Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ciencias Físico-Matemáticas

Mat. Luis Manuel Rivera Gutiérrez

Solución numérica de la ecuación de Richards aplicando métodos híbridos

T E S I S

que para obtener el grado de:

Licenciado en Ciencias Físico-Matemáticas

presenta:

Daniel Santana Quinteros

director de tesis:

Dr. Francisco Javier Domínguez Mota

Morelia, Michoacán Marzo, 2018

Solución numérica de la ecuación de Richards aplicando métodos híbridos

por

Daniel Santana Quinteros
dasaqui@gmail.com

Tesis presentada para obtener el grado de

Licenciado en Ciencias Físico-Matemáticas

en la

Facultad de Ciencias Físico-Matemáticas

Universidad Michoacana de San Nicolás de Hidalgo

Morelia, Michoacán. Marzo, 2018

A ti que me brindaste tu apoyo
durante todo el proceso.
A ti que soportaste mis cambios de
ánimo, y que me brindaste tu compañía.
A ti que me escuchaste en los momentos
más difíciles, y me brindaste tu apoyo.

Agradecimientos

Larga es la lista de personas a quienes podría agradecer, durante este proceso muchos me brindaron su apoyo y compañía. La escritura de una tesis es un proceso largo, complicado y en ocasiones bastante pesado, sin embargo he contado con el apoyo de grandes amigos que me escucharon, me acompañaron y me motivaron a seguir adelante durante todo el proceso.

Un gran éxito es para mi poder concluir este trabajo, durante años deseé dedicarme a las matemáticas y ahora veo logrado mi sueño. Un sueño que de alguna manera apenas se encuentra dando sus primeros pasos, sin embargo son firmes y muy esperados.

Nadie mejor que mis amigos conoce el esfuerzo y el trabajo que tuve que poner para el logro de este sueño, es a ustedes que les agradezco el tiempo, el apoyo y la comprensión que me brindaron durante todos estos meses.

A todas las personas que participaron de alguna manera en la realización de esta tesis, les quiero agradecer de una manera muy especial por compartir conmigo este anhelado sueño. Sin su apoyo difícilmente podría haber concluido esta meta. Angélica, muchas gracias por presionarme a escribir y escucharme las ocasiones que necesité hablar con alguien, Etiene y Jena, gracias por su compañía durante este proceso, Brandon, espero pronto cumplas tus metas. Leti, la mejor hermana de cubo que me pudo tocar, gracias. Doctor Francisco, gracias por su apoyo y los conocimientos que nos brindó. Y por último a mis padres que me apoyaron para lograr este sueño.

Índice general

Índice de figuras

Índice de tablas

Solución numérica de la ecuación de Richards aplicando métodos híbridos

por

Daniel Santana Quinteros

dasaqui@gmail.com

Resumen

Dentro de los problemas de modelación, la ecuación de Richards es un modelo de gran importancia en la ingeniería. Dicha ecuación es una expresión elíptico-parabólico no lineal la cual modela el flujo en medios porosos no saturados. Dada su gran importancia, diversos esquemas se han desarrollado para aproximar su solución; algunos de los esquemas mas eficientes son combinaciónes de métodos de Newton discretizando el espacio por medio de elementos finitos junto con metodos theta para la integración temporal. Sin embargo, debido a la discretización por elementos finitos se presentan oscilaciones numéricas cerca del frente de infiltración. Para lidiar con este problema, este trabajo presenta un nuevo esquema con control de paso adaptivo del tiempo basado en el método de Crank-Nicolson sobre mallas no rectangulares. El método propuesto es probado sobre un terraplén de carretera y sus resultados son comparados de manera ilustrativa con un método de elemento finito.

Palabras Clave: Diferencias finitas generalizadas, Regiones irregulares, Métodos numéricos, Ecuación de Richards, Flujo en medios porosos.

Solución numérica de la ecuación de Richards aplicando métodos híbridos

by

Daniel Santana Quinteros

dasaqui@gmail.com

Abstract

Among the flux problems the Richards equation is a mathematical model of great importance in engineering modeling. Such equation is an elliptic parabolic nonlinear expression which models flow in unsaturated porous media. Due to its importance, a number of schemes to approximate his solution has been proposed; Among the more efficient are combinations of Newtonian iterations for the spatial discretization using finite elements, and an implicit θ-method for the time integration. However, due to the finite elements formulation, numerical oscillations are present near the infiltration front. To overcome that problem, this work presents a novel generalized finite differences scheme and an adaptive stepsize Crank-Nicolson method over non rectangular meshes. The proposed method has been tested on an illustrative road embankment and the results are compared with a finite element method solution.

Key Words: Generalized finite differences, Irregular regions, Numerical methods, Richards equation, Flow in porous media.

Capítulo 1

Introducción

1.1. Ecuaciones Diferenciales Parciales

Entendemos por modelo matemático a un conjunto de ecuaciones o relaciones capaces de capturar las características esenciales de un sistema complejo [Fernández-Bonder, 2014]. Se busca que un modelo sirva para predecir o controlar la evolución de este sistema.

Los modelos clásicos, provienen principalmente de la física, química e ingenierías, sin embargo recientemente han surgido modelos provenientes de la biología, finanzas, ecología, sociología, entre otras áreas.

