$ m, $a = 2.0$ m y una longitud $s = 220.0$ m. Esta condición de frontera fue considerada por Toth (1963).

La especificación de la carga a lo largo de la frontera superior hace que sea una condición de frontera de tipo Dirichlet.

4.1.3. Problema 3

Consideremos el siguiente acuífero confinado [30] mostrado en la siguiente figura

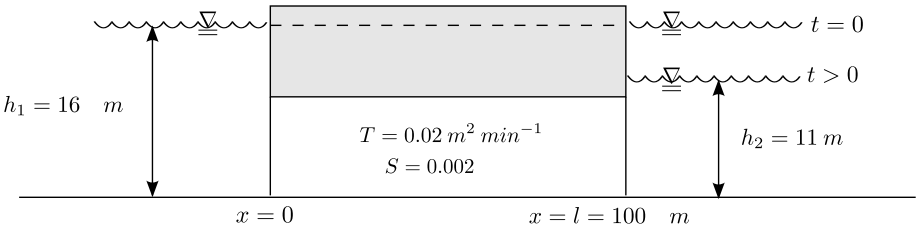


Figura 4.4: Ejemplo de depósito. El flujo es en una dimensión en la dirección de x

la carga es de 16 m en todo el acuífero.

Deseamos simular los cambios de la carga a través del tiempo en $t = 0$, repentinamente, el nivel cae en $x = l$ desde 16 m hasta 11 m.

Los parámetros del acuífero son $T = 0.02$ m^2 min^{-1} y $S = 0.002$. Despreciando el flujo en la dirección de y, la ecuación gobernante es

$$\frac{\partial^2 h}{\partial x^2} = \frac{S}{T}\frac{\partial h}{\partial t}$$

y las condiciones de frontera son

$$\begin{aligned} h(0,t) &= h_1 \\ h(l,t) &= h_2 \qquad \text{para } t > 0 \end{aligned}$$

la condición inicial es

$$h(x,0) = h_1 \qquad \text{para } 0 \leq x \leq l$$

Con estos datos y basándonos en el ejemplo anterior en la programación realizamos un nuevo script en FEniCS para la resolución del mismo; se exponen resultados más abajo.

Formulación variacional

Usando un esquema implícito en el tiempo tenemos:

$$\nabla^2 h^n = \frac{S}{T}\frac{h^n - h^{n-1}}{\Delta t}$$

entonces

$$\Delta t \cdot \left(\nabla^2 h^n\right) = \frac{S}{T}h^n - \frac{S}{T}h^{n-1}$$

Multiplicando por la función v e integrando en ambos lados de la ecuación

$$\begin{aligned} \int_\Omega \frac{S}{T}h^n v dx - dt\int_\Omega \left(\nabla^2 h^n\right) v dx &= \int_\Omega \frac{S}{T}h^{n-1} v dx \\ \frac{S}{T}\int_\Omega h^n v dx + dt\int_\Omega \nabla h^n \cdot \nabla v dx - dt\int_{\partial\Omega} \frac{\partial h}{\partial \mathrm{n}} \cdot v ds &= \frac{S}{T}\int_\Omega h^{n-1} v dx \end{aligned}$$

además, considerando que no existe flujo a través de las fronteras reescribimos la ecuación anterior como

$$\frac{S}{T}\int_{\Omega} h^n v dx + dt\int_{\Omega} \nabla h^n \cdot \nabla v dx = \frac{S}{T}\int_{\Omega} h^{n-1} v dx$$

en notación corta escribimos: dado h^{n-1} encontrar h^n tal que

$$a\left(h^n, v\right) = L\left(v\right) \qquad \forall v \in V$$

entonces

$$a\left(h^n, v\right) = \frac{S}{T}\int_{\Omega} h^n v dx + dt\int_{\Omega} \nabla h^n \cdot \nabla v dx$$

y

$$L\left(v\right) = \frac{S}{T}\int_{\Omega} h^{n-1} v dx$$

Solución analítica

La solución analítica de la ecuación de Laplace para las condiciones de frontera

$$\begin{aligned} h\left(0\right) &= h_1 \\ h\left(l\right) &= h_2 \end{aligned}$$

viene dada por la siguiente ecuación

$$h\left(x\right) = \left(\frac{h_2 - h_1}{l}\right) x + h_1$$

4.2. Problema con contaminante

Ahora vamos a considerar un acuífero (véase [14] y [24]), con las siguientes características para analizar: el dominio está delimitado por un cuadrado de lado l como se muestra en las figura 4.5.

Donde $l = 0.5$ mi (o 804.672 m) y c representa una fuente de contaminante. La fuente se encuentra ubicada del lado izquierdo en la figura, en la parte derecha la región está delimitada por un río.

Se considera al contaminante a modelar, como conservativo, es decir, no varía en su interacción con el medio y por tanto al atravesar el acuífero, mantiene todas sus propiedades durante todo el tiempo establecido.

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.

Las ecuaciones de flujo y transporte acopladas por la ley de Darcy, se usan para describir la evolución de la pluma de contaminante, las cuales se resuelven para la carga hidráulica h y las concentraciones del contaminante

$$\nabla \cdot (\mathbf{K} \cdot \nabla h) - S\frac{\partial h}{\partial t} + Q = 0 \tag{4.2}$$

$$\frac{\partial c}{\partial t} - \nabla \cdot (\mathbf{D} \cdot \nabla c - Vc) + Qc^w = 0 \tag{4.3}$$

y

$$V = -\frac{\mathbf{K}}{\phi} \cdot \nabla h \tag{4.4}$$

La ecuación (4.2) es la ecuación que utilizaremos para describir el flujo a través del acuífero para el estado estacionario, donde **K** es la conductividad hidráulica, Q es la fuerza de bombeo (fuente o sumidero) y S el coeficiente de almacenamiento; la ecuación (4.3) es la ecuación de transporte de contaminantes, que describe los cambios en la concentración para un soluto conservador c a través el tiempo, con **D** como la dispersión hidrodinámica y c^w es la concentración de fluido bombeado. Por último, en la ecuación (4.4) mostramos la ley de Darcy que calcula la velocidad V del agua en los poros del acuífero con ϕ como la porosidad efectiva.

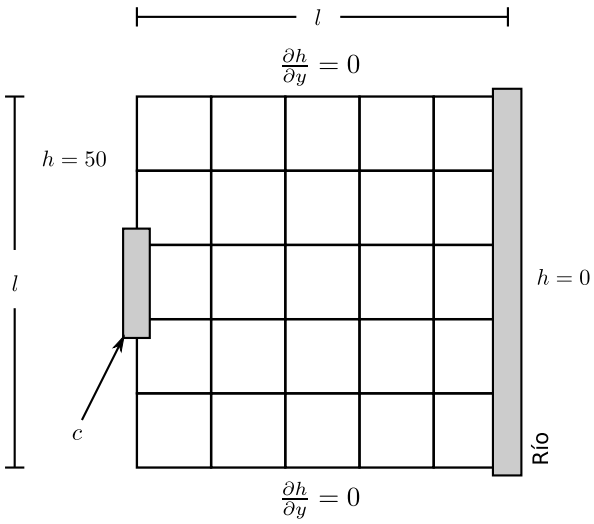
En la figura (4.5) (a) se muestran las condiciones de frontera para el flujo y en (b) para el transporte. Las concentraciones están dadas en partes por millón (ppm) y las cargas hidráulicas en metros.

Se utilizan 48 pasos de tiempo para simular un periodo de dos años. Para la parte del flujo tenemos un valor en la frontera del lado izquierdo de $h = 50$ m y para todos los nodos en la frontera derecha con un valor de $h = 0$ m.

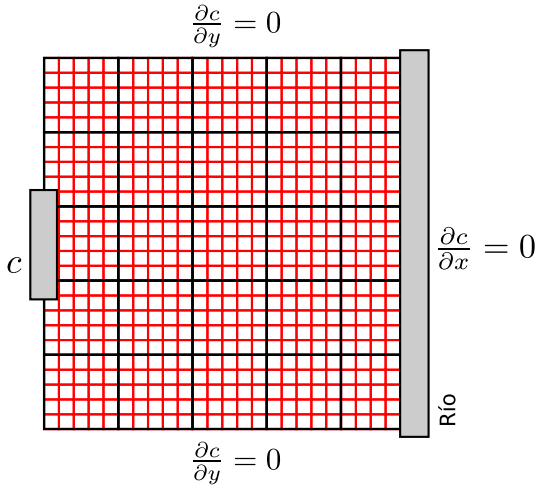
La fuente del contaminante esta activa durante todo este periodo, con una concentración constante de $c = 50$ ppm. En el modelo de transporte los nodos que no son parte de la fuente del contaminante se les asigna una condición de segundo tipo (Neumann) igualada a cero.

A la capa acuífera se le asigna un espesor de 55 m, una conductividad hidráulica en las tres direcciones (x, y, z) de 30, una carga hidráulica inicial de 70, una porosidad de $\phi = 0.25$, un coeficiente de almacenamiento de $S = 0.001$, una dispersividad de $\{33, 3.3, 1.0\}$ en las direcciones x, y, z respectivamente.

