Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXVI, págs. 605-625 (artículo completo)
José G. Etse, Bibiana M. Luccioni, Martín A. Pucheta, Mario A. Storti (Eds.)
San Miguel de Tucumán, 6-9 Noviembre 2018

EVALUACIÓN DEL DESEMPEÑO DE DIFERENTES ESQUEMAS TEMPORALES PARA LA RESOLUCIÓN DE UNA ECUACIÓN DE DIFUSIÓN NO LINEAL EN GPGPU

PERFORMANCE EVALUATION OF DIFFERENT TIME SCHEMES FOR THE RESOLUTION OF A NONLINEAR DIFFUSION EQUATION IN GPGPU

Lucas C. Bessone[a,c], Pablo Gamazo[a] y Mario A. Storti[b]

[a]*Departamento del agua, CENUR LN, UdelaR, Rivera 1350, Salto, Uruguay, http://agua.unorte.edu.uy*

[b]*Centro de Investigación de Métodos Computacionales (CIMEC), CONICET, Predio CONICET Santa Fe, Colectora Ruta Nac 168, Km 472, Paraje El Pozo, Santa Fe, Argentina, http://www.cimec.org.ar*

[c]*UTN - Facultad Regional Concordia , Salta 277, Concordia, Entre Ríos, Argentina*

Palabras clave: Ecuación de Difusión No Lineal, Volúmenes Finitos, GPUs.

Resumen. El uso de GPUs para cálculo científico se ha incrementado en la última década. La mayoría de los trabajos que resuelven la ED (Ecuación de Difusión) en GPU se centran en la paralelización de los resolvedores de sistemas de ecuaciones lineales generados por los métodos numéricos. Por otro lado, como los métodos explícitos presentan un mayor potencial para aprovechar la paralelización en las GPUs, aquí se compara un caso de estos últimos frente a un método implícito. En este trabajo se resuelve una ED no lineal en 3D, usando el método de los volúmenes finitos en mallas cartesianas. Se comparan diferentes esquemas temporales, explícito e implícito, considerando para este último, el método de NR (Newton-Raphson) y el solver CG (Gradientes Conjugados) para el sistema lineal de ecuaciones. Se evalúa el desempeño de cada esquema en GPU comparando precisión, velocidad de cálculo y tamaño de malla. Para evaluar las propiedades de convergencia de los diferentes esquemas en relación a la discretización espacial y temporal, se propone una solución analítica arbitraria, la cual satisface la ED no lineal mediante la elección de un término fuente elegido en función de la misma.

Keywords: Nonlinear diffusion Equation, Finite Volume Method, GPUs.

Abstract. The use of GPUs for scientific calculation has increased in the last decade. Most works using GPU focuses on the parallelization of the solvers of linear equations generated by the numerical methods. On the other hand, since explicit methods have a greater potential to take advantage of the parallelization in the GPUs, we compare a case of the latter versus an implicit method. In this work we solve a 3D nonlinear diffusion equation, using the finite volume method in cartesian meshes. Two different time schemes are compared, explicit and implicit, considering for the latter, the Newton-Raphson method and Conjugate Gradient solver for the system of equations. The performance of each GPU scheme is evaluated by comparing accuracy, calculation speed and mesh size. To evaluate the convergence properties of the different schemes in relation to spatial and temporal discretization, an arbitrary analytical solution is proposed, which satisfies the differential equation by choosing a source term chosen based on it.

1. INTRODUCCIÓN

La modelación matemática numérica es una técnica que se aplica desde hace más de 50 años para resolver una amplia gama de fenómenos físicos (Nash, 1990; Brezinski y Wuytack, 2001). En particular la ED no lineal se utiliza para representar diversos fenómenos físicos como el flujo de calor en sólidos (Sellitto et al., 2016), la difusión de solutos en líquidos (Obukhovsky et al., 2017), difusión en cristales (Janavičius y Turskienė, 2016) o el flujo de agua subterránea en acuíferos no confinados (Shaikh y Das, 2018) entre otros. Con el paso de los años los cambios en hardware han permitido el desarrollo de modelos con mayor resolución y complejidad. En la última década las GPUs han surgido como una alternativa para computación de propósito general y su uso para cálculo científico ha ido aumentando (Che et al., 2008). Varios autores han trabajado en la resolución de la ED y advección-difusión en GPU pero en su mayoría han considerado problemas lineales (Heimlich et al., 2011). La mayoría de estos trabajos se centran en desarrollar implementaciones eficientes de resolvedores de los sistemas de ecuaciones algebraicas generados por diferentes métodos numéricos (Brandao, 2009; Cotronis et al., 2014). Por otro lado, los métodos explícitos presentan un mayor potencial para aprovechar la capacidad de paralelización de las GPUs. Bondarenco et al. compararon la eficiencia de los métodos implícitos y explícitos para la resolución de la ecuación de advección-difusión y obtuvieron una mejor performance en GPU para el método explícito (Bondarenco et al., 2017). En este trabajo se resuelve una ED no lineal en 3D, usando el método de los volúmenes finitos en mallas estructuradas uniformes. Se comparan diferentes esquemas temporales, explícito e implícito, considerando para este último, el método de NR (Newton-Raphson) y el solver CG (Gradientes Conjugados) para el sistema lineal de ecuaciones. Se evalúa el desempeño de cada esquema en GPU comparando precisión, velocidad de cálculo y tamaño de malla. Para evaluar las propiedades de convergencia de los diferentes esquemas en relación a la discretización espacial y temporal, se propone una solución analítica arbitraria, la cual satisface la ecuación diferencial mediante la elección de un término fuente elegido en función de la misma.

2. ECUACIÓN DE DIFUSIÓN NO LINEAL

Se considera el problema parabólico no lineal para $t \in [0, T], T > 0$,

$$\begin{aligned} \frac{\partial \phi}{\partial t} = \nabla \cdot (\mathbf{A}(\phi) \nabla \phi) + f, \text{ en } \Omega \in \mathbb{R}^3, \\ \phi = 0, \text{ sobre } \partial\Omega, \\ \text{con } \phi(0) = \phi^0, \text{ en } \Omega \end{aligned} \tag{1}$$

donde $\mathbf{A}(\phi) = \text{diag}\,(a_1(\phi), a_2(\phi), a_3(\phi))$, es una función matricial a valores reales, acotada y estrictamente definida positiva. En adelante asumimos que existe una función $\phi(\mathbf{x}, t)$, con $\mathbf{x} = (x, y, z) \in \Omega$, suficientemente suave solución del problema (1). Para el presente trabajo se considerará directamente el caso de difusión isotrópica con $a_i(\phi) = a(\phi)$, $i = 1, 2, 3$.

3. MÉTODO DE DISCRETIZACIÓN DE LA ECUACIÓN DE DIFUSIÓN NO LINEAL

Se puede transformar el problema (1) en un sistema de ecuaciones algebraicas usando el método de los volúmenes finitos (FVM), para luego resolver el problema discretizado en una computadora.

3.1. Discretización espacial y temporal

Para la discretización espacial se consideró un dominio cúbico $[0, L] \times [0, L] \times [0, L], L > 0$, con mallas estructuradas de paso constante en cada dirección $\Delta x = \Delta y = \Delta z = h = L/m$, siendo m la cantidad de celdas por dirección. Se utilizó FVM centrado en celdas (colocado) y diferencias centradas como esquema espacial para la discretización del operador diferencial. Considerando la molécula computacional de la figura 1, queda definida la siguiente nomenclatura para los centros de celda C y los centros de celda vecinos en las direcciones x, y y z respectivamente: E, W, N, S, T, B. Por otra parte, los subíndices en minúscula indican centros de caras, así la sumatoria sobre $f \sim nb(C)$ indica que se suman expresiones evaluadas en centros de cara f sobre todas las caras que pertenecen a la celda C, es decir, caras vecinas (neighbors) al centro C y $F \sim NB(C)$ indica a todos los centros de celda F vecinas a la celda C.

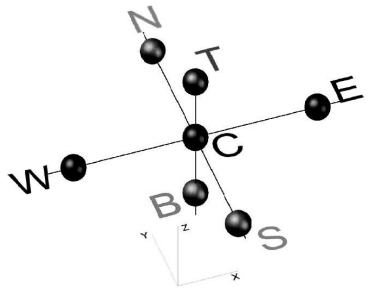

Figura 1: Molécula computacional.