Para la construcción de un modelo se precisan de dos ingredientes básicos:

- Leyes generales
- Relaciones constitutivas

Las leyes generales, provienen usualmente de la mecánica del continuo, como son la conservación de la masa, energía, momento, entre otros. En cambio, las relaciones constitutivas son de naturaleza experimental y dependen de la naturaleza del fenómeno en observación [Fernández-Bonder, 2014].

Al combinar ambos ingredientes, obtenemos como resultado ecuaciones en derivadas parciales, o sistemas de ecuaciones en derivadas parciales.

Las ecuaciones diferenciales son descripciones matemáticas de las relaciones que existen entre la variables involucradas en el sistema y sus derivadas [López-Garza y Martínez-Ortiz, 2013]. Decimos que una función es solución de la ecuación diferencial cuando su comportamiento es igual a la descripción que la ecuación diferencial hace de esta función. Muchos problemas en las ciencias naturales e ingeniería se pueden formular como un sistema de ecuaciones diferenciales.

Las ecuaciones diferenciales parciales (EDP) son una extensa familia de ecuaciones diferenciales que involucran una variable dependiente así como sus derivadas con respecto a más de una variable independiente.

De manera general, una EDP lineal de segundo orden en 2 dimensiones (2 variables independientes), la cual nos es de interés debido a que la ecuación de Richards entra en esta categoría, se puede escribir como

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu + G = 0 \tag{1-1}$$

donde los coeficientes A, B, C, D, E, F y G son funciones reales definidas en alguna región $\Omega \subset \mathbb{R}^2$ y cumplen la relación $|A|+|B|+|C| > 0$, lo cual garantiza que al menos una de ellas es diferente de cero [López-Garza y Martínez-Ortiz, 2013]. Si los coeficientes son constantes reales, salvo posiblemente G, decimos que la ecuación 1-1 es una EDP lineal de segundo orden con coeficientes constantes. Una observación importante es el parecido que tiene la ecuación 1-1 con la ecuación general de las cónicas en $\mathbb{R}^2$

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \tag{1-2}$$

con A, B, C, D, E y F reales y $|A| + |B| + |C| > 0$. Como es sabido, la ecuación 1-2 tiene una propiedad intrínseca que permite una clasificación de las cónicas de acuerdo al signo del discriminante $B^2 - 4AC$. Para la ecuación 1-1 ocurre algo similar a la ecuación 1-2, lo cual permite clasificar dicha familia de ecuaciones de manera análoga a las cónicas, de esta forma las podemos clasificar como hiperbólicas, parabólicas o elípticas en función del discriminante, ya sea mayor, igual o menor que cero respectivamente.

Si bien, las ecuaciones diferenciales nos sirven para modelar los fenómenos del mundo real, y sus soluciones deben corresponder al comportamiento del fenómeno físico que se está estudiando, en la gran mayoría de los casos, estas soluciones no existen de manera analítica [Shashkov, 1995], por lo cual es necesario recurrir a métodos alternativos que nos permitan aproximar de modo aceptable las soluciones a dicho modelo. Para solventar estas dificultades existen muy variados métodos entre los cuales cabe destacar las Diferencias finitas y los volúmenes finitos, los cuales se explican brevemente en este capítulo.

1.2. Diferencias finitas

Nuestro objetivo, es aproximar la solución de las ecuaciones diferenciales, es decir, encontrar una función que satisfaga las relaciones descritas por la ecuación diferencial, acotado a alguna región espacial y temporal, además de cumplir con ciertas restricciones en la frontera de esta región. En general este es un problema complicado [Shashkov, 1995], y sólo en raras ocasiones podemos encontrar su solución como una expresión analítica.

Para alcanzar este objetivo una herramienta bastante poderosa es la expansión en series de Taylor, mediante esta podemos aproximar cualquier función analítica, de manera local alrededor del punto a mediante una serie de potencias. Usando series de Taylor podemos crear aproximaciones a las derivadas, empleando la evaluación de una función en puntos cercanos. De esta forma, podemos cambiar el problema continuo en derivadas parciales, por un sistema de ecuaciones algebraicas de gran tamaño pero que podemos resolver.

Sea $u(x)$ una función suave en la vecindad de un punto $\bar{x}$, es decir, alrededor de este punto existen todas sus derivadas. Nuestro objetivo es aproximar el valor de su primer derivada $u'(\bar{x})$ por medio de una aproximación en diferencias finitas utilizando un número finito de puntos alrededor de $\bar{x}$. Si escribimos $u(\bar{x}+h)$ usando los primeros dos términos de su serie de Taylor,

y empleando la notación O-grande[1] acotamos el error debido a esta aproximación, tenemos

$$u(\bar{x}+h) = u(\bar{x}) + hu'(\bar{x}) + \mathcal{O}\left(h^2\right) \tag{1-3}$$

donde $\mathcal{O}\left(h^2\right)$ nos indica que el error de nuestra aproximación crece (o decrece) tan rápido como lo haga h^2, en este caso decimos que nuestra aproximación es de orden 2; de aquí podemos reescribir para obtener la siguiente aproximación en diferencias para la primer derivada de u