Cada período de esfuerzo (stress) tiene un paso de tiempo de 15.2083 días, estos períodos de esfuerzos se utilizan tanto para el modelo de flujo como para el modelo de transporte.



(a) Condiciones de frontera para el modelo de flujo



(b) Condiciones de frontera para el modelo de transporte

Figura 4.5: Ejemplo de acuífero con contaminante

4.2.1. Formulación variacional para flujo

Usando un esquema implícito en el tiempo para la ecuación (4.2), tenemos

$$\begin{aligned} \nabla \cdot (\mathbf{K} \cdot \nabla h^n) + Q &= S\frac{h^n - h^{n-1}}{\Delta t} \\ \Delta t \cdot \nabla \cdot (\mathbf{K} \cdot \nabla h^n) + Q\Delta t &= Sh^n - Sh^{n-1} \end{aligned}$$

Multiplicando lo anterior por una función v e integrando por partes:

$$\begin{aligned} \int_\Omega Sh^n v dx - dt \int_\Omega \nabla \cdot (\mathbf{K} \cdot \nabla h^n)\, v dx - dt \int_\Omega Q v dx &= \int_\Omega Sh^{n-1} v dx \\ S\int_\Omega h^n v dx + dt \int_\Omega (\mathbf{K} \cdot \nabla h^n) \cdot \nabla v dx - dt \int_{\partial\Omega} \mathbf{K}\frac{\partial h^n}{\partial \mathrm{n}} v ds - Qdt \int_\Omega v dx & \\ = S\int_\Omega h^{n-1} v dx & \end{aligned}$$

considerando que no hay flujo por las fronteras, la última ecuación queda

$$S\int_\Omega h^n v dx + dt \int_\Omega (\mathbf{K} \cdot \nabla h^n) \cdot \nabla v dx - Qdt \int_\Omega v dx = S\int_\Omega h^{n-1} v dx$$

y en una notación corta escribimos: dado $h^{n-1} \in V$ encontrar $h^n \in V\ \forall v \in \hat{V}$ y $n = 1, 2, \ldots$ tal que

$$a\left(h^n, v\right) = L\left(v\right)$$

entonces

$$a\left(h^n, v\right) = S\int_\Omega h^n v dx + dt \int_\Omega (\mathbf{K} \cdot \nabla h^n) \cdot \nabla v dx - Qdt \int_\Omega v dx$$

y

$$L\left(v\right) = S\int_\Omega h^{n-1} v dx$$

4.2.2. Formulación variacional para transporte

Ahora haremos la formulación variacional de la ecuación de transporte de contaminantes (4.3). Comenzamos pasando a un esquema implícito de la siguiente manera

$$\begin{aligned} \nabla \cdot (\mathbf{D} \cdot \nabla c^n - Vc^n) + Qc^w &= \frac{c^n - c^{n-1}}{\Delta t} \\ \Delta t \cdot \nabla \cdot (\mathbf{D} \cdot \nabla c^n - Vc^n) + Qc^w \Delta t &= c^n - c^{n-1} \end{aligned}$$

ahora multiplicando por v e integrando por partes lo anterior, tenemos

$$\int_\Omega c^n v dx + dt \int_\Omega \nabla \cdot (V c^n - \mathbf{D} \cdot \nabla c^n) \, v dx - dt \int_\Omega Q c^w v dx = \int_\Omega c^{n-1} v dx$$

$$\int_\Omega c^n v dx + dt \int_\Omega \nabla \cdot (V c^n) \, v dx - dt \int_\Omega \nabla \cdot (\mathbf{D} \cdot \nabla c^n) \, v dx - Q c^w dt \int_\Omega v dx = \int_\Omega c^{n-1} v dx$$

$$\int_\Omega c^n v dx + dt \int_\Omega \nabla \cdot (V c^n) \, v dx + dt \int_\Omega \mathbf{D} \nabla c^n \cdot \nabla v dx - dt \int_{\partial\Omega} \mathbf{D} \frac{\partial c^n}{\partial \mathrm{n}} v ds - Q c^w dt \int_\Omega v dx = \int_\Omega c^{n-1} v dx$$

Considerando el comportamiento en la frontera, la última ecuación queda

$$\int_\Omega c^n v dx + dt \int_\Omega \nabla \cdot (V c^n) \, v dx + dt \int_\Omega \mathbf{D} \nabla c^n \cdot \nabla v dx - Q c^w dt \int_\Omega v dx = \int_\Omega c^{n-1} v dx$$

y en una notación corta escribimos: dado $c^{n-1} \in V$ encontrar $c^n \in V \quad \forall v \in \hat{V}$ y $n = 1, 2, \ldots$ tal que

$$a\left(c^n, v\right) = L\left(v\right)$$

entonces

$$a\left(c^n, v\right) = \int_\Omega c^n v dx + dt \int_\Omega \nabla \cdot (V c^n) \, v dx + dt \int_\Omega \mathbf{D} \nabla c^n \cdot \nabla v dx - Q c^w dt \int_\Omega v dx$$

y aplicando propiedades de la divergencia llegamos a

$$a\left(c^n, v\right) = \int_\Omega c^n v dx + dt \int_\Omega c^n \left(\nabla \cdot V\right) v dx + dt \int_\Omega \left(\nabla c^n \cdot V\right) v dx + dt \int_\Omega \mathbf{D} \nabla c^n \cdot \nabla v dx - Q c^w dt \int_\Omega v dx$$

y

$$L\left(v\right) = \int_\Omega c^{n-1} v dx$$

4.3. Resumen

En este capítulo integramos los conceptos de la parte hidrológica. Se presentan algunos casos de estudio, los cuales se resuelven empleando los métodos de elemento finito y volumen finito (expuestos en el capítulo anterior).

Se describe cada problema, identificando las ecuaciones diferenciales parciales que intervienen, sus condiciones iniciales así como las condiciones de frontera y se presentan pequeños esquemas de cada uno de los problemas.

Además se hace la formulación variacional para cada problema, lo cual nos ayudará, posteriormente, al desarrollo del código apropiado para la solución numérica de cada caso.

Y al final del capítulo mostramos el problema a resolver: un problema de flujo y transporte de contaminantes en agua subterránea. También se presenta el desarrollo de la formulación axiomática para las dos ecuaciones.

5 Resultados experimentales

En el capítulo 4 hemos revisado algunos casos de estudio y un problema con contaminantes, ahora retomaremos esos problemas para mostrar sus resultados después de la implementación mediante FEniCS y TUNAM, así como los resultados de las ejecuciones en paralelo.

5.1. Problemas sin contaminantes

5.1.1. Problema 1

Implementación

A continuación se escribe el código usado en FEniCS

```
"""
Ejemplo 1. Tesis
Regional GroundWater Flow System
Toth (1962)
Autor: Amed Leones Viloria
toth1.py
"""

#importar las librerias de dolfin

from dolfin import *
from cmath import cos,cosh,sqrt,pi

#Creacion de la malla

malla = Rectangle(0,0,100,100,100,100)
```

```
17
18  #espacio de funciones
19  V = FunctionSpace(malla, 'CG', 1)
20  u = TrialFunction(V)
21  v = TestFunction(V)
22
23  #Solucion analitica
24  def h(x0, y0):
25          h1 = 100
26          c = 0.02
27          s = 100
28          suma = 0
29          if y0 < 1e-15:
30                  suma = h1
31          else:
32                  for n in range(1, 100):
33                          suma = suma + (cos((2*n+1)*pi*
                                x0/s)*cosh((2*n+1)*pi*y0/s
                                ))/(((2*n+1)**2)*cosh((2*n
                                +1)*pi*h1/s))
34                  suma = suma*((-4*c*s)/(pi**2))+h1+((c*
                        s)/2)
35          return suma.real
36
37  #graficar solucion analitica
38  xy = malla.coordinates() #extrae coordenadas de los
        nodos
39  ua = Function(V)
40
41  for i in range(len(ua.vector().array())):
42          x0 = xy[i][0]
43          y0 = xy[i][1]
44          ua.vector()[i] = h(x0, y0)
45
46  plot(ua)
47  #definicion de condicion de frontera
48  u0 = Expression('0.02*x[0]+100')
49
50  def frontera(x, on_boundary):
51          return on_boundary
52
53  def on_boundary(x):
54          return abs(x[1]-100.) < 1E-15
55
56  bc = DirichletBC(V, u0, on_boundary)
57  #frontera)
58
59  f = Constant(0.0)
60
61  #forma bilineal, forma lineal
```

```
a = inner(grad(u), grad(v))*dx
L = f*v*dx

#Calcular la solucion
u = Function(V)
solve(a == L, u, bc)

#Graficar solucion y la malla
plot(u)
#plot(malla)

#archivos con valores
file = File('toth.pvd')
file << u

#calculo del error
error = 0
for i in range(len(ua.vector().array())):
        x0 = xy[i][0]
        y0 = xy[i][1]
        error = error + (u.vector()[i] - h(x0, y0)) **
            2
error = sqrt(error) / len(ua.vector().array())

print 'El error es: ',error

#grafico del error en cada punto de la malla abs(u-ua)
uerror = Function(V)
for i in range(len(ua.vector().array())):
        x0 = xy[i][0]
        y0 = xy[i][1]
        uerror.vector()[i] = abs(u.vector()[i] - h(x0,
            y0))

plot(uerror)
#Mantener la grafica
interactive()
```

Los resultados para el problema uno, con un core se representan en la figura 5.1 En la figura 5.2 puede verse la misma solución para el problema 1, ahora empleando 2 cores.