Para aplicar el FVM al problema (1), primero se integra sobre la celda C:

$$\int_{V_C} \frac{\partial \phi}{\partial t} dV = \int_{V_C} \nabla \cdot (\mathbf{A}(\phi) \nabla \phi)\, dV + \int_{V_C} f dV \tag{2}$$

usando el teorema de la divergencia para el segundo término y el teorema del valor medio para el término temporal y fuente:

$$\begin{aligned} \left(\frac{\partial \phi}{\partial t}\right)_C V_C = \int_{\partial V_C} (\mathbf{A}(\phi) \nabla \phi)\, dS + f_C V_C \Leftrightarrow \\ \left(\frac{\partial \phi}{\partial t}\right)_C h^3 = \sum_{f \sim nb(C)} (\mathbf{A}(\phi_f) \nabla \phi_f)\, h^2 + f_C h^3 = \sum_{f \sim nb(C)} (a(\phi_f) \nabla \phi_f)\, h^2 + f_C h^3 \end{aligned} \tag{3}$$

donde la sumatoria abarca como se dijo anteriormente todos los centros de cara f de la celda C. Haciendo la suma de las contribuciones a través de todas las caras de la celda C, usando diferencias centradas para el gradiente sobre las caras e interpolación lineal para el coeficiente de difusión $a(\phi)$ se obtiene forma semi-discretizada de (3):

$$\left(\frac{\partial \phi}{\partial t}\right)_C h^3 = h^2 \left(\sum_{F \sim NB(C)} \left(\frac{a_C + a_F}{2}\right) \left(\frac{\phi_F - \phi_C}{h}\right) + f_C h \right) \tag{4}$$

Para la discretización temporal se aplicó el método theta (θ) a la ecuación (4):

$$\left(\frac{\phi^{t+\Delta t}-\phi^{t}}{\Delta t}\right)_C h^3 = h^2(1-\theta)\left(\sum_{F\sim NB(C)}\left(\frac{a_C+a_F}{2}\right)\left(\frac{\phi_F-\phi_C}{h}\right)+f_C h\right)^{t} + h^2\theta\left(\sum_{F\sim NB(C)}\left(\frac{a_C+a_F}{2}\right)\left(\frac{\phi_F-\phi_C}{h}\right)+f_C h\right)^{t+\Delta t} \tag{5}$$

donde $\theta \in [0,1]$, de forma que para $\theta = 0$ se obtiene el esquema explícito de primer orden, conocido como diferencias hacia adelante o FE (Forward Euler) y para $\theta = 1/2$ se obtiene el esquema implícito o CN (Crank-Nicolson) con precisión de segundo orden temporal. En lo que sigue se usarán supraíndices de forma que ϕ_C^n corresponde a la variable evaluada en el centro de celda C en tiempo actual $t = n\Delta t$. En las subsecciones siguientes se desarrollan ambos esquemas, explícito e implícito.

3.2. Esquema explícito - Diferencias hacia adelante

Reemplazando $\theta = 0$ en (5), reordenando y agrupando se obtiene el método explícito o FE:

$$\phi_C^{n+1} = \phi_C^n + \frac{\Delta t}{2h^2}\left(\sum_{F\sim NB(C)}\left(a_C^n + a_F^n\right)\left(\phi_F^n-\phi_C^n\right)+2f_C^n h^2\right) \tag{6}$$

Aquí se destaca que si bien aparecen términos en la sumatoria no lineales en la variable ϕ en la ecuación (6) debido a que se consideran los casos en que $a_C^n = a(\phi_C^n)$, todos son evaluados en tiempo actual, resultando en una expresión explícita que se puede evaluar en tiempo futuro $n+1$ conociendo la solución actual, esto es lo que constituye la principal ventaja de los métodos explícitos. En las secciones siguientes se darán mas detalles del esquema.

3.3. Esquema implícito - Diferencias centradas

Para el esquema CN se reemplaza $\theta = 1/2$ en (5):

$$\frac{\phi_C^{n+1}-\phi_C^n}{\Delta t}h^3 = \frac{h^2}{2}\left(\sum_{F\sim NB(C)}\left(\frac{a_C^n+a_F^n}{2}\right)\left(\frac{\phi_F^n-\phi_C^n}{h}\right)+f_C^n h\right) + \frac{h^2}{2}\left(\sum_{F\sim NB(C)}\left(\frac{a_C^{n+1}+a_F^{n+1}}{2}\right)\left(\frac{\phi_F^{n+1}-\phi_C^{n+1}}{h}\right)+f_C^{n+1} h\right) \tag{7}$$

El segundo término del lado derecho de la ecuación (7) indica que para calcular la solución en instante $n+1$ es necesario resolver un sistema lineal de ecuaciones algebraicas. Para el caso en que a es independiente de la variable (ED lineal) el sistema que se debe resolver es lineal. Cuando a depende de la variable se debe resolver un sistema no lineal, así por ejemplo si se

considera un coeficiente de difusión $a(\phi) = \kappa(\phi + 1)$ el sistema no lineal luce:

$$\begin{aligned}\frac{\phi_C^{n+1} - \phi_C^n}{\Delta t} h^3 = \frac{h^2\kappa}{2} \left(\sum_{F \sim NB(C)} \left(\frac{(\phi_C^n + 1) + (\phi_F^n + 1)}{2} \right) \left(\frac{\phi_F^n - \phi_C^n}{h} \right) + f_C^n h \right) \\ + \frac{h^2\kappa}{2} \left(\sum_{F \sim NB(C)} \left(\frac{(\phi_C^{n+1} + 1) + (\phi_F^{n+1} + 1)}{2} \right) \left(\frac{\phi_F^{n+1} - \phi_C^{n+1}}{h} \right) + f_C^{n+1} h \right)\end{aligned} \tag{8}$$

Para resolver el sistema no lineal que surge de considerar a la ecuación (7) para todo el dominio se utilizó el método de NR. Reescribiendo (7) en forma de residuo se obtiene:

$$\begin{aligned}R(\phi^{n+1}) = \phi_C^n - \phi_C^{n+1} + \frac{\Delta t}{2} \left(f_C^{n+1} + f_C^n \right) + \\ \frac{\Delta t}{4h^2} \sum_{F \sim NB(C)} \left[\left(A_C^n + A_F^n \right) \left(\phi_F^n - \phi_C^n \right) + \left(A_C^{n+1} + A_F^{n+1} \right) \left(\phi_F^{n+1} - \phi_C^{n+1} \right) \right] = 0\end{aligned} \tag{9}$$

El método de NR consiste en resolver iterativamente el sistema linealizado

$$R(\phi) \approx R(\phi^{(k)}) + \mathbf{J}(\phi^{(k)}) \left(\phi^{(k+1)} - \phi^{(k)} \right) = 0 \tag{10}$$

Definiendo $\Delta\phi^{(k+1)} = \phi^{(k+1)} - \phi^{(k)}$ la ecuación (10) puede expresarse como

$$R(\phi^{(k)}) + \mathbf{J}(\phi^{(k)}) \Delta\phi^{(k+1)} = 0 \tag{11}$$

o equivalentemente

$$\phi^{(k+1)} = \phi^{(k)} - \left(\mathbf{J}(\phi^{(k)}) \right)^{-1} R(\phi^{(k)}) \tag{12}$$

donde $\mathbf{J}(\phi^{(k)})$ es el Jacobiano del residuo R evaluado en $\phi^{(k)}$ que corresponde a la variable en tiempo futuro ϕ^{n+1} en la k-ésima iteración.

3.4. Algoritmo para el método de Newton-Raphson con un esquema temporal implícito

En la diciplina usualmente el método de NR resulta en el siguiente algoritmo para resolver (7), en cada paso de tiempo se debe realizar lo siguiente:

1. Usar $\phi^{(0)} = \phi^n$.
2. Evaluar $\mathbf{J}(\phi^{(k)})$ y ensamblar el sistema lineal (10).
3. Resolver el sistema lineal para $\Delta\phi^{(k+1)}$.
4. Setear $\phi^{(k+1)} = \phi^{(k)} + \Delta\phi^{(k+1)}$ y volver al paso 2 hasta la convergencia del esquema iterativo.

Es decir, que para cada paso de tiempo se deben ensamblar y resolver un sistema lineal K veces, uno por cada iteración NR, para poder obtener la solución en el paso de tiempo siguiente ϕ^{n+1}. Sin embargo, un detalle a tener en cuenta es que resolver algoritmo iterativo de NR hasta la convergencia puede no ser necesario como se mostrará en la siguiente sección.

3.5. Orden de aproximación de la iteración Newton-Raphson al usar el esquema Crank-Nicolson

En vías de optimizar la implementación y como en general no es mencionado en la literatura, siendo un factor que debe tenerse en cuenta en las implementaciones, a continuación se mostrará que el orden de aproximación temporal para una iteración NR al usar el esquema de diferencias centradas (CN) es al menos $O(\Delta t^2)$. Esto implica que el nivel de precisión de la solución no cambia de orden con las sucesivas iteraciones de NR. Para probar esto, se considerará sin pérdida de generalidad el sistema de ODEs:

$$\frac{\partial \phi}{\partial t} = \mathbf{f}(\phi, t), \text{ con } \mathbf{f}(\phi, t) : \mathbb{R}^m \to \mathbb{R}^m \text{ una función no lineal de } \phi \tag{13}$$

recordando las discretizaciones temporales básicas FE, BE (Backward Euler) y CN y sus errores de truncamiento respectivamente:

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = \mathbf{f}(\phi^n, t) + O(\Delta t) \tag{14}$$

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = \mathbf{f}(\phi^{n+1}, t) + O(\Delta t) \tag{15}$$

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = \frac{1}{2}\left(\mathbf{f}(\phi^{n+1}, t) + \mathbf{f}(\phi^n, t)\right) + O(\Delta t^2) \tag{16}$$

definiendo $\mathbf{z}^n = \phi^{n+1} - \phi^n$ y reemplazando en (16) se obtiene:

$$\mathbf{z}^n = \frac{\Delta t}{2}\left[\mathbf{f}(\phi^n + \mathbf{z}^n, t) + \mathbf{f}(\phi^n, t)\right] + O(\Delta t^3) \tag{17}$$

escribiendo (17) de manera truncada y en forma de residuo:

$$\mathbf{R}(\mathbf{z}^n) = \mathbf{z}^n - \frac{\Delta t}{2}\left[\mathbf{f}(\phi^n + \mathbf{z}^n, t) + \mathbf{f}(\phi^n, t)\right] = 0 \tag{18}$$

se calcula el Jacobiano del residuo:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{z}^n} = \mathbf{I} - \frac{\Delta t}{2}\mathbf{J}_f(\phi^n + \mathbf{z}^n, t) \tag{19}$$

siendo $\mathbf{I}$ la matriz identidad y $\mathbf{J}_f$ el Jacobiano de la función no lineal $\mathbf{f}$ de (13). El esquema iterativo de NR luce así:

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} - \left(\mathbf{J}_f(\mathbf{z}^{(k)})\right)^{-1}\mathbf{R}(\mathbf{z}^{(k)}) \tag{20}$$

Para evaluar el error de aproximación temporal cometido para una iteración de NR, se considerará $\mathbf{z}^{(0)} = 0$, obteniéndose los siguientes residuo y Jacobiano:

$$\begin{aligned} \mathbf{R}(0) &= -\Delta t \mathbf{f}(\phi^n, t) \\ \frac{\partial \mathbf{R}}{\partial \mathbf{z}^n}(0) &= \mathbf{I} - \frac{\Delta t}{2}\mathbf{J}_f(\phi^n, t) \end{aligned} \tag{21}$$

reemplazando en (20) se obtiene:

$$\mathbf{z}^{(1)} = 0 - \left(\mathbf{I} - \frac{\Delta t}{2}\mathbf{J}_f(\phi^n, t)\right)^{-1}(-\Delta t \mathbf{f}(\phi^n, t)) = \left(\mathbf{I} - \frac{\Delta t}{2}\mathbf{J}_f(\phi^n, t)\right)^{-1}\Delta t \mathbf{f}(\phi^n, t) \tag{22}$$

Considerando la serie geométrica $\frac{1}{1-x} = 1 + x + x^2 + x^3 + ...$ se puede escribir la inversa que aparece en la ecuación (22) como:

$$\left(\mathbf{I} - \frac{\Delta t}{2}\mathbf{J}_f(\phi^n, t)\right)^{-1} = \mathbf{I} + \frac{\Delta t}{2}\mathbf{J}_f(\phi^n, t) + \frac{\Delta t^2}{4}\left[\mathbf{J}_f(\phi^n, t)\right]^2 + O(\Delta t^3) \tag{23}$$

combinando (22), (23) y usando $\phi^{n+1} = \phi^n + \mathbf{z}^n$:

$$\begin{aligned} \phi^{n+1} &= \phi^n + \Delta t \mathbf{f}(\phi^n, t) + \frac{\Delta t^2}{2}\mathbf{J}_f(\phi^n, t)\mathbf{f}(\phi^n, t) + O(\Delta t^3) \\ \Leftrightarrow \phi^{n+1} &= \phi^n + \frac{\Delta t}{2}\mathbf{f}(\phi^n, t) + \frac{\Delta t}{2}\left[\mathbf{f}(\phi^n, t) + \Delta t\ \mathbf{J}_f(\phi^n, t)\ \mathbf{f}(\phi^n, t)\right] + O(\Delta t^3) \end{aligned} \tag{24}$$

como se probará más adelante, la expresión entre corchetes en (24) se puede aproximar como:

$$\mathbf{f}(\phi^n, t) + \Delta t\ \mathbf{J}_f(\phi^n, t)\ \mathbf{f}(\phi^n, t) = \mathbf{f}(\phi^{n+1}, t) + O(\Delta t^2) \tag{25}$$

reemplazando (25) en (24) se obtiene:

$$\begin{aligned} \phi^{n+1} = \phi^n + \frac{\Delta t}{2}\mathbf{f}(\phi^n, t) + \frac{\Delta t}{2}\left[\mathbf{f}(\phi^{n+1}, t) + O(\Delta t^2)\right] + O(\Delta t^3) \\ \phi^{n+1} = \phi^n + \frac{\Delta t}{2}\mathbf{f}(\phi^n, t) + \frac{\Delta t}{2}\mathbf{f}(\phi^{n+1}, t) + O(\Delta t^3) \\ \phi^{n+1} = \phi^n + \frac{\Delta t}{2}\left(\mathbf{f}(\phi^n, t) + \mathbf{f}(\phi^{n+1}, t)\right) + O(\Delta t^3) \end{aligned} \tag{26}$$

Puede observarse que la última expresión de la ecuación (26) es idéntica a (16). En la práctica al implementar el método de NR se resuelve el sistema lineal que equivale a obtener la inversa del Jacobiano en (20). Considerando la ecuación (26) se puede observar que al resolver el sistema linealizado de NR una única vez se obtiene una solución con un error de truncamiento proporcional a Δt^3. Dicho error es del mismo orden que el del esquema temporal de CN (ecuación (16)). Por lo tanto, si bien sucesivas iteraciones de NR producirían soluciones con un error numéricamente menor, el mismo mantendría el mismo orden. Expresado de otra manera, una única iteración en el esquema NR entrega una solución con un error del mismo orden que el de la solución de convergencia.

Para finalizar se demostrará la equivalencia considerada en la ecuación (25). Usando el polinomio de Taylor se puede escribir

$$\mathbf{f}(\phi^{n+1}, t) = \mathbf{f}(\phi^n, t) + \mathbf{J}_f(\phi^n, t)\left(\phi^{n+1} - \phi^n\right) + \frac{1}{2}\frac{\partial^2 \mathbf{f}}{\partial \phi^{n2}}(\phi^{n+1} - \phi^n)^2 \tag{27}$$

usando (14) puede verse que:

$$\begin{aligned} \phi^{n+1} - \phi^n &= \Delta t \mathbf{f}(\phi^n, t) + O(\Delta t^2) \\ (\phi^{n+1} - \phi^n)^2 &= \Delta t^2 \left(\mathbf{f}(\phi^n, t)\right)^2 + O(\Delta t^3) \end{aligned} \tag{28}$$

reemplazando las últimas dos en la expresión (27) queda:

$$\begin{aligned} \mathbf{f}(\phi^{n+1}, t) = \mathbf{f}(\phi^n, t) + \mathbf{J}_f(\phi^n, t)\left[\Delta t \mathbf{f}(\phi^n, t) + O(\Delta t^2)\right] \\ + \frac{\partial^2 \mathbf{f}}{\partial \phi^{n2}}\left[\frac{\Delta t^2}{2}\left(\mathbf{f}(\phi^n, t)\right)^2 + O(\Delta t^3)\right] \\ \Leftrightarrow \mathbf{f}(\phi^{n+1}, t) = \mathbf{f}(\phi^n, t) + \Delta t \mathbf{J}_f(\phi^n, t)\mathbf{f}(\phi^n, t) + O(\Delta t^2) \end{aligned} \tag{29}$$

esto valida la aproximación (25). Con esto se prueba que para sitemas no lineales en los que no se tenga grandes inestabilidades debidas a la no linealidad, el algoritmo de la subsección anterior puede modificarse realizando una sola iteración NR sin perder precisión, esto implica la solución de un único sistema lineal por cada paso de tiempo como los esquemas implícitos de las ecuaciones lineales.

4. ALGORITMOS

Se implementaron dos algoritmos: uno explícito y otro considerando el esquema de CN. La implementación del esquema explícito (ecuación (6)) es trivial ya que por cada paso de tiempo hay una única instrucción. Los criterios considerados para definir el paso de tiempo junto a otros detalles de la implementación se detallarán en las secciones posteriores. A continuación se describe el algoritmo para el esquema implícito de CN.

4.1. Algoritmo para el esquema de Crank-Nicolson

Los pasos a realizar para el esquema implícito son:

1. $\phi^n \leftarrow \phi^0, \phi^{(0)} \leftarrow \phi^0$
2. Computar $R^{(0)}$, y el residuo inicial $r^{(0)} \leftarrow ||R^{(0)}||$
3. Mientras $r^{(k)} > (\text{tol}_{rel}\ r^{(0)} + \text{tol}_{abs})$ hacer:
   - Computar $\mathbf{J}^{(k)} \leftarrow \mathbf{J}(\phi^n, \phi^{(k)})$
   - Resolver el sistema $\mathbf{J}^{(k)} \Delta\phi^{(k)} = -R^{(k)}$
   - $\phi^{(k+1)} \leftarrow \phi^{(k)} + \Delta\phi^{(k)}$
   - Computar $R^{(k)}$, y el residuo inicial $r^{(k)} \leftarrow ||R^{(k)}||$
4. $\phi^{n+1} \leftarrow \phi^{(k+1)}$, $\phi^n \leftarrow \phi^{n+1}$, volver al paso 2 mientras $(n\Delta t < T)$

Aquí debe notarse que hay problemas para los que en el paso 3 una sola iteración es suficiente y solo se requiere iterar hasta la convergencia si el problema es fuertemente no lineal.

5. DETALLES DE LA IMPLEMENTACION

5.1. Hardware

Los algoritmos implementados se ejecutaron en equipos Dell PowerEdge R720 con las siguientes características de hardware: 2 procesadores Intel Xeon(R) CPU E5-2620v2 (6 cores de 2.1GHz, aquitectura Sandy-Bridge) con 128GB DDR3 de memoria RAM. Cada equipo cuenta con 2 tarjetas NVIDIA K40 GPU (2880 CUDA cores corriendo a 745MHz, arquitectura Kepler, con 12GB DDR5 de memoria), conectada via el bus PCI-e.

5.2. Software

El sistema operativo es CentOS Linux v7.5. El compilador GCC 4.8.5, la version 9.1 para el compilador nvcc de CUDA. Todos las implementaciones se realizaron en DP (doble precisión). Para las operaciones de álgebra lineal se utilizaron las librerías provistas por Thrust[1] y CUSP[2] v0.5.0.