$$u'(\bar{x}) = \frac{u(\bar{x}+h) - u(\bar{x})}{h} + \mathcal{O}(h) \tag{1-4}$$

en esta ecuación a la cual conocemos como diferencias hacia adelante, se observa que cambió el orden de la aproximación al dividirla entre h, resultando en una aproximación de primer orden. Cabe observar que las aproximaciones a una derivada no son únicas, por ejemplo si cambiamos el signo de h podemos obtener una expresión alterna de la forma

$$u(\bar{x}-h) = u(\bar{x}) - hu'(\bar{x}) + \mathcal{O}\left(h^2\right) \tag{1-5}$$

que reescrita nos da como resultado

$$u'(\bar{x}) = \frac{u(\bar{x}) - u(\bar{x}-h)}{h} + \mathcal{O}(h), \tag{1-6}$$

expresión que conocemos como diferencias hacia atrás.

De manera similar a las ecuaciones 1-4 y 1-6 es posible crear aproximaciones para derivadas de mayor grado, por ejemplo, si deseamos calcular una segunda derivada podríamos emplear las

[1] La notación O-grande se emplea para acotar de manera asintótica el crecimiento de un término, decimos que algo ($g(x)$) es O-grande de $f(x)$ ($\mathcal{O}(f)$) cuando existe k tal que $g(x) < kf(x)$

ecuaciones 1-3 y 1-5, expandiendo con tres términos las respectivas series y sumando obtenemos

$$\begin{aligned} u(\bar{x}+h) &= u(\bar{x}) + hu'(\bar{x}) + h^2u''(\bar{x}) + \mathcal{O}\left(h^3\right) \\ u(\bar{x}-h) &= u(\bar{x}) - hu'(\bar{x}) + h^2u''(\bar{x}) + \mathcal{O}\left(h^3\right) \end{aligned}$$

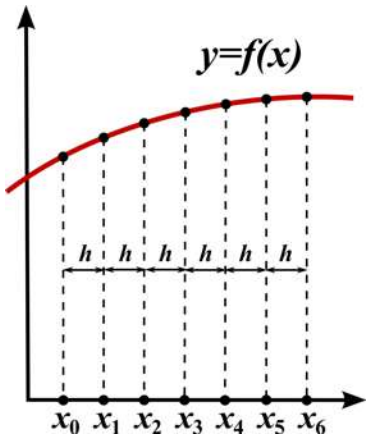
$$u(\bar{x}+h) + u(\bar{x}-h) = 2u(\bar{x}) + 2h^2u''(\bar{x}) + \mathcal{O}\left(h^3\right) \tag{1-7}$$

que al despejar nos lleva a la ecuación

$$u''(\bar{x}) = \frac{u(\bar{x}-h) - 2u(\bar{x}) + u(\bar{x}+h)}{2h} + \mathcal{O}\left(h^2\right), \tag{1-8}$$

expresión que conocemos con el nombre de diferencias centradas.

De esta forma, cuando construimos un método en diferencias finitas, uno elige la región que desea estudiar y la parte en segmentos uniformes, a cada extremo (llamado nodo) le asociamos la ecuación que le rige y sustituimos todas las derivadas por alguna aproximación en diferencias finitas. El resultado es un sistema de ecuaciones con tantos renglones como nodos, el cual se puede reescribir como una matriz rala y resolver por algún método adecuado.



1.3. Diferencias finitas generalizadas

Cuando empleamos diferencias finitas para estudiar fenómenos en 2 dimensiones, resulta natural trabajar con regiones rectangulares las cuales nos permiten partir el dominio en rectángulos uniformes. La sencillez que genera esta partición radica en la uniformidad que genera, ya que como las distancias hacia los nodos vecinos no cambian para todo el interior, los mismos tienen todos la misma ecuación en diferencias finitas.

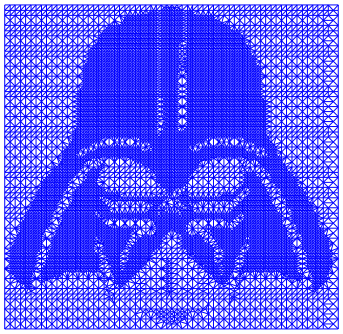


Figura 1-1: Ejemplo de malla no rectangular

En algunas ocasiones, no resulta práctico trabajar con rectángulos, ya sea por que se desea modelar una región no rectangular o por que requerimos aumentar la densidad de puntos en alguna zona. En estos casos, resulta impráctico calcular a mano las aproximaciones que empleamos para cada punto del espacio que estamos modelando. Dado que se ha perdido la uniformidad que teníamos, para calcular las nuevas aproximaciones a nuestras ecuaciones es necesario desarrollar un algoritmo que permita realizar automáticamente los cálculos. Es aquí donde surgen las diferencias finitas generalizadas (DFG).