Cálculo del error

Para el cálculo del error usamos la ecuación

$$Error = \frac{1}{NI \cdot NJ} \sum_{i=1}^{NI} \sum_{j=1}^{NJ} \left[T_n\left(i,j\right) - T\left(i,j\right)\right]^2 \tag{5.1}$$

donde NI y NJ son el número de nodos para la malla en las direcciones x y y, respectivamente; T_n es la solución calculada y T es la solución analítica. Y para el cálculo

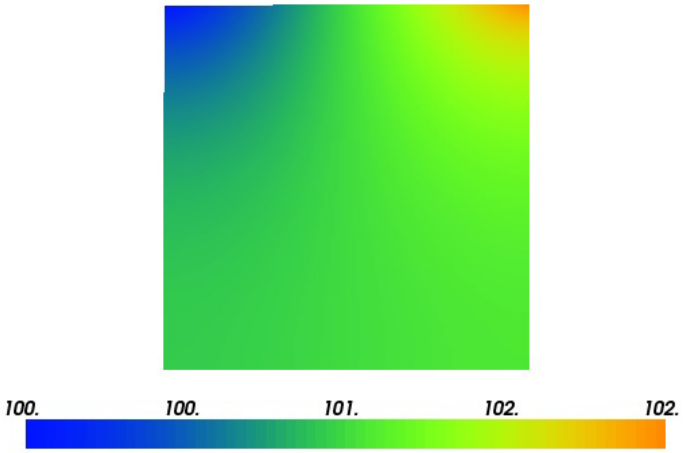


Figura 5.1: Solución Problema 1. 1 Core

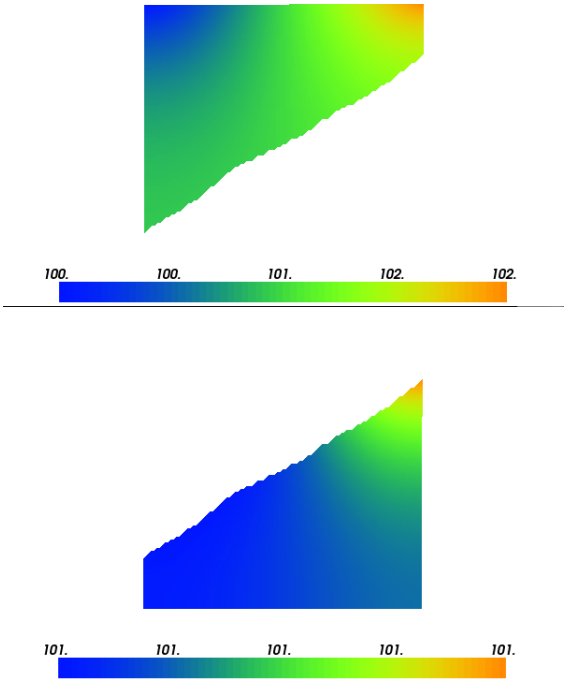


Figura 5.2: Solución Problema 1. 2 Core

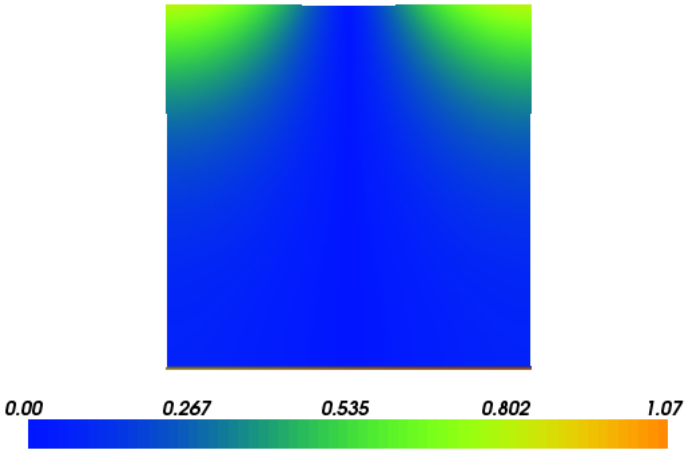


Figura 5.3: Error en la malla $|u - u_a|$, problema 1

NI	NJ	$Error$
40	40	0.00698457537279
80	80	0.00321385080064
160	160	0.00152860003148
320	320	0.000743411024384

Cuadro 5.1: Errores en el Problema 1

del error en la malla, tenemos

$$Error_{malla} = |T_n(i,j) - T(i,j)| \tag{5.2}$$

La figura 5.3 muestra la distribución del error en la malla. Con la ecuación (5.1) formamos el cuadro 5.1 para diferentes tamaños de la malla y se calcula el error.

5.1.2. Problema 2

Implementación

La implementación usando FEniCS para este problema es

```
"""
Ejemplo 2. Tesis
Regional GroundWater Flow System
Toth (1963)
Autor: Amed Leones Viloria
toth2.py
```

```
"""

#importar las librerias de dolfin

from dolfin import *
#from math import *

#Creacion de la malla
malla = Rectangle(0,0,100,100,100,100)

#espacio de funciones
V = FunctionSpace(malla, 'CG', 1)

u = TrialFunction(V)
v = TestFunction(V)

#definicion de condicion de frontera

u0 = Expression('100 + x[0]*tan(0.019199)+2*(sin(2*pi*x[0]/80.)
    /cos(0.019199))')

def frontera(x, on_boundary):
        return on_boundary

def on_boundary(x):
        return abs(x[1]-100.) < 1E-15

bc = DirichletBC(V, u0, on_boundary)
#frontera)

f = Constant(0.0)

#forma bilineal, forma lineal
a = inner(nabla_grad(u), nabla_grad(v))*dx

L = f*v*dx

#Calcular la solucion
u = Function(V)
solve(a == L, u, bc)

#Graficar solucion y la malla
plot(u)
plot(malla)

#archivos con valores
file = File('toth2.pvd')
file << u

#Mantener la grafica
```

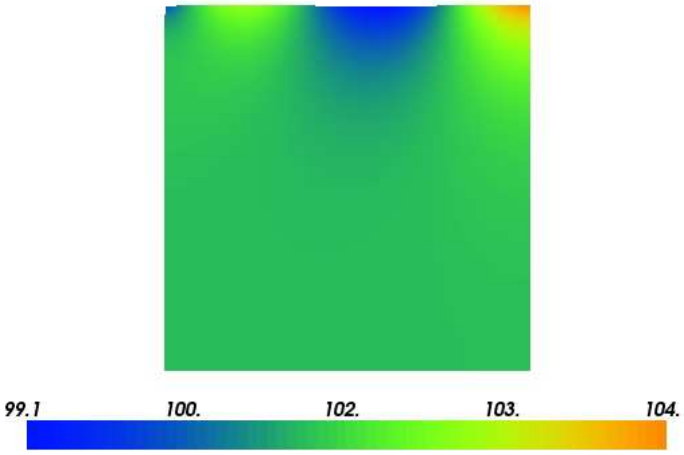


Figura 5.4: Solución Problema 2

```
56 interactive()
```

En la figura 5.4 se muestran los resultados para el problema sinusoidal.

En la figura 5.5 puede verse la misma solución para el problema 2, empleando dos cores para el cómputo.

5.1.3. Problema 3

Implementación

El siguiente código es la implementación del tercer problema en FEniCS

```
1  """
2  Ejemplo 3. Tesis
3  The reservoir example
4  Autor: Amed Leones Viloria
5  problema3.py
6  """
7
8  from dolfin import *
9  from cmath import sqrt
10
11 #Creacion de la malla
12
```