[1] http://code.google.com/p/thrust/
[2] http://code.google.com/p/cusp_library/

5.3. Implementación GPGPU

5.3.1. Paralelización del método explícito

La implementación se realizó teniendo en cuenta que el almacenamiento de datos y la totalidad de cálculos se realicen completamente en la GPU. Para mejorar la eficiencia se tuvo en cuenta la arquitectura SIMT (Single Instruction Multiple Threads) de CUDA, se dividió el procesamiento de la malla en diferentes CUDA kernels, de manera que se tiene un kernel para cada caso de borde y uno para las celdas interiores de acuerdo al siguiente esquema:

- 8 kernels para procesar cada vertice del dominio.
- 12 kernels para procesar las aristas.
- 6 kernels para procesar las caras.
- 1 kernel para procesar el interior del dominio.

de esta forma todos estos kernels pueden correrse de manera concurrente ya que los resultados sólo dependen de los datos del paso de tiempo anterior.

Usando la librería CUSP se pueden definir tipos de vectores alocados en la memoria del dispositivo, esto evita hacer la alocatación y posterior liberación de la memoria de manera explícita por parte del usuario (`cudaMalloc()` y `cudaFree()` respectivamente), a la vez que el acceso a los datos del vector almacenado en el dispositivo para el usuario son directos desde el host, evitando el uso de la llamada a `cudaMemcpy()`, por ejemplo para exportar los resultados de la simulación o salidas en pantalla (esto no cambia la performance del código pero hace más amena su lectura).

La implementación del kernel para procesar el interior del dominio consiste en un lazo que permite recorrer el dominio en dirección z mediante planos xy uno para cada elevación z del dominio discretizado (ver figura 2). Respecto a los accesos a los datos del dispositivo dentro del kernel, se requiere por cada celda 7 accesos, de los cuales 2 corresponden al plano superior e inferior al plano xy que se está procesando. La estrategia adoptada entonces es que cada hilo (CUDA threads) procese un punto del plano actual, que el plano actual xy sea cargado en memoria compartida y que el plano actual pase a ser el plano posterior en la siguiente iteración del lazo interno del kernel, evitando así los accesos redundantes a la memoria global del dispositivo. Para los planos del borde se procede igual que los planos xy interiores del dominio usando la memoria compartida. Los cálculos asociados a las aristas y a los vértices se realizan sin usar memoria compartida debido a que no se observaron mejoras al usarla en esos casos.

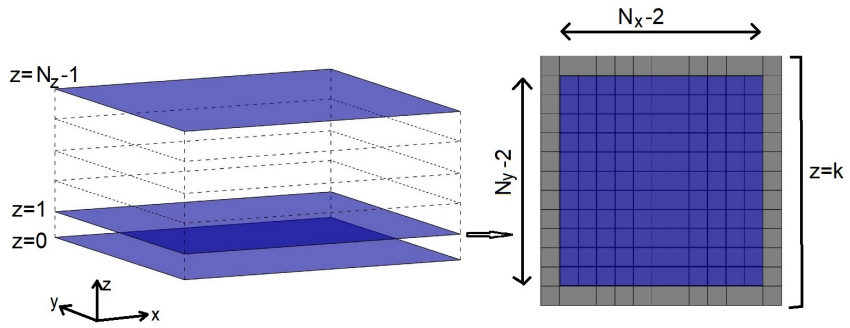

Figura 2: Procesamiento del dominio computacional por planos.

A falta de un análisis con el método de Von Neumann para la estabilidad global del método explícito, en la ecuación no lineal que se esta estudia aquí, se optó por hacer una comprobación de la estabilidad restringiendo el paso de tiempo de acuerdo al número de Fourier local, que para el problema 3D resuelto es $\Delta t < \frac{h^2}{a(\phi)\,6}$. Como el coeficiente de difusión va cambiando al cambiar el tiempo, el paso de tiempo es adaptativo de forma que $Fo \approx 0{,}998$ durante toda la simulación, en particular, se actualiza Δt cada 2000 pasos de tiempo determinando $\phi_{\text{máx}}$ en el dominio lo que permite calcular el $\Delta t_{\text{máx}}$ de acuerdo al estado de la solución.

Para ilustrar el algoritmo para el método explícito se usará la siguiente notación: ϕ^*_{input} y ϕ^*_{output} indican los punteros a los únicos dos vectores almacenados en la memoria global del dispositivo. El algoritmo es el siguiente:

1. $\phi^*_{\text{input}} \leftarrow \phi^n$ apunta al vector que contiene el estado inicial ϕ^n y $\phi^*_{\text{output}} \leftarrow \phi^{n+1}$ apunta al vector donde se guardará la nueva solución.
2. $t \leftarrow t + \Delta t$
3. Actualización de la solución:
   - kernel_Interior$(\phi^*_{\text{output}}, \phi^*_{\text{input}})$
   - kernel_Vertex1$(\phi^*_{\text{output}}, \phi^*_{\text{input}})$
   - ...
   - kernel_Edge1$(\phi^*_{\text{output}}, \phi^*_{\text{input}})$
   - ...
   - kernel_Side1$(\phi^*_{\text{output}}, \phi^*_{\text{input}})$
4. Sincronización del dispositivo
5. Intercambio de punteros:
   - $\phi^*_{\text{output}} \longleftrightarrow \phi^*_{\text{input}}$
6. Mientras ($t < T$) volver al paso 2, sino retornar

5.3.2. Paralelización del método implícito

Para el algoritmo implícito se requiere resolver el sistema lineal de ecuaciones, la matriz es SPD (simétrica definida positiva) así que se utilizó el resolvedor CG implementado en la librería CUSP, que tiene como opcional usar el estilo matrix free. Lo anterior permite definir una clase para el operador lineal, llamado al hacer el producto matriz vector Ax que se requiere en el resolvedor. Así es que el operador lineal que llama el resolvedor consiste en varios kernels, dividiendo el procesamiento del dominio como en el caso explícito, de esta manera cada hilo procesa las operaciones requeridas por una celda del dominio. En este caso, los cálculos de una celda consisten en el producto de una fila de la matriz $\mathbf{J}(\phi^{(k)})$ por el vector auxiliar del resolvedor. Al usar este estilo se evita el almacenamiento de la matriz $\mathbf{J}$ ya que se ensambla en el momento de usarla.

Respecto al cómputo del residuo $R(\phi^{(k)})$, se realiza con la misma estrategia de paralelización que el caso explícito, mediante procedimientos del tipo: kernel$(R^*, \phi^{n*}, \phi^{(k)*})$. El algoritmo para el método implícito es el siguiente:

1. $\phi^*_{\text{input}} \leftarrow \phi^n$ apunta al vector que contiene el estado inicial ϕ^n y $\phi^*_{\text{output}} \leftarrow \phi^{n+1}$ apunta al vector donde se guardará la nueva solución.
2. $t \leftarrow t + \Delta t$
3. Cálculo del vector residuo inicial $R(\phi^{(0)})$ con kernels concurrentes, sincronización,
4. Cálculo del la norma del residuo inicial $r^{(0)}$ (usando rutinas de CUSP),
5. Seteo del operador lineal para el producto Ax (función de $\phi^{(k)}$)
6. Solución del sistema usando el resolvedor CG
7. $\phi^{(k+1)} \leftarrow \phi^{(k)} + \Delta\phi^{(k)}$ operación axpy con kernels concurrentes, sincronización,
8. Cálculo del vector residuo $R(\phi^{(k)})$ con kernels concurrentes, sincronización,
9. Cálculo del la norma del residuo $r^{(k)}$ (usando rutinas de CUSP),
10. Si ($r^{(k)} > \text{tol}_{rel}\ r^{(0)} + \text{tol}_{abs}$) volver al paso 5, sino seguir (se usa $\phi^{(k+1)}$ como la solución actualizada ϕ^{n+1})
11. Intercambio de punteros
    - $\phi^*_{\text{output}} \longleftrightarrow \phi^*_{\text{input}}$
12. Si ($t < T$) volver al paso 2, sino retornar

En el método implícito se evaluaron variantes de los algoritmos encontrándose que en los casos en que el sistema es muy no lineal, resolver una sola iteración NR implica la solución del sistema lineal con CG hasta una precisión aceptable y esto requiere de una cantidad considerable de iteraciones CG, mientras que si el sistema se resuelve de manera aproximada en CG, esto es, por ejemplo fijando la cantidad máxima de iteraciones CG y actualizando el sistema lineal a resolver (iteración NR) se obtienen ganancias de tiempo de cómputo en un factor de 10 para un mismo error final en la solución obtenida al salir del lazo de NR. Por el contrario, en la medida que el problema se vuelve mas lineal, es más conveniente hacer una sola iteración de NR y resolver el sistema con CG hasta una precisión aceptable.

Los mejores desempeños para el algoritmo implícito se encontraron variando dinámicamente el número de iteraciones CG, el criterio que se utilizó para esto fue el de incrementar en un porcentaje dado el número $IT_{\text{máx},CG}$ (iteraciones máximas del CG) si la tasa de convergencia del residuo del método de NR baja en menos de un orden de magnitud ($\log(r^{(k)}/r^{(k-1)}) \sim 1$) y reducir $IT_{\text{máx},CG}$ si la tasa de convergencia de NR es muy alta ($\log(r^{(k)}/r^{(k-1)}) > 3/2$). De esta manera se evita realizar iteraciones CG sin antes actualizar el sistema a resolver, es decir pasar a la siguiente iteración de NR en los casos que la convergencia NR avanza rápido, y resolver mejor el sistema (aumentando $IT_{\text{máx},CG}$) si la tasa de convergencia de NR es baja.