Para comprender este método, primero debemos recordar la forma general de una EDP homogénea de segundo grado en 2 dimensiones, aquí introducimos el operador L que aplicado a la variable u se define como

$$Lu = A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu$$

nuestra intención es, dado un punto p_0 y q puntos cercanos a él, desarrollar una aproximación L_0 que se comporte de manera muy similar al operador original L. Para fijar ideas podemos observar

Figura 1-2: Distribución arbitraria de puntos

la figura 1-2, para aproximar el operador L usando los q vecinos del punto p_0 emplearemos series de Taylor de tal forma que podamos expresar L_0 como una combinación lineal u evaluado en los puntos $p_0, p_1, \ldots, p_q$, es decir

$$L_0 u = \sum_{i=0}^{q} \Gamma_i u(p_i). \tag{1-9}$$

Para que L_0 sea una buena aproximación debemos imponer algunas condiciones, en nuestro caso buscaremos que el error local de truncamiento se aproxime a cero conforme los puntos p_i se acercan al punto $p0$, es decir

$$\left[A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu\right]_{p_0} - \sum_{i=0}^{q} \Gamma_i u(p_i) \to 0, \tag{1-10}$$

debe cumplirse. Si empleamos series de Taylor de u hasta segundo orden y reagrupamos los términos obtenidos obtenemos la expresión

$$\begin{aligned}\left[A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu\right]_{p_0} - \sum_{i=0}^{q} \Gamma_i u(p_i) \quad = \\ \left(A|_{p_0} - \sum_{i=0}^{q} \frac{\Gamma_i \Delta x_i^2}{2}\right) \frac{\partial^2 u}{\partial x^2}\bigg|_{p_0} + \left(B|_{p_0} - \sum_{i=0}^{q} \frac{\Gamma_i \Delta x_i y_i}{2}\right) \frac{\partial^2 u}{\partial xy}\bigg|_{p_0} + \left(C|_{p_0} - \sum_{i=0}^{q} \frac{\Gamma_i \Delta y_i^2}{2}\right) \frac{\partial^2 u}{\partial y^2}\bigg|_{p_0} \quad + \\ \left(D|_{p_0} - \sum_{i=0}^{q} \Gamma_i \Delta x\right) \frac{\partial u}{\partial x}\bigg|_{p_0} + \left(E|_{p_0} - \sum_{i=0}^{q} \Gamma_i \Delta y\right) \frac{\partial u}{\partial y}\bigg|_{p_0} \quad + \\ \left(F|_{p_0} - \sum_{i=0}^{q} \Gamma_i\right) u|_{p_0} + \mathcal{O}\left(max\left(\Delta x_i, \Delta y_i\right)^3\right)\end{aligned} \tag{1-11}$$

donde los términos Δx_i y Δy_i representan las distancias en x y y respectivamente entre los

puntos p_i y p_0.

Nos interesa entonces, que la ecuación 1-11 cumpla las condiciones descritas por la ecuación 1-10, de esta forma es necesario que cada termino entre paréntesis se anule, por lo cual obtenemos el sistema de ecuaciones

$$\begin{aligned}
A|_{p_0} \quad & -\sum_{i=0}^{q} \frac{\Gamma_i \Delta x_i^2}{2} && = 0 \\
B|_{p_0} \quad & -\sum_{i=0}^{q} \frac{\Gamma_i \Delta x_i y_i}{2} && = 0 \\
C|_{p_0} \quad & -\sum_{i=0}^{q} \frac{\Gamma_i \Delta y_i^2}{2} && = 0 \\
D|_{p_0} \quad & -\sum_{i=0}^{q} \Gamma_i \Delta x && = 0 \\
E|_{p_0} \quad & -\sum_{i=0}^{q} \Gamma_i \Delta y && = 0 \\
F|_{p_0} \quad & -\sum_{i=0}^{q} \Gamma_i && = 0,
\end{aligned} \tag{1-12}$$

Si encontramos los valores gama que cumplen con este sistema, entonces el error entre nuestra aproximación y el operador L quedará acotada superiormente por $\mathcal{O}\left(max\left(\Delta x_i, \Delta y_i\right)^3\right)$, el cual disminuye conforme los puntos vecinos se aproximan a p_0.

Las condiciones de convergencia impuestas por el sistema 1-12 pueden ser reescritas de forma matricial por medio de

$$\begin{bmatrix}
1 & 1 & \cdots & 1 \\
0 & \Delta x_1 & \cdots & \Delta x_q \\
0 & \Delta y_1 & \cdots & \Delta y_q \\
0 & (\Delta x_1)^2 & \cdots & (\Delta x_q)^2 \\
0 & \Delta x_1 \Delta y_1 & \cdots & \Delta x_q \Delta y_q \\
0 & (\Delta y_1)^2 & \cdots & (\Delta y_q)^2
\end{bmatrix}
\begin{bmatrix}
\Gamma_0 \\ \Gamma_1 \\ . \\ . \\ . \\ \Gamma_q
\end{bmatrix}
=
\begin{bmatrix}
F|_{p_0} \\ D|_{p_0} \\ E|_{p_0} \\ 2A|_{p_0} \\ B|_{p_0} \\ 2C|_{p_0}
\end{bmatrix}. \tag{1-13}$$