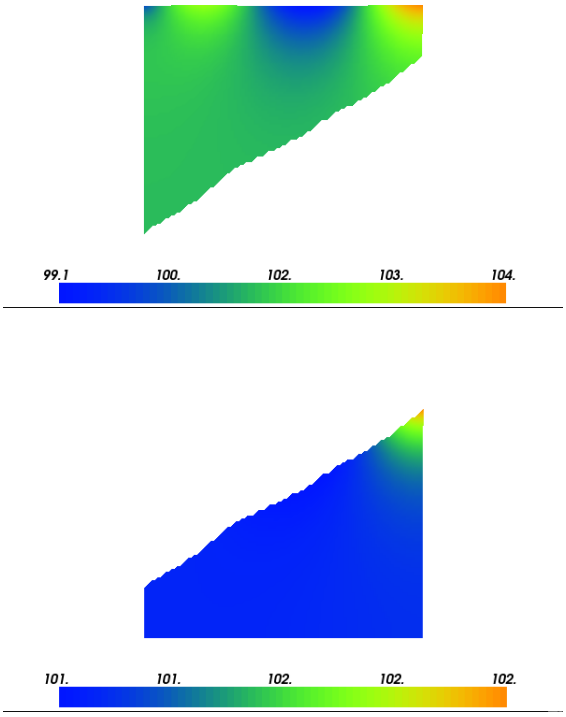


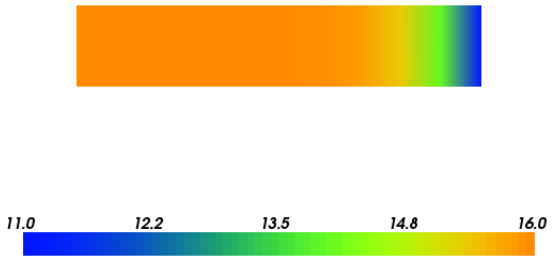
Figura 5.5: Solución Problema 2. 2 Core

```
malla = Rectangle(0,0,100,20,80,160)

#espacio de funciones
V = FunctionSpace(malla, 'CG', 1)

#Parametros constantes
T = 0.02
S = 0.002

#definicion de solucion analitica
def ha(x0):
        h1 = 16
        h2 = 11
        l = 100
        if x0 < 0:
                s = h2
        else:
                s = (h2 - h1) * (100 ** -1) * x0 + h1
        return s

#graficar la funcion analitica
xy = malla.coordinates()
ua = Function(V)

for i in range(len(ua.vector().array())):
        x0 = xy[i][0]
        ua.vector()[i] = ha(x0)

plot(ua)

#definicion de la condiciones de frontera
tol = 1E-15
def on_boundary_L(x):
        return abs(x[0]) < tol
hL = Constant(16)
bcL = DirichletBC(V, hL, on_boundary_L)

def on_boundary_R(x):
        return abs(x[0] - 100.0) < tol
hR = Constant(11)
bcR = DirichletBC(V, hR, on_boundary_R)

#Condiciones iniciales
h_0 = Constant(16)
h0 = interpolate(h_0, V)

#Problema variacional
#paso del tiempo
dt = 5#10
h = TrialFunction(V)
```

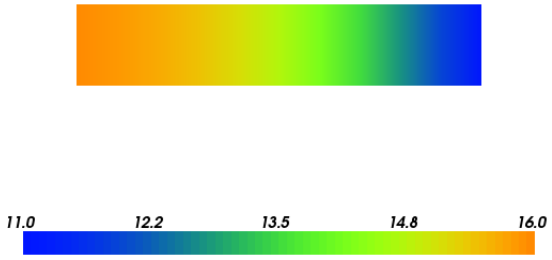
```
v = TestFunction(V)
a = (S / T) * h * v * dx + dt * inner(grad(h), grad(v)) * dx
L = (S / T) * h0 * v * dx

#Solucion del problema variacional
#tiempo maximo
tmax = 400
#tiempo actual
t = dt
#guarda la solucion actual
h = Function(V)
#se ensambla la matriz se usa solo una vez
A = assemble(a)
#ciclo de tiempo
while t <= tmax:
        #se ensambla el lado derecho
        b = assemble(L)
        #se aplica bc
        bcL.apply(A, b)
        bcR.apply(A, b)
        #se resuelve el sistema de ecuaciones
        solve(A, h.vector(), b)
        #se actualiza la solucion anterior
        h0.assign(h)
        t += dt
        plot(h)
#se grafica la solucion actual

#calculo de error
error=0
for i in range (len(ua.vector().array())):
   x0=xy[i][0]
   error=error+(h.vector()[i]-ha(x0))**2
error=sqrt(error)/len(ua.vector().array())
print "El error es: ", error

uerror = Function(V)
for i in range (len(ua.vector().array())):
   x0=xy[i][0]
   uerror.vector()[i]=abs(h.vector()[i]-ha(x0))

plot(uerror)
interactive()
```

En la figura 5.6 se muestran los resultados de comparar el tiempo máximo en que se alcanza la solución del problema.

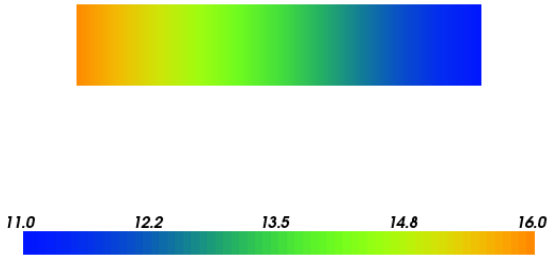
Puede notarse que para 10 min es una mala aproximación a la solución analítica que se encuentra a los 400 min.



(a) 10 min



(b) 100 min



(c) 400 min

Figura 5.6: Gráfica que muestra la salida del problema 3

NI	NJ	$Error$
10	20	0.00292644914652
20	40	0.00157835468951
40	80	0.00081687953153
80	160	0.000415206488984

Cuadro 5.2: Errores en el problema 3

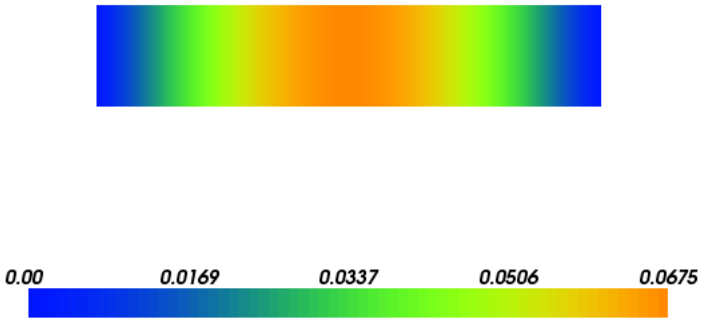


Figura 5.7: Error en la malla $|u - u_a|$, problema 3

Cálculo del error

En el cuadro 5.2, vemos el error calculado en base a la ecuación (5.1) para este problema y el error en la malla en la figura 5.7.

5.2. Problema con contaminante

5.2.1. Implementación para Flujo

A continuación se escribe un fragmento de código del cálculo de la parte de flujo

```
"""
Ejemplo 4b. Tesis MMC
The general flow and the conservative
convectiondispersion transport equations,
coupled through Darcy's Law, are used
to describe the contaminant plume evolution
Autor: Amed A. Leones Viloria
```

```
problema4b.py
"""

#Importar los modulos necesarios

from dolfin import *

#Definicion de parametros constantes
S = 0.001
Q = 0
phi = 0.25
#Tensores
#K = 30,30,30
#D = 33,3.3,1
K = Expression((('30.', '0.'),
                           ('0.', '30.')))

D = Expression((('33.', '0'),
                ('0', '1.')))
###

#Definicion de dominio y malla
mesh = Rectangle(0, 0, 804.672, 804.672, 40, 40)
###

#Definicion del espacio de funciones
#CG: Continuos Galerkin
#1: Linear Triangle
V = FunctionSpace(mesh, 'CG', 1)
W = FunctionSpace(mesh, 'CG', 1)
###

###Flujo
#Condiciones de frontera
# = 1e-15
def on_boundary_L(x):
        return abs(x[0]) < DOLFIN_EPS
hL = Constant(50)
fbcL = DirichletBC(V, hL, on_boundary_L)

def on_boundary_R(x):
        return abs(x[0] - 804.672) < DOLFIN_EPS
hR = Constant(0)
fbcR = DirichletBC(V, hR, on_boundary_R)

fbc = [fbcL, fbcR]
###

#Condiciones Iniciales
h_0 = Constant(70)
```

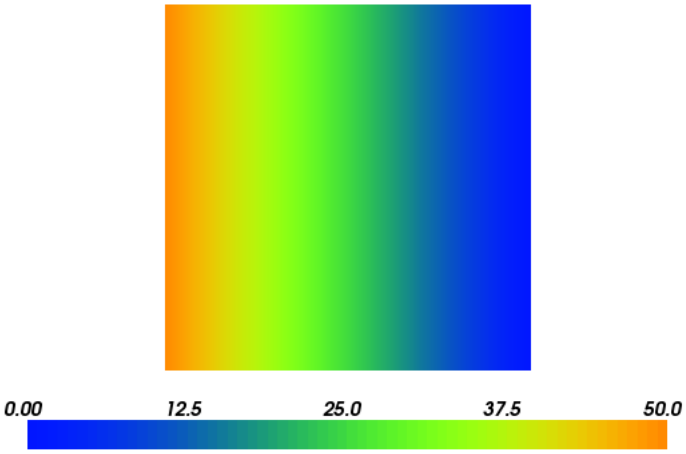


Figura 5.8: Solución flujo, problema del contaminante

```
h0 = interpolate(h_0, V)
###

#Problema Variacional
#Flujo
#paso del tiempo
dt = 152.083#15.2083

v = TestFunction(V)
h = TrialFunction(V)
af = S * h * v * dx + dt * inner(K * grad(h), grad(v)) * dx - Q
     * dt * v * dx
Lf = S * h0 * v * dx
###
```

En la figura 5.8 se muestra el cómputo de la carga hidráulica.