6. CASO DE ESTUDIO

6.1. Ecuación de difusión no lineal y solución analítica

Se resolvió el problema (1) en el dominio $\Omega = [0,1] \times [0,1] \times [0,1]$, con condiciones de borde $\phi(\mathbf{x}, t) = 0$ sobre $\partial\Omega$ y $t > 0$. Se consideró un coeficiente de difusión $a(\phi) = \kappa(\phi + 1)$, donde κ es análogo al coeficiente de difusión lineal, para las simulaciones se usó $\kappa = 1$.

Se consideró la siguiente solución analítica:

$$\phi(\mathbf{x}, t) = 160e^{-t}(x - x^2)(y - y^2)(z - z^2) \tag{30}$$

y se calculó f de forma $\phi(\mathbf{x}, t)$ safisfaga (1):

$$\begin{aligned} f = {} & 160e^{-t}xyz(x-1)(y-1)(z-1) \\ & -320\kappa xye^{-t}(x-1)(y-1)\left(160xyze^{-t}(x-1)(y-1)(z-1)-1\right) \\ & -320\kappa xze^{-t}(x-1)(z-1)\left(160xyze^{-t}(x-1)(y-1)(z-1)-1\right) \\ & -320\kappa yze^{-t}(y-1)(z-1)\left(160xyze^{-t}(x-1)(y-1)(z-1)-1\right) \\ & -25600x^2y^2e^{-2t}(2z-1)^2(x-1)^2(y-1)^2 \\ & -25600x^2z^2e^{-2t}(2y-1)^2(x-1)^2(z-1)^2 \\ & -25600y^2z^2e^{-2t}(2x-1)^2(y-1)^2(z-1)^2 \end{aligned} \tag{31}$$

Con esta solución propuesta, la condición inicial para el problema es $\phi(\mathbf{x}, 0) = \phi^0 = 160(x - x^2)(y - y^2)(z - z^2)$. Mediante esta solución es posible evaluar los errores para las diferentes implementaciones que se realizaron.

6.2. Implementación de las condiciones de borde

En la resolución numérica de ecuaciones diferenciales ya sea que se utilice el método de diferencias finitas o el de volúmenes finitos como en el presente trabajo, se debe hacer un tratamiento especial en los stencils de los bordes debido a que en general el orden de precisión de los esquemas se deteriora y baja el orden de aproximación, esto depende tanto del esquema espacial que se esté usando (por ejemplo CD (esquema centrado ó Center Differences), SOU (Second Order Upwind), QUICK (Quadratic Upstream Interpolations for Convective Kinematics) entre otros), como de las condiciones de borde que se deben aplicar (por ejemplo condiciones de borde tipo Dirichlet o a valor de la variable fijado, Neumann o especificación del valor en la derivada normal al borde), algunas estrategias para abordar esto pueden encontrarse en Ferziger y Peric (2012). En el presente caso, se utilizó un esquema espacial de derivadas centradas y la estrategia adoptada para mantener el segundo orden en la discretización espacial en las fronteras es la siguiente: en los contornos se ajustan los valores de los dos puntos internos más cerca próximo al borde (en dirección normal) y la condición de borde usando un perfil parabólico. Luego con la derivada del polinomio se evalúa el gradiente normal al borde.

Esquema explícito. A modo de ejemplo, se muestra el stencil para los nodos internos y para los del lado este del dominio ($\mathbf{x} \in \Omega$ tales que $x = 1 - h/2$, $y \in [0,1]$, $z \in [0,1]$):

$$\phi_C^{n+1} = \phi_C^n + f_C^n \Delta t + \frac{\kappa \Delta t}{2h^2}\left[\sum_{F\sim\{E,N,W,S,T,B\}} (\phi_F^n + 1)^2 \; - 6\,(\phi_C^n + 1)^2\right] \tag{32}$$

$$\phi_C^{n+1} = \phi_C^n + f_C^n \Delta t + \frac{\kappa \Delta t}{2h^2} \left[\sum_{F \sim \{N,S,T,B\}} (\phi_F^n + 1)^2 + \left(\phi_W^n + \frac{4}{3}\right)^2 - 5\left(\phi_C^n + \frac{8}{5}\right)^2 + \frac{316}{45} \right] \tag{33}$$

Esquema implícito. En el caso del esquema implícito se requiere calcular el residuo R y el Jacobiano J para el método de NR, el stencil para el residuo en celdas internas queda así:

$$\begin{aligned} R_C^{(k+1)} &= \frac{h^3}{\Delta t}\left(\phi_C^n - \phi_C^{(k)} + \frac{\Delta t}{2}(f_C^n + f_C^{n+1})\right) \\ &+ \frac{\kappa h}{4}\left(\sum_{F \sim \{E,N,W,S,T,B\}} (\phi_F^n + 1)^2 - 6(\phi_C^n + 1)^2\right) \\ &+ \frac{\kappa h}{4}\left(\sum_{F \sim \{E,N,W,S,T,B\}} (\phi_F^{(k)} + 1)^2 - 6(\phi_C^{(k)} + 1)^2\right) \end{aligned} \tag{34}$$

para el caso del lado norte:

$$\begin{aligned} R_C^{(k+1)} &= \frac{h^3}{\Delta t}\left(\phi_C^n - \phi_C^{(k)} + \frac{\Delta t}{2}(f_C^n + f_C^{n+1})\right) \\ &+ \frac{\kappa h}{4}\left(\sum_{F \sim \{E,W,T,B\}} (\phi_F^n + 1)^2 + \left(\phi_S^n + \frac{4}{3}\right)^2 - 5\left(\phi_C^n + \frac{8}{5}\right)^2 + \frac{316}{45}\right) \\ &+ \frac{\kappa h}{4}\left(\sum_{F \sim \{E,W,T,B\}} (\phi_F^{(k)} + 1)^2 + \left(\phi_S^{(k)} + \frac{4}{3}\right)^2 - 5\left(\phi_C^{(k)} + \frac{8}{5}\right)^2 + \frac{316}{45}\right) \end{aligned} \tag{35}$$

respecto a las componentes del Jacobiano, para celdas internas queda:

$$\begin{aligned} \frac{\partial R}{\partial \phi_C^{n+1}}(\phi^{(k)}) &= \frac{h^3}{\Delta t} + 3\kappa h\left(\phi_C^{(k)} + 1\right) \\ \frac{\partial R}{\partial \phi_F^{n+1}}(\phi^{(k)}) &= -\frac{\kappa h}{2}\left(\phi_F^{(k)} + 1\right) \text{ , con } F \sim \{E,N,W,S,T,B\} \end{aligned} \tag{36}$$

para el lado oeste por ejemplo:

$$\begin{aligned} \frac{\partial R}{\partial \phi_C^{n+1}}(\phi^{(k)}) &= \frac{h^3}{\Delta t} + \frac{\kappa h}{2}\left(5\phi_C^{(k)} + 8\right) \\ \frac{\partial R}{\partial \phi_E^{n+1}}(\phi^{(k)}) &= -\frac{\kappa h}{2}\left(\phi_E^{(k)} + \frac{4}{3}\right) \\ \frac{\partial R}{\partial \phi_F^{n+1}}(\phi^{(k)}) &= -\frac{\kappa h}{2}\left(\phi_F^{(k)} + 1\right) \text{ , con } F \sim \{N,S,T,B\} \end{aligned} \tag{37}$$

6.3. Selección del caso para realizar las comparaciones

Para comparar los esquemas de discretización temporal y espacial, se eligió un tiempo final de simulación igual a 1.84s, considerando que la condición inicial alcanza un máximo de $\phi^0_{\text{máx}} = \phi(1/2, 1/2, 1/2, 0) = 2{,}5$ resultando el coeficiente de difusión igual a $a(\phi^0_{\text{máx}}) = 3{,}5$, mientras que para $t = 1{,}84$s, $\phi^{1{,}84}_{\text{máx}} = 0{,}397$ y el coeficiente de difusión vale $a(\phi) = 1{,}397$. Observando que $a(\phi) = \kappa(\phi + 1) \to 1$ al avanzar en el tiempo, por lo tanto el problema va perdiendo la no linealidad conforme avanza la simulación. En el intervalo de simulación seleccionado el problema mantiene un componente no lineal importante.

6.4. Solución numérica

En la figura 3 se muestra la solución numérica obtenida para diferentes instantes de tiempo.