En general, este es un problema subcondicionado, ya que cuenta con 6 ecuaciones y $q+1$ incógnitas, dando lugar a varios grados de libertas, por lo cual, de existir, su solución no será única. Para abordar la solución de este sistema existen diferentes estrategias, uno de los primeros

pasos que podemos tomar consiste en separar la primer ecuación para obtener la condición

$$\Gamma_0 = -\sum_{i=1}^{q} \Gamma_i \tag{1-14}$$

junto con el sistema

$$\begin{bmatrix} \Delta x_1 & \cdots & \Delta x_q \\ \Delta y_1 & \cdots & \Delta y_q \\ (\Delta x_1)^2 & \cdots & (\Delta x_q)^2 \\ \Delta x_1 \Delta y_1 & \cdots & \Delta x_q \Delta y_q \\ (\Delta y_1)^2 & \cdots & (\Delta y_q)^2 \end{bmatrix} \begin{bmatrix} \Gamma_1 \\ . \\ . \\ . \\ \Gamma_q \end{bmatrix} = \begin{bmatrix} D|_{p_0} \\ E|_{p_0} \\ 2A|_{p_0} \\ B|_{p_0} \\ 2C|_{p_0} \end{bmatrix}. \tag{1-15}$$

Para resolver este sistema resultante, existen distintas estrategias, entre las cuales se encuentran emplear la pseudo inversa de Penrose o resolverlo como un problema de optimización para encontrar el vector Γ que minimice la ecuación

$$A\Gamma = b \tag{1-16}$$

1.4. Volúmenes finitos

El método de diferencias finitas es un método bastante útil para la aproximación de soluciones a ecuaciones diferenciales, sin embargo, cuenta con inconvenientes para aproximar la solución para algunas familias de ecuaciones, en particular, presenta dificultades para resolver problemas con discontinuidades. Para enfrentar estas dificultades, una alternativa consiste en cambiar el enfoque del problema buscando la solución débil al mismo, la cual se encuentra al sustituir la ecuación diferencial por su versión integral.

Un ejemplo en el que podemos estudiar sus soluciones débiles para fijar ideas, es la ecuación de difusión de calor, la cual en una dimensión está dada por la ecuación $\frac{\partial u}{\partial t} - k\frac{\partial^2 u}{\partial x^2} = 0$ la cual

podemos integrar y reescribir por medio de

$$\begin{aligned}\int_0^T \int_a^b \left(\frac{\partial u}{\partial t} - k\frac{\partial^2 u}{\partial x^2}\right) dxdt &= \\ \int_a^b \int_0^T \frac{\partial u}{\partial t} dtdx - \int_0^T \int_a^b k\frac{\partial^2 u}{\partial x^2} dxdt &= \\ \int_a^b \left(u|_{t=0}^T\right) dx - \int_0^T \left(k\frac{\partial u}{\partial x}\Big|_{x=a}^b\right) dt &= 0, \end{aligned} \tag{1-17}$$

donde podemos observar que el cambio de temperatura en una región durante un periodo de tiempo, debe ser igual a lo que atraviesa sus fronteras durante dicho periodo de tiempo.

Para abordar este nuevo enfoque, pasaremos de estudiar puntualmente el comportamiento de las ecuaciones, para estudiarlas por medio de volúmenes de control, los cuales representan el comportamiento promedio de la ecuación sobre toda su extensión. De esta forma, nuestra interpretación de la distribución de calor como se observa en la figura 1-3 pasa de ser una función continua a trozos a convertirse en una colección de funciones constantes definidas en el mismo intervalo que la función original, y resolver dicha ecuación diferencial se traduce en resolver el problema de Euler en cada una de las discontinuidades.

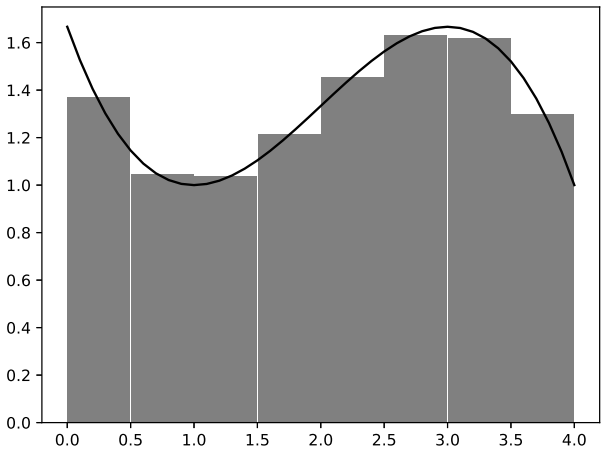


Figura 1-3: Volúmenes finitos.

De esta forma, para el intervalo $\overline{ab}$, tenemos la ecuación 1-17 que se puede reescribir como

$$\int_a^b \left(u|_{t=0}^T \right) dx = \int_0^T \left(k \frac{\partial u}{\partial x}\bigg|_{x=a}^b \right) dt, \tag{1-18}$$

donde, empleando series de Taylor, podemos expresar las derivadas por medio de

$$\begin{aligned} \int_a^b \left(u|_{t=0}^T \right) dx &= \int_0^T \left(k_b \left.\frac{\partial u}{\partial x}\right|_b - k_a \left.\frac{\partial u}{\partial x}\right|_a \right) dt \\ \int_a^b \left(u^T - u^0 \right) dx &= \int_0^T \left(k_b \frac{u_{b+\frac{1}{2}} - u_{b-\frac{1}{2}}}{\Delta l_b} - k_a \frac{u_{a+\frac{1}{2}} - u_{a-\frac{1}{2}}}{\Delta l_a} \right) dt, \end{aligned} \tag{1-19}$$