5.2.2. Implementación para transporte

Con FEniCS

En esta parte tomaremos en cuenta el código recién hecho para el problema de flujo, ya que hace el cálculo de la ecuación de velocidad y se adiciona la formulación variacional del transporte de contaminantes.

```
###Transporte
#Condiciones de frontera
```

```
cmin = 804.672 * 1.5 / 4.0
cmax = 804.672 * 2.5 / 4.0
def on_boundary_L2(x):
        return (abs(x[0]) < DOLFIN_EPS and x[1] > cmin and x[1]
             < cmax)
cL2 = Constant(50)
def on_boundary_L1(x):
        return (abs(x[0]) < DOLFIN_EPS and x[1] > cmax and x[1]
             < 804)
cL1 = Constant(0)
def on_boundary_L3(x):
        return (abs(x[0]) < DOLFIN_EPS and x[1] > DOLFIN_EPS
            and x[1] < cmin)
cL3 = Constant(0)

cbcL1 = DirichletBC(W, cL1, on_boundary_L)
cbcL2 = DirichletBC(W, cL2, on_boundary_L2)
cbcL3 = DirichletBC(W, cL3, on_boundary_L3)
cbc = [cbcL1, cbcL2, cbcL3]
###

#condiciones iniciales
c_0 = Constant(0)
c0 = interpolate(c_0, W)
###

#Problema Variacional
c_w = 0
VV = VectorFunctionSpace(mesh, 'CG', 1)
velocity = Function(VV)
vc = TestFunction(W)
c = TrialFunction(W)
ac = c * vc * dx + dt * c * div(velocity) * vc * dx \
        + dt * inner(D*grad(c), grad(vc)) * dx - Q * c_w * dt *
             vc * dx
Lc = c0 * vc * dx
###

#Solucion del problema flujo
T = 48 * dt
t = dt
h = Function(V)
Af = assemble(af)

#Solucion del problema concentracion
t = dt
c = Function(W)
Ac = assemble(ac)
while t <= T:
        bf = assemble(Lf)
```

```
49          for condition in fbc:
50                  condition.apply(Af, bf)
51          solve(Af, h.vector(), bf)#, 'gmres')
52          h0.assign(h)
53          velocity = - project((K / phi) * grad(h), VV)
54          Ac = assemble(ac)
55          #print t
56          bcc = assemble(Lc)
57          for condition in cbc:
58                  condition.apply(Ac, bcc)
59          solve(Ac, c.vector(), bcc)#, 'gmres')
60          c0.assign(c)
61          t += dt
62          plot(c)
63          #print t
64  plot(h)
65  plot(c)
66  plot(velocity)
67
68  interactive()
69  ###
```

Con TUNAM

La solución del problema con contaminante usando TUNAM requiere los siguientes pasos:

1. Construcción de la malla:

```
1  StructuredMesh<Uniform<double, 2> > mesh(length_x,
       num_nodes_x,length_y, num_nodes_y);
2  double dx = mesh.getDelta(X);
3  double dy = mesh.getDelta(Y);
4  int num_vols_x = mesh.getExtentVolumes(X);
5  int num_vols_y = mesh.getExtentVolumes(Y);
6  mesh.print();
```

   La línea 1 define una malla estructurada en dos dimensiones, de longitudes `length_x` y `length_y`. El número de nodos se especifica con `num_nodes_x` y `num_nodes_y`. El objeto `mesh` representa la malla del dominio y puede calcular el tamaño de las celdas en cada dirección (líneas 2 y 3). También se puede obtener el número de volúmenes de la malla, líneas 4 y 5. Finalmente se puede imprimir información acerca de la malla construida.

2. Definición de los campos escalares sobre la malla:

```
ScalarField2D h  ( mesh.getExtentVolumes() );
ScalarField2D c  ( mesh.getExtentVolumes() );
ScalarField2D h_n( mesh.getExtentNodes()   );
ScalarField2D c_n( mesh.getExtentNodes()   );
ScalarField2D us(num_nodes_x, num_vols_y); // staggered u-
    velocity
ScalarField2D vs(num_vols_x, num_nodes_y); // staggered v-
    velocity
ScalarField2D un(mesh.getExtentNodes());
ScalarField2D vn(mesh.getExtentNodes());
```

Lo anterior define varios objetos para almacenar la solución de la carga hidráulica y la concentración. Ademas se definen campos para la velocidad. Todo esto se hace tanto en los nodos como en los centros de los volúmenes.

3. Definición del sistema de ecuaciones discreto:

```
SparseMatrix< Diagonal< double, 2> > A(num_nodes_x,
    num_nodes_y);
ScalarField2D                        b(num_nodes_x,
    num_nodes_y);
```

4. Se definen las ecuaciones a resolver:

```
ScalarEquation<CTCH1<double,2> > carga(h, A, b, mesh.
    getDeltas());
carga.setDeltaTime(dt);
carga.setGamma(permeability);
carga.setDirichlet   (LEFT_WALL);
carga.setDirichlet   (RIGHT_WALL);
carga.setNeumann     (TOP_WALL);
carga.setNeumann     (BOTTOM_WALL);
carga.print();

ScalarEquation<CTCO1<double,2> > conc(c, A, b, mesh.
    getDeltas());
conc.setDeltaTime(dt);
conc.setGamma(D);
conc.setDirichlet   (LEFT_WALL);
conc.setNeumann     (RIGHT_WALL);
conc.setNeumann     (TOP_WALL);
conc.setNeumann     (BOTTOM_WALL);
conc.setUvelocity(us);
conc.setVvelocity(vs);
conc.print();
```

5. Finalmente, se resuelven las ecuaciones:

```
36  while (t <= Tmax) {
37        carga.calcCoefficients();
38        iterH = Solver::TDMA2DY(carga, tolerance, tdma_iter_h
              , 1.0);
39        error_h = carga.calcErrorL2();
40        res_h = carga.calcResidual();
41        carga.update();
42
43      // Calculate the velocity
44        for(int i = bi; i < ei; i++)
45               for(int j = bj; j <= ej; j++)
46                 us(i,j) = -coef * (h(i+1,j) - h(i,j) ) / dx;
47
48        for(int i = bi; i <= ei; i++)
49               for(int j = bj; j < ej; j++)
50                 vs(i,j) = -coef * (h(i,j+1) - h(i,j) ) / dy;
51
52        conc.calcCoefficients();
53        iterC = Solver::TDMA2DY(conc, tolerance, tdma_iter_c,
               1.0);
54        error_c = conc.calcErrorL2();
55        res_c = conc.calcResidual();
56        conc.update();
57        ++iteration;
58
59        t += dt;
60  }
```

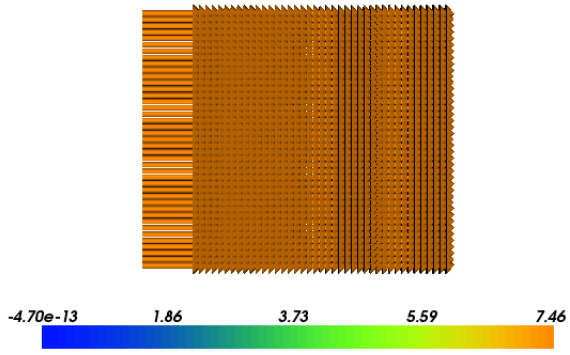
En este ejemplo, primero se calculan los coeficientes de la ecuación; posteriormente se resuelve el sistema; se calculan algunas cantidades, como el error y el residuo; y finalmente se actualiza la solución. Nótese que se calcula la velocidad justo después de haber calculado la carga hidráulica. Esto se hace usando diferencias centradas.

Para el cálculo de la concentración en la ecuación de contaminantes se involucra la velocidad, la cual se muestra en las figura 5.9. Y las soluciones para el contaminante, tanto con FEniCS (a) como para TUNAM (b), se muestran en la figura 5.10.

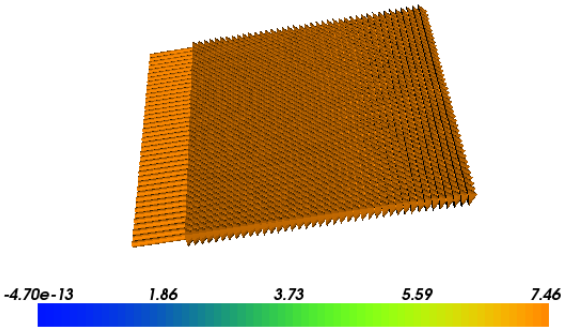
Los datos medidos se hicieron en el clúster *Olintlali* adquirido por el Instituto de Geofísica de la UNAM en 2011, en el que principalmente se aplican técnicas de alto desempeño en la solución computacional de los modelos desarrollados y aplicados a diferentes problemas de interés nacional, en especial la recuperación mejorada de hidrocarburos.