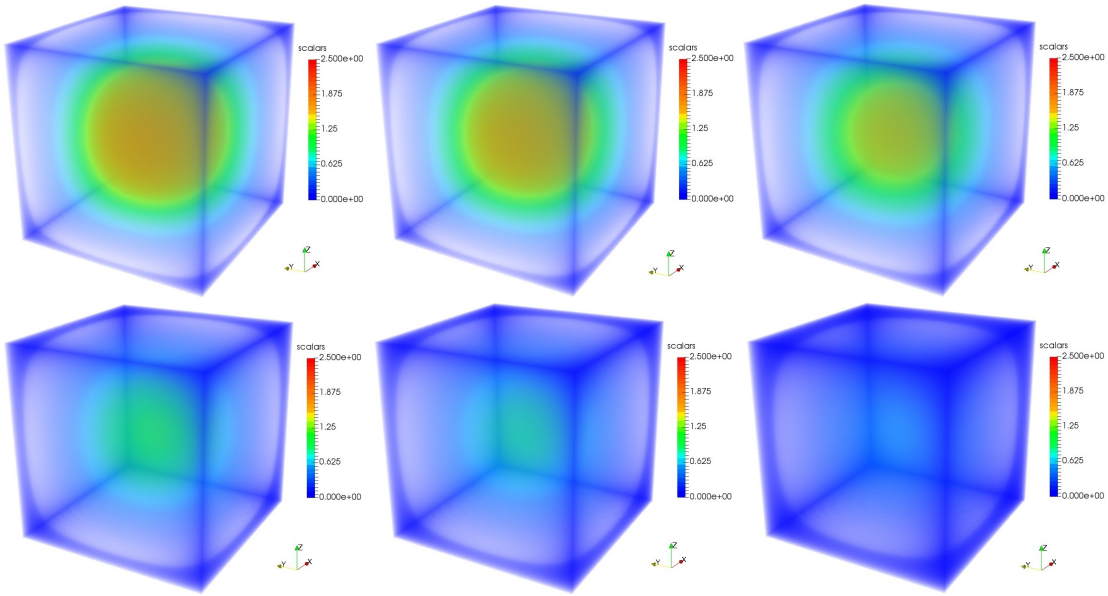

Figura 3: Solución numérica para diferentes instantes de tiempo distribuidos uniformemente desde $t = 0s$ a $t = 1{,}84$s.

7. ESTUDIO DE CONVERGENCIA NUMÉRICA

Se realizó un estudio de convergencia en malla para la discretización espacial, tanto para el esquema explícito como para el implícito y un estudio para la convergencia respecto a la discretización temporal, en este caso se realizó el análisis solamente para el esquema implícito, debido a que el paso de tiempo en el esquema explícito viene condicionado por el número de Fourier local $Fo = 6a(\phi)\Delta t/h^2$ y para valores de Δt menores, el error debido a la discretización temporal es menor al que brinda el esquema espacial usado, luego la solución numérica obtenida resulta prácticamente igual para cualquier Δt menor al impuesto por el número Fourier.

Para las simulaciones se consideraron diversas discretizaciones espaciales teniendo en cuenta la arquitectura SIMT, para esto se eligen grillas de manera que el número de celdas internas N_{int} por dirección (x, y, z) sea múltiplo de 32 (a excepción de las dos grillas mas pequeñas), como los casos de borde se tratan en kernel separados el número de celdas total por dirección resulta en $(1 + N_{int} + 1)$. Los tamaños de grillas analizadas en este trabajo variaron desde 10x10x10 a 898x898x898 (724.150.792 celdas). Cabe aclarar que para el caso implícito el tamaño máximo

que se pudo correr fue de 514x514x514 (135.796.744 celdas) debido a la limitación de tamaño en la memoria del dispositivo. En la tabla 1 se muestran las dimensiones de los casos simulados. Por último mencionamos que los resultados en las grillas que exceden las 514 celdas por dirección del caso explícito se utilizaron para evaluar el desempeño (sección 8) y que para el estudio de convergencia en espacial y temporal solo se muestran las comparaciones hasta ese tamaño de grilla.

$N_xN_yN_z$	# cell	$N_xN_yN_z$	# cell
10x10x10	1.000	322x322x322	33.386.248
18x18x18	5.832	418x418x418	73.034.632
34x34x34	39.304	450x450x450	91.125.000
66x66x66	287.496	514x514x514	135.796.744
98x98x98	941.192	610x610x610*	226.981.000*
130x130x130	2.197.000	706x706x706*	351.895.816*
194x194x194	7.301.384	770x770x770*	456.533.000*
258x258x258	17.173.512	898x898x898*	724.150.792*

Tabla 1: Dimensiones de los casos de prueba utilizados ((*) Solamente en el esquema explícito).

7.1. Convergencia del esquema temporal

Se evaluó la tasa de convergencia para el esquema temporal de CN en las grillas de 66^3, 130^3, 258^3 y 514^3, el tiempo de simulación para realizar la comparación con la solución analítica fue de $1{,}84$s, el error (ϵ) se evaluó usando el RMS (Root Mean Square) entre la solución numérica y la analítica

$$RMS = \sqrt{\left(\sum_{i=0}^{N_xN_yN_z} \left(\phi_{\mathrm{i,exac}} - \phi_{\mathrm{i,num}}\right)^2 \right) / N_xN_yN_z} \tag{38}$$

En la figura 4 se muestra los resultados obtenidos para las 4 grillas, en todos los casos el error se reduce hasta estabilizarse en un valor que viene dado por la resolución espacial de la grilla, puede apreciarse también que el esquema temporal es efectivamente de $O(\Delta t^2)$, por lo tanto se tiene una tasa de convergencia que es cuadrática al variar el paso de tiempo. En concreto puede apreciarse $\epsilon_1/\epsilon_2 \sim (\Delta t_1/\Delta t_2)^2$ y que la pendiente en escala logarítmica es aproximadamente de $\frac{(\log(\epsilon_1)-\log(\epsilon_2))}{(\log(\Delta t_1)-\log(\Delta t_2))} \sim 2$, y que por lo tanto, al reducir el paso de tiempo a la mitad el error se reduce en un factor de 4.

7.2. Convergencia del esquema espacial

Para evaluar la convergencia en malla hay que tener en cuenta algunos factores. El primero a considerar es que como la estabilidad condicional del esquema explícito está sujeta al número de Fourier, el paso de tiempo debe acompañar el tamaño del paso de malla bajo la relación $\Delta t \sim h^2$, de esta manera en general se tiene que el error del esquema temporal no afecta el orden de la solución. El segundo factor a considerar es que el esquema implícito resultó ser incondicionalmente estable y por lo tanto un paso de malla dado h_1 se pudieron correr casos para diferentes pasos de tiempo. Entonces para realizar una comparación entre ambos esquemas se eligió el paso de tiempo de manera que el error temporal esté por debajo del error espacial. Esto último permite evaluar diferentes grillas que pueden dar errores mayores al de la discretización

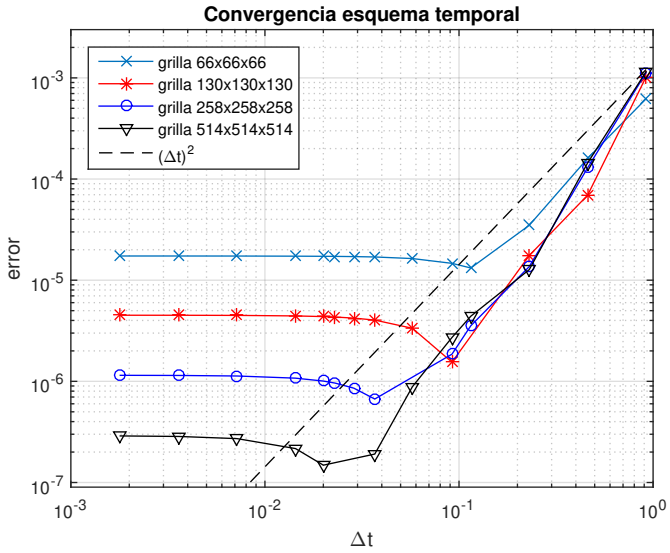

Figura 4: Estudio de la convergencia del esquema temporal para 4 dimensiones de grilla diferentes.

temporal. El segundo factor que se menciona puede verse en la figura 5 (izquierda) la cual muestra dos corridas del caso implícito usando $\Delta t = 0{,}115$s y $\Delta t = 0{,}0575$s, de manera que para mallas cada vez mas finas, el error se estabiliza en un valor dado por el orden de precisión del método CN como es de esperar. En la figura 5 (derecha) se muestra la tasa de convergencia para el método explícito e implícito juntos, puede verse que ambos tienen una tasa de convergencia cuadrática al refinar la malla.

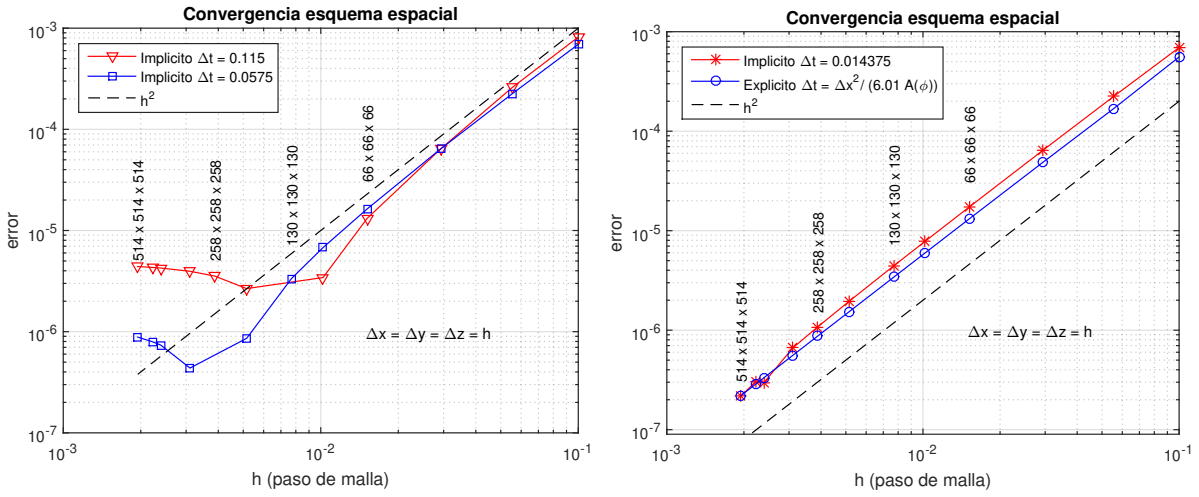

Figura 5: Convergencia del esquema espacial del método implícito para dos pasos de tiempo diferentes (izquierda) y comparación con el método explícito (derecha).