tomando en cuenta que u representa el valor promedio del intervalo y si T es suficientemente pequeño para que u no presente una variación significativa, la ecuación anterior se puede reescribir como

$$\begin{aligned} \left(u^T - u^0 \right) \int_a^b dx &= \left(k_b \frac{u_{b+\frac{1}{2}} - u_{b-\frac{1}{2}}}{\Delta l_b} - k_a \frac{u_{a+\frac{1}{2}} - u_{a-\frac{1}{2}}}{\Delta l_a} \right) \int_0^T dt \\ \left(u^T - u^0 \right) \Delta l &= \left(k_b \frac{u_{b+\frac{1}{2}} - u_{b-\frac{1}{2}}}{\Delta l_b} - k_a \frac{u_{a+\frac{1}{2}} - u_{a-\frac{1}{2}}}{\Delta l_a} \right) T \\ u^T &= u_0 + \frac{T}{\Delta l} \left(k_b \frac{u_{b+\frac{1}{2}} - u_{b-\frac{1}{2}}}{\Delta l_b} - k_a \frac{u_{a+\frac{1}{2}} - u_{a-\frac{1}{2}}}{\Delta l_a} \right). \end{aligned} \tag{1-20}$$

Capítulo 2

Problema de flujo

2.1. Antecedentes: Ecuación de Richards

La construcción de carreteras involucra el desplazamiento y ubicación de materiales porosos para nivelar el terreno y dar sustento adecuado a la capa de rodamiento. La adecuada compactación de estos materiales, reduce los vacíos del mismo, con lo cual aumenta el poder de soporte del material , y se reduce la cantidad de agua que puede penetrar en este; la cual puede afectar la resistencia al corte y provocar cambios volumétricos perjudiciales [Crespo, 2007].

Llamamos terraplén a esta estructura de tierra empleada para nivelar el terreno. Una vez que la estructura ha sido construida, ésta se encuentra sometida a condiciones climáticas muy variables que pueden producir cambios en la humedad del terreno. Si bien, la compactación mejora la calidad de los suelos, los cambios en el contenido de humedad pueden disminuir la resistencia y modificar estructura de los terraplenes [Alonso *et al.*, 1990; Chávez y Alonso, 2003; Mualem, 1976].

Actualmente, en el diseño de pavimentos se emplean diversas metodologías que toman en cuenta el efecto del contenido volumétrico de agua sobre las propiedades mecánicas de los suelos empleados [AASHTO, 2008]. Los terraplenes son la base de los pavimentos y también están sujetos a infiltración de agua debido a las condiciones climáticas, y en consecuencia a deformaciones y cambios en la resistencia.

La hidrodinámica de los procesos de infiltración, es descrita por la ecuación de Richards [Richards, 1931], una ecuación no lineal que requiere un cuidadoso tratamiento numérico para aproximar su solución. Para llegar a su deducción, es necesario partir de algunos conceptos previos. Para describir flujos en suelos saturados y no saturados, la carga hidráulica es empleada como medida de la energía del agua en los suelos. La carga hidráulica h se puede expresar como

$$h = y + \frac{u_w}{\gamma_w} \tag{2-1}$$

donde, y es la carga gravitacional, u_w es la presión de los poros de agua, y γ_w es el peso unitario del agua. La ecuación 2-1, alternativamente puede ser reescrita como

$$u_w = (h - y)\,\gamma_w \tag{2-2}$$

.

La succión es considerada una variable de estado la cual describe el comportamiento de los suelos no saturados [Fredlund *et al.*, 2012]. Algunas propiedades de los suelos como el contenido volumétrico de agua y la permeabilidad están definidas como función de la succión, la cual podemos definir como

$$s = u_a - u_w \tag{2-3}$$

donde u_a es la presión en los poros de aire y u_w es la presión en los poros de agua.

La descripción clásica para el flujo en suelos saturados está dada por la ley de Darcy, la cual también es válida para medios no saturados. Esta ley indica que la taza de flujo o velocidad a la que fluye un líquido a través de un suelo es directamente proporcional al gradiente de la carga hidráulica, dicha ley se puede escribir como

$$v_\xi = -k_\xi(s)\frac{dh}{d\xi} \tag{2-4}$$

donde v_ξ es la taza de flujo a lo largo del eje ξ, $k_\xi(s)$ es el coeficiente de permeabilidad, y $\frac{dh}{d\xi}$ es el gradiente de la carga hidráulica a lo largo del eje ξ.