El clúster posee las siguientes características técnicas[1]:

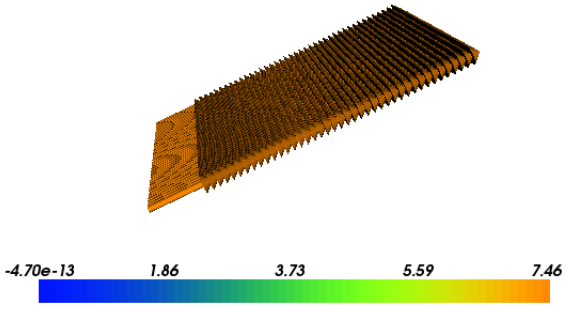
[1] Para más información revisar el sitio http://mmc3.geofisica.unam.mx/olintlali/bin/view/Olintlali/OlintlaliEquipo



(a)

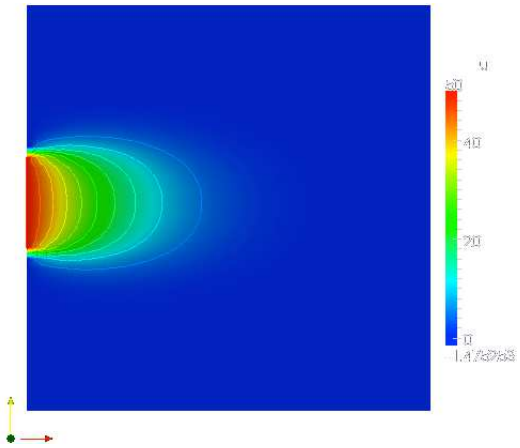


(b)

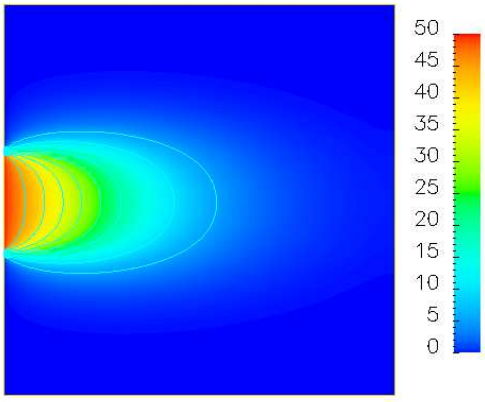


(c)

Figura 5.9: Campo de velocidades



(a) Solución con FEniCS



(b) Solución con TUNAM

Figura 5.10: Solución al transporte, problema del contaminante

- Información del sistema
  Sistema Operativo ROCKs 5.4
  SRMS Torque 2.4
  Nodos 8+1
  Procesadores 36
  Núcleos 216
  Hilos 432
  GPGPU 2
  Memoria 432 GB
  Interconexión Gigabit Ethernet
  Almacenamiento 10 TB (RAID 5)
- Descripción física
  Nodo maestro
  Marca Supermicro
  - Procesador
    4x Intel(R) Xeon(R) CPU X5650 @ 2.67GHz
    4x6 Núcleos
    4x12 Hilos de ejecución
  - Memoria
    Hyundai DIMM 1333 MHz (0.8 ns)
    48 GB
  - Tarjeta madre
    Supermicro X8DT3
    Intel® 5500 (Tylersburg) Chipset
    Hasta 192GB DDR3 1333/ 1066/ 800MHz
    Intel® 82574-L Gigabit Ethernet
    SAS Controller
    6x SATA2 (3 Gbps)
  - Almacenamiento
    - HD SATA FUJITSU MHZ2250B
      250GB
    - Arreglo RAID
      MegaRAID SAS 1078
      7995 GB
      Nivel 5

Nodos de cómputo (8 nodos)
Marca Supermicro

- Procesador
  4x Intel(R) Xeon(R) CPU X5650 @ 2.67GHz
  4x6 Núcleos
  4x12 Hilos de ejecución
- Memoria
  Hyundai DIMM 1333 MHz (0.8 ns)
  48 GB
- Tarjeta madre
  Supermicro X8DTL
  Intel® 5520 (Tylersburg) Chipset
  Hasta 96GB DDR3 1333/ 1066/ 800MHz
  Intel® 82576 Dual-Port Gigabit Ethernet
  SAS Controller
  x SATA2 (3 Gbps)
- Almacenamiento
  HD Seagate ST3500514NS
  500GB
- Unidades de procesamiento gráfico (1 nodo)
  - 2 Procesadores GPU Tesla M2050
    515 GFlops
    148 GB/sec memory bandwidth
    448 CUDA cores *

Aplicando las ecuaciones (2.49) y (2.50) formamos los cuadros 5.3, 5.4, 5.5, 5.6 y 5.11 para mallas de 512×512, 640×640, 768×768, 1024×1024 y 2048×2048 nodos, respectivamente.

En la figuras 5.12, 5.13 y 5.14 se grafica el tiempo de ejecución, la aceleración y la eficiencia, respectivamente, contra el número de procesadores empleados para diferentes mallas del problema.

En estas tres gráficas, se observa que el programa es eficiente solo hasta 8 procesadores. Después de ese número, el tiempo de ejecución se incrementa, con la consecuencia de una mala eficiencia. La razón de lo anterior es debida a que el clúster donde se ejecutan estos programas, tiene una conectividad de baja velocidad. Cada nodo del clúster tiene a lo más 12 procesadores. Cuando un problema requiere más de ese número, es necesario intercambiar información con otros nodos. Esta comunicación es lenta

p	T_p (seconds)	S_p	E_p
1	496	1	1
2	404	1.227722772	0.613861386
4	305	1.626229508	0.406557377
8	266	1.864661654	0.233082707
16	307	1.615635179	0.100977199
32	364	1.362637363	0.042582418
64	349	1.421203438	0.022206304

Cuadro 5.3: Tiempo, speedup y eficiencia en malla de 512×512

p	T_p (seconds)	S_p	E_p
1	967	1	1
2	657	1.471841705	0.735920852
4	494	1.957489879	0.48937247
8	416	2.324519231	0.290564904
16	444	2.177927928	0.136120496
32	526	1.838403042	0.057450095
64	690	1.401449275	0.021897645

Cuadro 5.4: Tiempo, speedup y eficiencia en malla de 640×640

p	T_p (seconds)	S_p	E_p
1	1621	1	1
2	1079	1.50231696	0.75115848
4	755	2.147019868	0.536754967
8	661	2.452344932	0.306543117
16	695	2.332374101	0.145773381
32	792	2.046717172	0.063959912
64	1078	1.503710575	0.023495478

Cuadro 5.5: Tiempo, speedup y eficiencia en malla de 768×768

p	T_p (seconds)	S_p	E_p
1	3941	1	1
2	2309	1.70679948	0.85339974
4	1611	2.446306642	0.611576661
8	1323	2.978835979	0.372354497
16	1254	3.142743222	0.196421451
32	1412	2.791076487	0.08722114
64	1642	2.400121803	0.037501903

Cuadro 5.6: Tiempo, speedup y eficiencia en malla de 1024×1024

p	T_p (seconds)	S_p	E_p
1	28321	1	1
2	15378	1.841656913	0.920828456
4	9903	2.859840452	0.714960113
8	7621	3.716178979	0.464522372
16	6546	4.326458906	0.270403682
32	6333	4.471972209	0.139749132
64	6875	4.119418182	0.064365909

Figura 5.11: Tiempo, speedup y eficiencia en malla de 2048×2048

y ocasiona un bajo rendimiento. Lo anterior se demuestra cuando elevamos el número de grados de libertad del problema. Se observa que conforme este número aumenta, la aceleración y la eficiencia mejoran. Esto es debido a que, cuando se tienen muchos nodos, cada procesador requiere de mayor trabajo numérico, el cual es comparable con el tiempo de comunicación. Por lo tanto, entre más grande sea la malla, mejor eficiencia se tendrá en la ejecución en paralelo.

5.3. Resumen

Este capítulo aborda los resultados obtenidos de los problemas de los casos de estudios planteados en el capítulo previo.

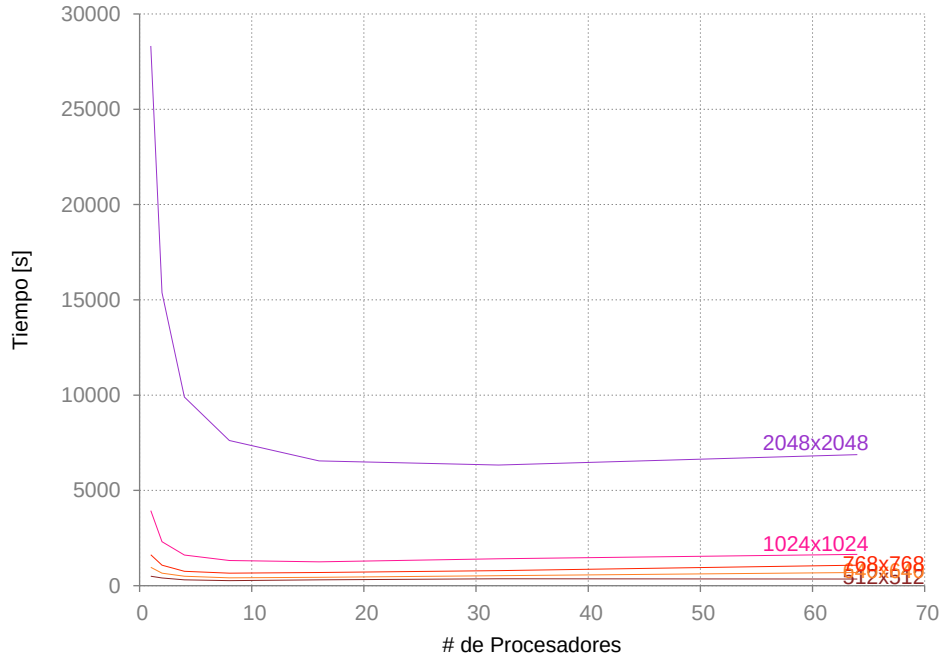
Se presenta el código fuente (propio) para calcular la solución de cada problema, además de los cálculos de los errores.