8. EVALUACION DEL DESEMPEÑO

Para medir la eficiencia del algoritmo explícito se evalúa la velocidad a la que se procesan las celdas de la siguiente forma:

$$\text{rate} = \frac{N_{steps} \times N_x \times N_y \times N_z}{t_{total} \times 10^6} \left[\frac{Mcells}{sec}\right] \tag{39}$$

siendo N_{steps} el número de pasos de tiempo realizados, t_{total} el tiempo total de la simulación en segundos. Para el caso implícito se puede estimar una *rate* equivalente considerando que para cada paso de tiempo se tienen que realizar varias iteraciones en el resolvedor CG y calcular el residuo o lado derecho para el método de NR al menos una vez por cada iteración, de manera que la eficiencia viene dada por:

$$\text{rate} = \frac{(N_{it,CG} + N_{it,NR}) \times N_x \times N_y \times N_z}{t_{total} \times 10^6} \left[\frac{Mcells}{sec}\right] \tag{40}$$

siendo $N_{it,CG}$ el número total de iteraciones CG realizadas, $N_{it,NR}$ el acumulado de iteraciones en el método de NR. Esto se eligió así debido a que la cantidad de operaciones calculadas en los kernels, tanto para el producto Ax de CG como para el cálculo del residuo $R^{(k)}$ de NR son similares las del método explícito en cada paso de tiempo, las únicas dos diferencias entre los esquemas es que en el método explícito por cada paso de tiempo y por celda se requieren dos lecturas/escrituras: (ϕ_{input}, ϕ_{output}), mientras que en el implícito se requieren tres lecturas/escrituras en la memoria global: ($\Delta\phi^{(k)}_{\text{input}}$, $\Delta\phi^{(k)}_{\text{output}}$, $\phi^{(k)}$) y ($R^{(k)}$, ϕ^n, $\phi^{(k)}$) para CG y residuo de NR respectivamente, y por otro lado en el resolvedor CG se realizan algunas operaciones de reducción (normas, producto interno, etcétera) y del tipo axpy. Sin embargo se considera que las operaciones más costosas son las dos usadas para el cálculo del rate.

Además se evaluaron los tiempos de cálculo de cada método para diferentes resoluciones de grilla, y los tiempos de cálculo para obtener diferentes errores al comparar con la analítica.

8.1. Evaluación de la tasa de procesamiento

En la tabla 2 se muestra un resumen de los resultados obtenidos para las tasas de procesamiento, mas valores se muestran en la figura 6. Puede apreciarse que el algoritmo explícito explota la potencia de cálculo de la GPU hasta que se produce una saturación en la tasa para dominios computacionales mayores a los 33 millones de celdas hasta los 135 millones donde se ve una caída del desempeño para dominios mayores manteniéndose un valor residual de 2880 Mcell/sec. Puede verse que el algoritmo implícito no aprovecha al máximo el hardware y la relación entre las tasas de procesamiento en cada método es de 1 a 3 en todas las grillas analizadas.

$N_x = N_y = N_z$	66	130	258	322	418	514	706*	898*
[Mcell]	0,33	2,2	17,2	33,4	73,0	135,8	351,9*	724,1*
rate Explícito [Mcell/sec]	521	1444	2608	2931	2940	2983	2885	2834
rate Implícito [Mcell/sec]	138	516	837	888	899	856	-	-
r_{exp}/r_{imp}	3,77	2,79	3,11	3,30	3,27	3,48	-	-

Tabla 2: Tasas de procesamiento obtenidas para los dos métodos explícito e implícito ((*) Solamente en el esquema explícito).

8.2. Errores y tiempos de calculo

Se evaluó el tiempo computacional o de cálculo requerido por celda por cada segundo de simulación. El costo computacional de la simulación es proporcional al número de celdas procesadas $N = N_x N_y N_z$ y a la cantidad de pasos de tiempo requeridos en cada segundo de simulación, es decir $\text{costo}_{comp} = O(steps/sec \times N) = O(N/\Delta t)$, considerando que en el

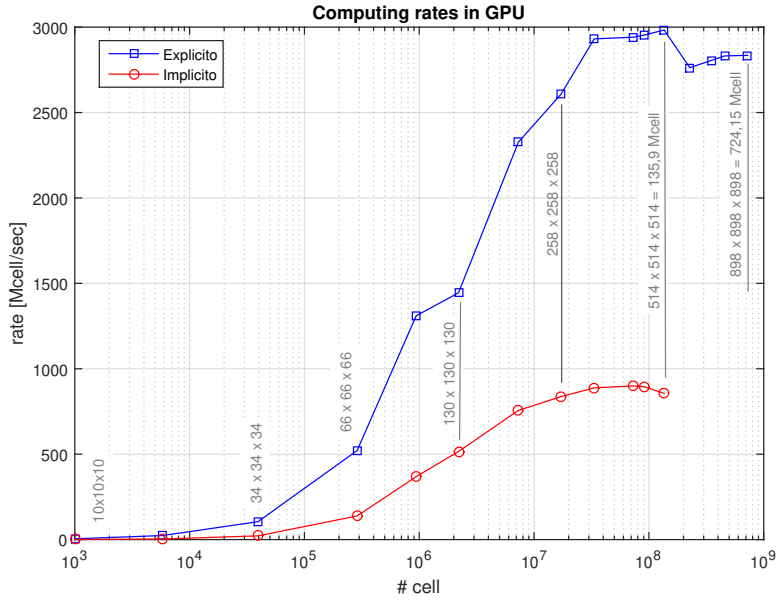

Figura 6: Eficiencia de los dos algoritmos en base a la tasa de procesamiento.

método explícito el paso de tiempo está restringido por el número de Fourier, deducimos que:

$$\Delta t = O(h^2) = O\left((1/N_x)^2\right) = O\left(\left(1/N^{1/3}\right)^2\right) = O(N^{-2/3}) \tag{41}$$

esto es:

$$\text{costo}_{comp} = O(N/N^{-2/3}) = O(N^{5/3}) \tag{42}$$

entonces se espera que el tiempo computacional se incremente a la potencia 5/3 del número total de celdas procesadas para un mismo tiempo de simulación. En la figura 7 (izquierda) se muestra que la tendencia final en el método explícito es la esperada mientras que para el implícito la pendiente es un poco menor y próxima a 4/3, esto se explica es debido a que el paso de tiempo se mantuvo constante en dichas pruebas, en particular en dichas se fijó $\Delta t = 0{,}014375$, o en otras palabras se fijó la cantidad de pasos de tiempo en cada simulación. Por otro lado en el método implícito el costo computacional por cada paso de tiempo es dado por la cantidad de iteraciones del resolvedor CG siendo esto último proporcional al la cantidad de celdas por dirección ($\sqrt[3]{N}$) como puede verse en la figura 7 (derecha). Entonces se espera un costo computacional para el método implícito igual a:

$$\text{costo}_{comp} = \underbrace{1/\Delta t}_{const} \times N_{it,CG,total} \times N = O(N^{1/3} \times N) = O(N^{4/3}) \tag{43}$$

Respecto a los errores, en la figura 8 (izquierda) se muestra que decrecen a una tasa del orden $O(N^{-2/3})$ en ambos casos (explícito e implícito) esto es equivalente a lo mostrado en el análisis de convergencia de la discretización espacial debido a que $h = N^{-1/3}$.

Finalmente en la figura 8 (derecha) se comparan los errores obtenidos en los dos esquemas para sus respectivos tiempos de cálculo, se observa que para problemas pequeños (menores a 0.3 Mcell $\approx 66^3$) es mas eficiente el método explícito y para problemas mas grandes la relación se invierte, siendo mas eficiente el método implícito, esta diferencia crece al aumentar el número de celdas llegando una relación de 1/10 en los tiempos de cálculo requeridos por el método implícito y explícito respectivamente. Esto se debe a que el $\Delta t_{\text{máx}}$ en el esquema explícito está