Empleando la ecuación de continuidad podemos escribir la ecuación de flujo como

$$\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}\right) = \frac{\partial \theta_w}{\partial t}, \tag{2-5}$$

donde θ_w es el contenido volumétrico de agua, el cual se obtiene de la curva característica. Reemplazando la ley de Darcy en la ecuación de continuidad 2-5 y empleando la regla de la cadena tenemos

$$\frac{\partial}{\partial x}\left(-k_x\frac{\partial h}{\partial x}\right) + \frac{\partial}{\partial y}\left(-k_y\frac{\partial h}{\partial y}\right) + \frac{\partial}{\partial z}\left(-k_z\frac{\partial h}{\partial z}\right) + = \frac{\partial \theta_w}{\partial s}\frac{\partial s}{\partial u_w}\frac{\partial u_w}{\partial t}. \tag{2-6}$$

Empleando la definición de carga hidráulica 2-3, podemos obtener la derivada de la presión de poro u_w con respecto al tiempo

$$\frac{\partial u_w}{\partial t} = \gamma_w \frac{\partial h}{\partial t}, \tag{2-7}$$

calculando la derivada de la succión con respecto a la presión de poro

$$\frac{\partial s}{\partial u_w} = -1, \tag{2-8}$$

finalmente sustituyendo las ecuaciones 2-1, 2-7 y 2-8 en la ecuación 2-6 y reescribiendo obtenemos

$$k_x\frac{\partial^2 u_w}{\partial x^2} + k_y\frac{\partial^2 u_w}{\partial y^2} + k_z\frac{\partial^2 u_w}{\partial z^2} + \frac{\partial k_x}{\partial x}\frac{\partial u_w}{\partial x} + \frac{\partial k_y}{\partial y}\frac{\partial u_w}{\partial y} + \frac{\partial k_z}{\partial z}\frac{\partial u_w}{\partial z} + \gamma_w\frac{\partial k_y}{\partial y} = m_w\frac{\partial u_w}{\partial t}, \tag{2-9}$$

donde $m_w = \frac{\partial \theta_w}{\partial s}$ y le llamaremos módulo de almacenamiento de agua. Esta ecuación 2-9 es la ecuación de Richards.

Una expresión usual para la permeabilidad está dada por el modelo de Van Genuchten-Mualem [Mualem, 1976; Genuchten, 1980], el cual lo podemos expresar por medio de

$$k(\theta_n) = k_s\sqrt{\theta_n}\left(1 - (1 - \theta_n^{\frac{1}{m}})^m\right)^2, \tag{2-10}$$

donde $\theta_n = \frac{\theta_w - \theta_r}{\theta_s - \theta_r}$ es contenido volumétrico de agua normalizado, θ_r es el contenido volumétrico residual de agua, θ_s es el contenido saturado volumétrico de agua y m es un parámetro de ajuste.

Dado que la permeabilidad está definida en función del contenido volumétrico de agua, es necesario definir las curvas de retención de agua, una de estas propuesta por Van Genuchten [Genuchten, 1980]

$$\theta_n(s) = \frac{1}{((\alpha s)^n + 1)^m}, \tag{2-11}$$

donde α es un parámetro de ajuste relacionado al punto de entrada de aire en el suelo, m y n son parámetros de ajuste y $m = 1 - \frac{1}{n}$. De manera alternativa podemos emplear la ecuación de Fredlund y Xing

$$\theta_n(s) = \frac{1}{(ln(e + (\frac{s}{a_f})^{n_f}))^{m_f}}, \tag{2-12}$$

donde s el negativo de la presión de poro, a_f es un parámetro relativo a la presión de entrada de aire, n_f y m_f son parámetros de ajuste.

2.2. Generación numérica de mallas

La generación de mallas, es una herramienta útil para simular fenómenos y procesos en espacios físicos. Esta, es tanto un arte como una ciencia, donde las matemáticas proveen un fundamento esencial para convertir la creación de mallas en un proceso automatizado. Sin embargo, también es un arte debido a que no existen leyes por descubrir que rijan a la creación de mallas. El proceso de creación de mallas no es único.

El proceso de generación de mallas en general, parte primero de definir la geometría de los contornos. Se debe generar una adecuada distribución de puntos sobre las curvas que definen las fronteras, para posteriormente generar las distribuciones de puntos del interior.

La solución numérica de EDPs, involucra primero la discretización de las ecuaciones para remplazar las ecuaciones diferenciales continuas por un sistema de ecuaciones algebraicas simultaneas. Existen diferentes puntos de bifurcación en el proceso de construcción de la solución, uno de los primeros consiste en la representación de la ecuación diferencial en puntos discretos

o celdas discretas, las cuales pueden estar conectadas entre si de maneras muy variadas.

En el primer caso, las derivadas son representadas sobre puntos por expresiones algebraicas obtenidas empleando expansiones en series de Taylor de la solución sobre varios vecinos del punto a evaluar. Esto se traduce en representar la solución por medio de polinomios entre los puntos. Aproximar de esta manera puede ser impreciso si la solución varía significativamente entre los puntos involucrados, un remedio a este problema, puede ser emplear más puntos en esta región de forma que el espacio entre ellos se reduzca. Por otro lado, resolver así puede resultar muy costoso computacionalmente debido al aumento del número de puntos donde la ecuación debe ser evaluada.

El aumento del costo computacional puede ser más notorio si el número de puntos está distribuido uniformemente y las grandes variaciones de la solución ocurren sobre regiones muy dispersas, dado que gran cantidad de puntos son desperdiciados en regiones de poca variación. Una alternativa, por supuesto, es distribuir los puntos de manera no uniforme, dando mayor importancia a las regiones más complicadas.