Las implementaciones que se muestran se realizan con el software FEniCS y para nuestro problema de contaminantes también se resuelve utilizando TUNAM.

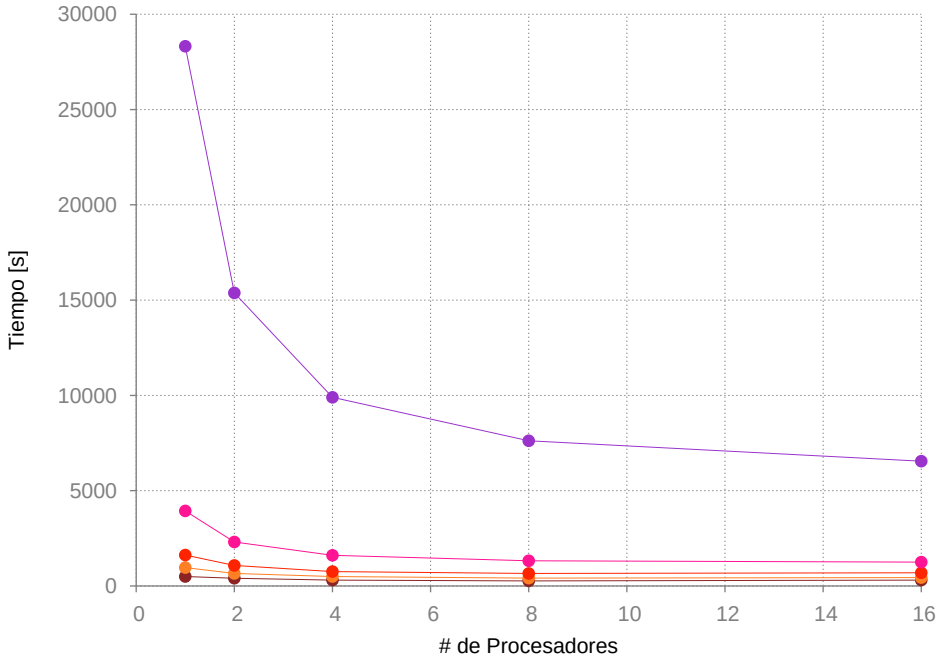
Se muestra las corridas en paralelo que se hicieron en el clúster de adquirido por el Instituto de Geofísica de la UNAM, así como las especificaciones técnicas de éste.

Se citan los parámetros a evaluar (aceleración y eficiencia) dentro de cada una de las ejecuciones que se midieron, para después comparar estos datos.

Se tabulan las corridas de hasta 64 procesadores y se grafican las mediciones de tiempo de procesamiento contra el número de procesarores, aceleración contra el número de procesadores y por último eficiencia contra procesadores.

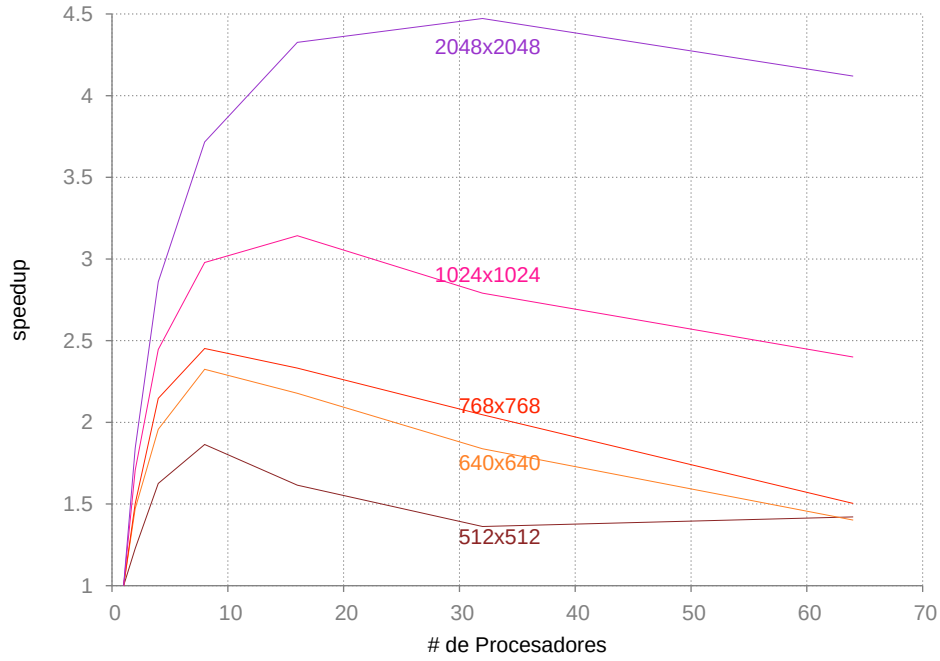


(a)

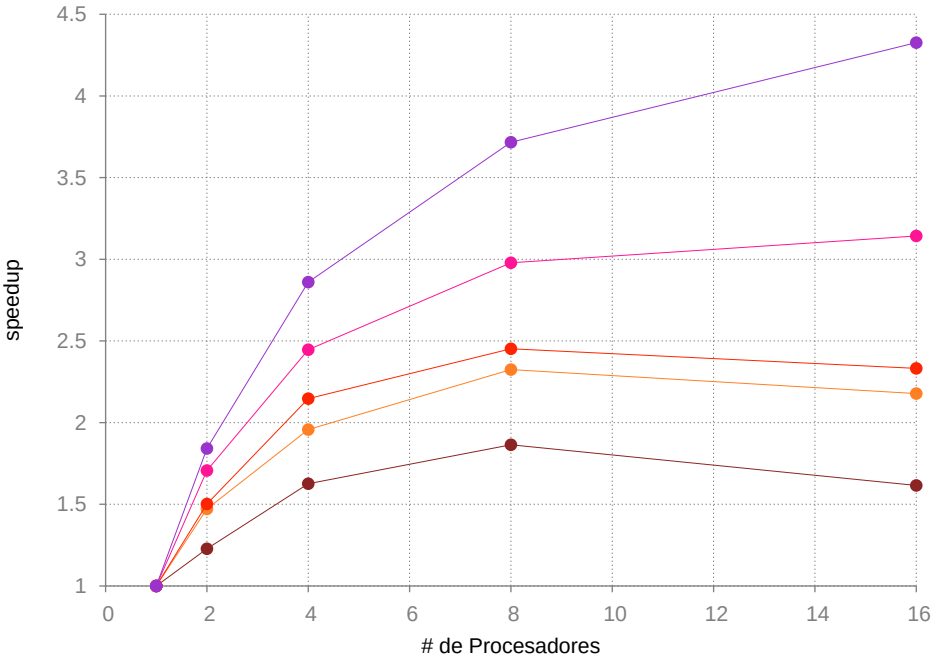


(b)

Figura 5.12: T_p Vs p

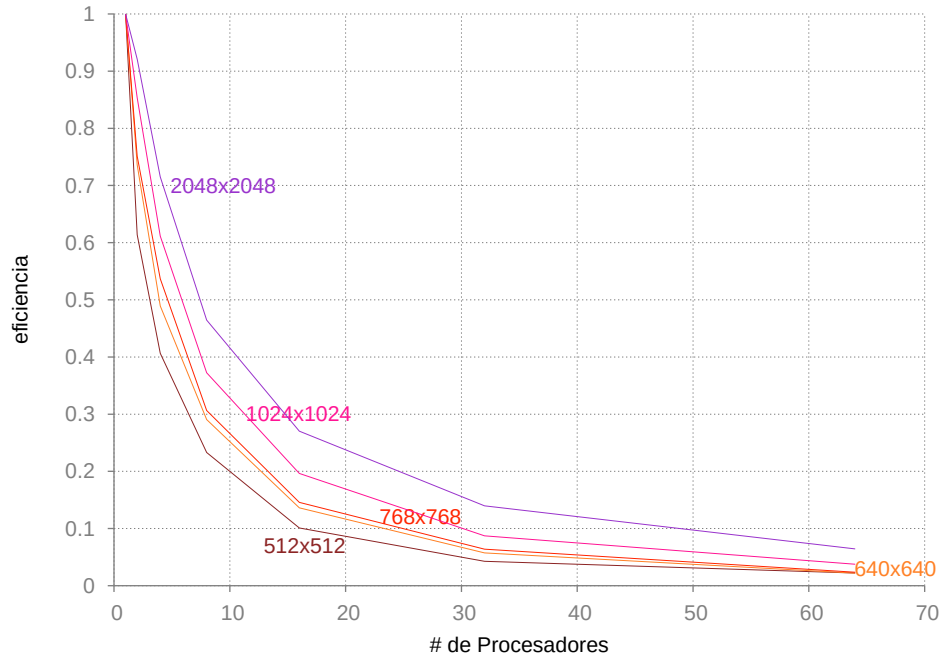


(a)