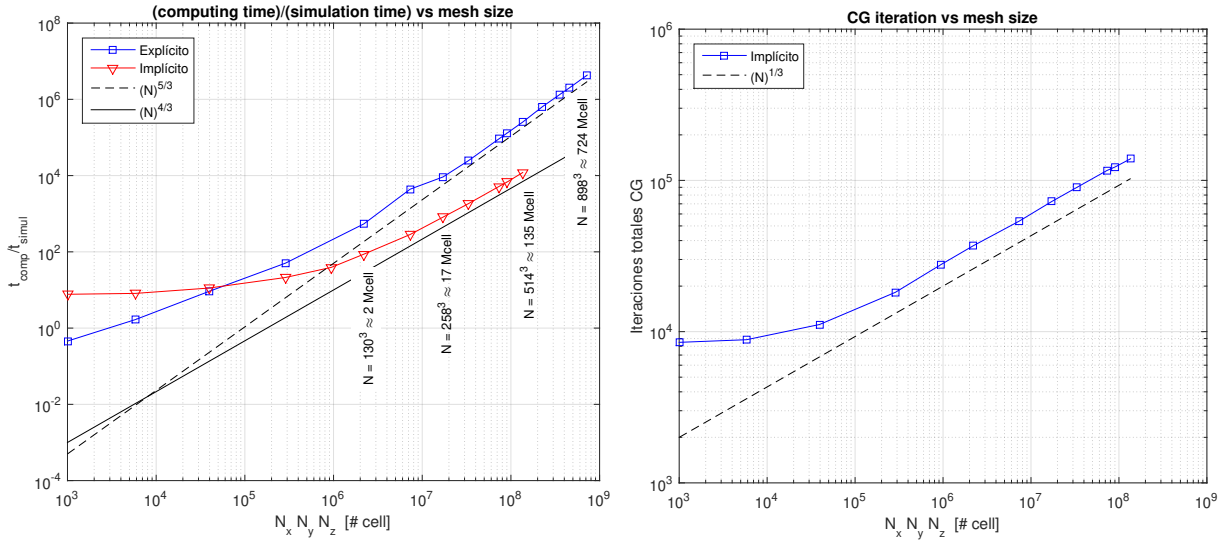

Figura 7: Tiempos de cálculo por segundo de simulación versus el tamaño de la grilla (izquierda) y número de iteraciones de CG acumuladas en la simulación (derecha)

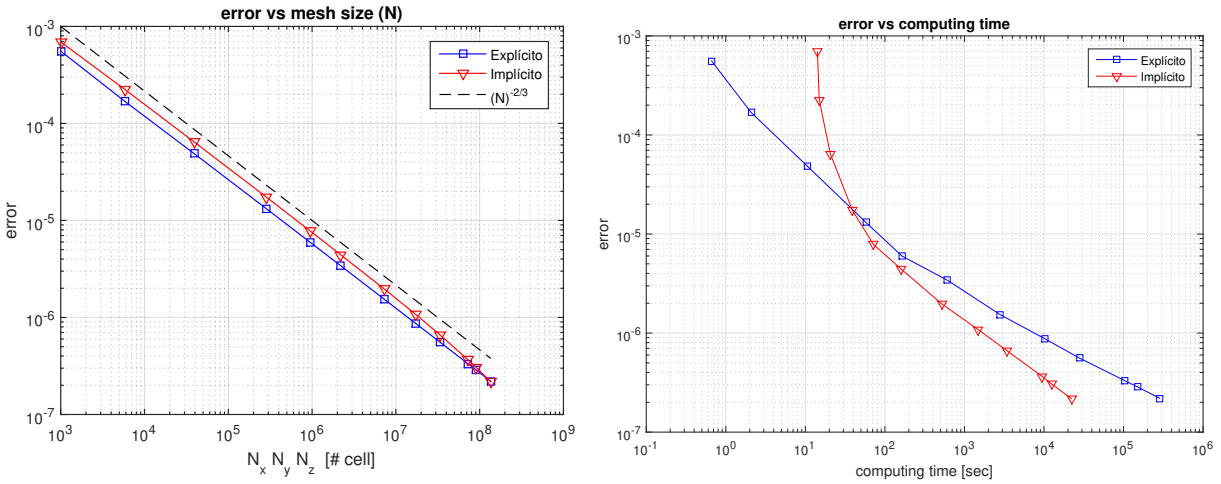

Figura 8: Error obtenido para $t = 1{,}84$s versus el tamaño de la grilla (izquierda) y comparación del error obtenido para diferentes tiempos de cálculo en el método explícito e implícito (derecha).

fuertemente condicionado con el número de Fourier y el implícito permite pasos de tiempo mayores (en este ejemplo $\Delta t_{\text{implicito}} = 0{,}014375$), además de esto, en el método implícito se pueden variar los criterios de convergencia del resolvedor CG y el lazo de NR permitiendo un fuerte incremento en la performance del algoritmo.

9. CONCLUSIONES

Se implementaron algoritmos eficientes en GPU para resolver una la ED no lineal en dominios tridimensionales de acuerdo a las características específicas del hardware y se comprobó la precisión numérica de los diferentes métodos. Los resultados muestran que ambos algoritmos (explícito e implícito) explotan al máximo las capacidades de la GPU, logrando simular problemas con muy alta resolución en el caso explícito (hasta 724 millones de celdas) sin embargo en el caso analizado se tiene una fuerte restricción en la estabilidad numérica del método requiriendo pasos de tiempo extremadamente pequeños.

Respecto al método implícito se comprobó que a pesar de tener valores de rates mas bajos respecto al algoritmo explícito, variando los criterios de convergencia en el resolvedor CG según la evolución del residuo en el método de NR, es posible obtener un buen desempeño global. En particular, se gana mucha eficiencia fijando las iteraciones máximas del resolvedor CG en cada iteración de NR en lugar de forzar la solución de CG hasta cierta precisión. En relación a esto, se probó que el orden de precisión temporal en el método de NR no cambia al realizar más iteraciones, pero como contrapartida se vió que para problemas cada vez mas no lineales, resolver una única vez el sistema de ecuaciones con el resolvedor CG (una sola iteración de NR) requiere un número considerablemente mayor de iteraciones CG. Por lo tanto se obtuvieron mucho mejores resultados resolviendo de forma inexacta el sistema en el resolvedor (fijando el número de iteraciones CG dinámicamente) y actualizando mas veces el sistema lineal a resolver (iteraciones NR) hasta obtener una precisión deseada al salir del lazo NR. Esta misma característica se encuentra en los algoritmos segregados por ejemplo el método SIMPLE (Semi Implicit Method for Pressure Linked Equation) usados para resolver las ecuaciones de Navier Stokes en los cuales no se resuelve el sistema de forma precisa en cada iteración interna del algoritmo sino que solo se fuerza la conservación de masa al final de cada paso de tiempo Ferziger y Peric (2012).

Cabe mencionar que los algoritmos implementados en este trabajo alcanzan el límite del ancho de banda de la tarjeta antes de alcanzar el límite en la capacidad de procesamiento de la GPU. Esto se debe a que las operaciones realizadas por cada CUDA Threads tiene una intensidad aritmética muy baja y se consume más tiempo en la lectura-escritura de datos requeridos para los cálculos. Lo anterior explica el porqué de los resultados obtenidos en términos del rate explícito versus implícito, ya que en el último se requiren tres veces mas operaciones de lectura-escritura en memoria global. Finalmente se destaca que la combinación de criterios y métodos en el algoritmo implícito aceleran la convergencia global del problema, requiriendo menos operaciones totales para llegar al mismo resultado que el algoritmo explícito. Los resultados del presente trabajo, muestran un comportamiento contrario a lo observado en problemas lineales, como por ejemplo el caso de la ecuación de advección difusión lineal presentada en (Bondarenco et al., 2017), en donde el algoritmo explícito en GPU resulta mucho más eficiente que los implícitos.

10. AGRADECIMIENTOS

- Universidad Tecnológica Nacional, Facultad Regional Concordia (*"Becas de formación de doctores para fortalecer las áreas de I+D+i"*, Res. 1460/15).
- Departamento del Agua, Centro Universitario Regional Litoral Norte - Sede Salto, Universidad de la República.

REFERENCIAS

Bondarenco M., Gamazo P., y Ezzatti P. A comparison of various schemes for solving the transport equation in many-core platforms. *The Journal of Supercomputing*, 73(1):469–481, 2017.

Brandao G. *Solution of the Transport Equation using Graphical Processing Units*. Tesis de Doctorado, Mechanical Engineering Department, Instituto Superior Técnico, Technical University of Lisbon, Portugal, 2009.

Brezinski C. y Wuytack L. Numerical analysis in the twentieth century. *Numerical Analysis:*

Historical Developments in the 20th Century, C. Brezinski e L. Wuytack, Editors, North–Holland, Amsterdam, páginas 1–40, 2001.

Che S., Boyer M., Meng J., Tarjan D., Sheaffer J.W., y Skadron K. A performance study of general-purpose applications on graphics processors using cuda. *Journal of parallel and distributed computing*, 68(10):1370–1380, 2008.

Cotronis Y., Konstantinidis E., y Missirlis N.M. A gpu implementation for solving the convection diffusion equation using the local modified sor method. En *Numerical Computations with GPUs*, páginas 207–221. Springer, 2014.

Ferziger J.H. y Peric M. *Computational methods for fluid dynamics*. Springer Science & Business Media, 2012.

Heimlich A., Mol A.C., y Pereira C.M. Gpu-based monte carlo simulation in neutron transport and finite differences heat equation evaluation. *Progress in Nuclear Energy*, 53(2):229–239, 2011.

Janavičius A. y Turskienė S. Nonlinear diffusion in cubic crystals. *Rom. Journ. Phys*, 61(7-8):1245–1254, 2016.

Nash S.G. *A history of scientific computing*. AMC, 1990.

Obukhovsky V.V., Kutsyk A.M., Nikonova V.V., y Ilchenko O.O. Nonlinear diffusion in multicomponent liquid solutions. *Physical Review E*, 95(2):022133, 2017.

Sellitto A., Cimmelli V.A., y Jou D. Linear and nonlinear heat-transport equations. En *Mesoscopic Theories of Heat Transport in Nanosystems*, páginas 31–51. Springer, 2016.

Shaikh B.Y. y Das S.K. Tide-induced groundwater flow properties along sloping unconfined coastal aquifer. *Environmental Processes*, 5(1):131–154, 2018.