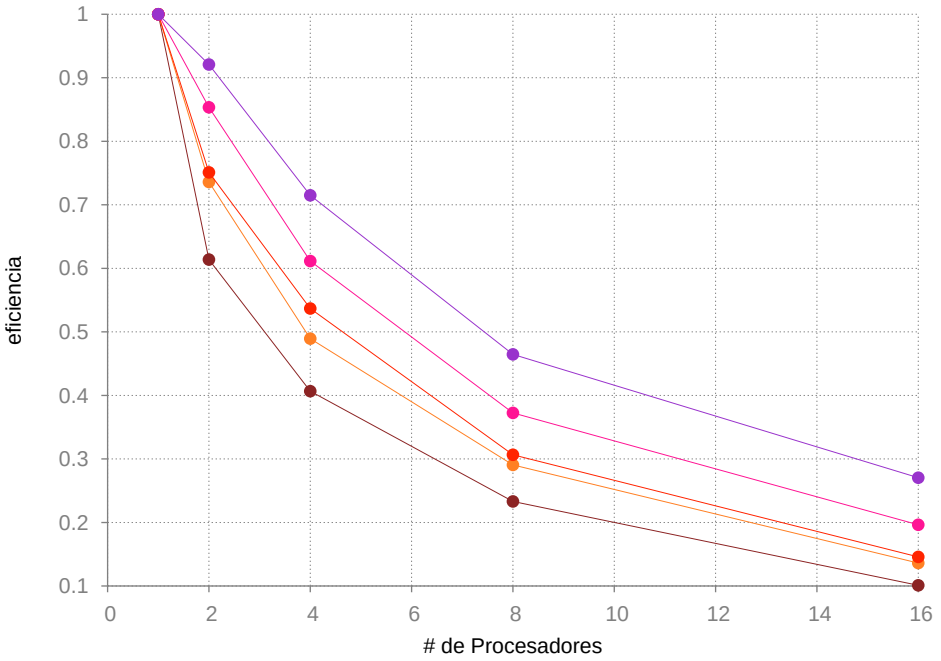


(b)

Figura 5.13: S_p Vs p



(a)



(b)

Figura 5.14: E_p Vs p

6

Conclusiones

A través de los casos de estudio básicos, se hizo una comparación con la solución analítica obteniendo errores del 0.6 % al 0.07 % en el problema 1 y en el problema 3 de 0.2 % al 0.04 % al ir aumentando el tamaño de la malla en ambos casos; estos ejemplos se usaron para calibrar y entender mejor el software en cuanto a la familiarización con las rutinas y en la forma de visualizar los resultados que se obtenían.

La solución de sistemas de ecuaciones lineales es un paso fundamental en la solución ecuaciones diferenciales por el método de elemento finito o volumen finito. En concreto el objetivo del trabajo es aplicar las estructuras de datos basadas en PETSc que se integran a DOLFIN mediante el uso de FEniCS para un problema con contaminante planteado en la bibliografía. Este problema se resolvió tanto en FEniCS como en TUNAM, con el primero teniendo posibilidades para paralelizar el código y pudiendo elegir métodos más aproximados para su solución.

Se realizó un estudio de aceleración del problema de contaminación de un acuífero, se observó que el tiempo de cálculo disminuyó conforme se aumentaba el número de procesadores, sin embargo este comportamiento se detuvo cuando se usaron ocho procesadores. Pero se observó que cuando se incrementa el número de nodos en la malla, tanto la aceleración como la eficiencia son mejores. Como se explicó antes, esto se debe a la mala conectividad de los nodos del clúster, los cuales sólo tienen doce procesadores.
A causa a la baja granularidad del sistema, es decir, el problema cuenta con muy pocos nodos debido a que se ha resuelto en dos dimensiones, la simuación requiere de mayor comunicación entre los nodos del clúster y el procesamiento disminuye, aumentando el tiempo de cálculo.

Como trabajo futuro, se desea ejecutar el mismo código en un clúster conectado mediante la tecnología infiniband[1]. Ésta tecnología permite una mayor velocidad en la

[1] Véase www.infinibandta.org

transmisión de datos. De igual manera, se desea ejecutar el problema en tres dimensiones, pues en estos casos el número de nodos se incrementa drásticamente, con lo que se espera que la aceleración y la eficiencia sean muy buenas.

Como parte del trabajo a futuro, puede verse la implementación numérica mediante el uso de GPUs (graphics processing unit es un co-procesador masivamente paralelo, dedicado a proceso de threads[2]) con CUDA (Compute Unified Device Architecture), que proporcionen importantes mejoras en el rendimiento, dado que el paralelismo de los GPUs se duplica cada año y los modelos de programación permiten ser escalados transparentemente.

Para producir mejoras, en el futuro puede emplearse unidades de procesamiento gráfico, lo cual implique cambios en el código fuente de una manera drástica.

Los resultados que se obtuvieron de la simulación numérica, son similares a los que se obtienen con software típico de simulación de aguas subterráneas con análisis para contaminantes.

[2] Véase www.nvidia.es/object/computational_fluid_dynamics_es.html

Bibliografía

[1] M. Allen, I. Herrera, and G. Pinder. *Numerical Modelling in Science and Engineering*. John Wiley & Sons, 1998.

[2] S. Balay et al. *PETSc User Manual*. Mathematics and Computer Science Division, Argonne National Laboratory, South Cass Avenue, Argonne, Illinois 60439, 3.1 edition, March 2010.

[3] J. J. Barton and L. R. Nackman. *Scientific and Engineering C++*. Addison–Wesley, 1994.

[4] M. R. Bhujade. *Parallel Computing*. New Age International, 2004.

[5] Blitz++. Object-oriented library for scientific computing. `http://www.oonumerics.org/blitz/`.

[6] Z. Chen. *Finite Element Methods And Their Applications*. Springer, 2005.

[7] Z. Chen, G. Huan, and Y. Ma. *Computational Methods for Multiphase Flows in Porous Media.* SIAM, 2006.

[8] V. T. Chow. *Handbook of Applied Hydrology*. McGraw-Hill, 1993.

[9] J. O Coplien. *Curiously recurring template patterns. C++ Report*. 1995.

[10] L. M. de la Cruz. Tuna: Template units for numerical applications. `https://code.google.com/p/tunam`.

[11] L. M. de la Cruz. *Tesis*. sometido a Trans. Math. Soft., 2010.

[12] L. M. de la Cruz and E. Ramos. *General Template Units for the Finite Volume Method in Box-shaped Domains*. sometido a Trans. Math. Soft., 2011.

[13] J. J. Dongarra et al. *Numerical linear algebra for high performance computers*. SIAM, 1998.

[14] G. Herrera. *Cost Effective Groundwater Quality Sampling Network Design*. PhD thesis, University of Vermont, 1998.

[15] I. Herrera. *Boundary methods. An algebraic theory*. Pitman Advanced Publishing Program, 1984.

[16] I. Herrera and G. Pinder. *Mathematical Modeling in Science and Engineering: An Axiomatic Approach*. Wiley, 2012.

[17] T. Hughes. *The Finite Element Method – Linear Static and Dynamic Finite Element Analysis*. Dover Publications, 2000.

[18] C. Johnson. *Numerical Solution of Partial Diferential Equations by the Finite Element Method*. Dover Publications, 2009.

[19] D. B. Kirk and W. W. Hwu. *Programming Massively Parallel Processors. A Hands-on Approach*. Elsevier, 2010.

[20] H. P. Langtangen. *A Primer on Scientific Programming with Python. Texts in Computational Science and Engineering*, volume 6. Springer, 2009.

[21] H. P. Langtangen. *Python Scripting for Computational Science*. Springer, 3rd edition, 2009.

[22] A. Logg, G. N. Wells, and Kent-Andre Mardal. *Automated Solution of Differential Equations by The Finite Element Method*. Springer, 2011.

[23] N. Myers. *Traits: a new and useful template technique. C++ Report*. 1995.

[24] J. L. Olivares. *Groundwater Quality Monitor GUI. User's Guide*.

[25] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[26] L. R. Scott, T. Clark, and B. Baghery. *Scientific Parallel Computing*. Princeton University Press, 2005.

[27] B. F. Smith, P. E. Bjørstad, and W. D. Gropp. *Domain Decomposition. Parallel Multilevel Methods for elliptic partial differential equations*. Cambridge University Press, 1996.

[28] D. Vandevoorde and N. M. Josuttis. *C++ Templates*. Addison–Wesley, 2003.

[29] T. L. Veldhuizen. *Expression templates. C++ Report*. Reprinted in C++ Gems. Stanley Lippman, 1995.

[30] H. F. Wang and M. P. Anderson. *Introduction to groundwater modeling: Finite difference and finite element methods.* Freeman and Co., 1995.