Los métodos de generación de mallas basados en interpolación han sido extensivamente estudiados para tomar ventaja de sus dos principales fortalezas: rápido calculo de las mallas comparado con los métodos de ecuaciones diferenciales parciales, y control directo sobre la ubicación de los puntos de las mallas. Por otro lado, estas ventajas pueden ser opacadas debido a que estos métodos podrían no generar mallas suaves; en particular, discontinuidades en la pendiente de las fronteras se puede propagar fácilmente a los nodos del interior. El método estándar para la generación de mallas por interpolación es conocido como interpolación transfinita [Gordon y Hall, 1973].

Todo problema de generación de mallas comienza con la descripción de la región de frontera, la cual consta de cuatro ecuaciones paramétricas

$$\begin{aligned} &\vec{x_b}(\xi), \vec{x_t}(\xi), 0 \leq \xi \leq 1, \\ &\vec{x_r}(\eta), \vec{x_l}(\eta), 0 \leq \eta \leq 1, \end{aligned} \tag{2-13}$$

donde los subíndices b, t, l y r indican que dicha ecuación pertenece a la frontera inferior, superior, izquierda y derecha respectivamente.

Empleamos los polinomios de Lagrange de primer grado $1-\xi$, ξ, $1-\eta$, y η como funciones de combinación en la fórmula básica de interpolación transfinita. Dicha fórmula es

$$\begin{aligned}\vec{x}(\xi,\eta) &= (1-\eta)\vec{x_b}(\xi)+\eta\vec{x_t}+(1-\xi)\vec{x_l}(\eta)+\xi\vec{x_r}(\eta)\\ &- [\xi\eta\vec{x_t}(1)+\xi(1-\eta)\vec{x_b}(1)]\\ &- [\eta(q-\xi)\vec{x_t}(0)+(1-\xi)(1-\eta)\vec{x_b}(0)]\,. \end{aligned} \tag{2-14}$$

2.3. Integración en el tiempo

Existen distintos esquemas para integrar en el tiempo, cada uno con sus propias ventajas y desventajas, unos de los más conocidos es el método de Euler, un método de integración numérica conocido así en honor de Leonhard Euler. El método de Euler propuesto entre 1768 y 1770, es el más simple de todos los métodos numéricos de integración y es útil para resolver problemas de valor inicial de la forma

$$PVI = \begin{cases} \frac{\partial u}{\partial t} = F(u(t),t) \\ u(0) = u_0. \end{cases} \tag{2-15}$$

La idea fundamental de este método, está basada en un principio muy simple. Supongamos que una partícula está moviéndose de forma tal que en el tiempo t_0 su posición es igual a u_0, también conocemos su velocidad en este punto, la cual es v_0. El principio del que parte este método es que, en un corto periodo de tiempo, suficientemente corto para que la velocidad no varíe significativamente con respecto de v_0, el cambio de posición, es aproximadamente igual al cambio en el tiempo multiplicado por la velocidad de la partícula (v_0).

Si el desplazamiento de la partícula es gobernado por una ecuación diferencial, conocemos el valor de v_0 por medio de una función que depende de la posición u_0 y del tiempo t_0 de forma que $v_0 = F(u(t_0), t_0)$. De aquí que, conocidos los valores t_0 y u_0, es posible aproximar la

solución en el tiempo t_1, asumiendo que está muy cerca de t_0, por medio de

$$u_1 = u_0 + (t_1 - t_0)F(u_0, t_0), \tag{2-16}$$

el cual podemos calcular por medio de los valores conocidos t_0, u_0 y t_1. Asumiendo que podemos calcular con suficiente precisión el valor de v_1 a partir de los valores u_1 y t_1, entonces podemos dar un segundo paso para aproximar la solución en t_2 por medio de la formula

$$u_2 = u_1 + (t_2 - t_1)F(u_1, t_1). \tag{2-17}$$

Siguiendo esta secuencia, es posible obtener una serie de aproximaciones $u_1, u_2, u_3, ...$ a la solución de la ecuación en los tiempos $t_1, t_2, t_3, ...$, la cual eventualmente podría dejar de ser aceptablemente precisa conforme t va alejándose del punto inicial.

Evidentemente, la interpretación del método de Euler es mucho más amplia que el movimiento de una sola partícula, desplazándose a lo largo del tiempo sobre una línea. Incluso, la variable independiente, la cual denotamos por t, no siempre tiene que representar al tiempo. La variable dependiente u no necesariamente representa distancia así como tampoco tiene por que ser un escalar. Si u es un vector, entonces, puede ser interpretado como una colección de componentes escalares $u_1, u_2, u_3, ...u_N$. De esta forma representar el vector como

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_N(t) \end{bmatrix}. \tag{2-18}$$

La ecuación diferencial, y los datos de valores iniciales, los cuales en conjunto determinan los valores de las componentes de u conforme el tiempo varía, pueden ser escritos de forma

$$u'(t) = f(t, u(t)), \quad u(t_0) = u_0. \tag{2-19}$$

En el caso vectorial, la función f está definida en $\mathbb{R} \times \mathbb{R}^N \to \mathbb{R}^N$.

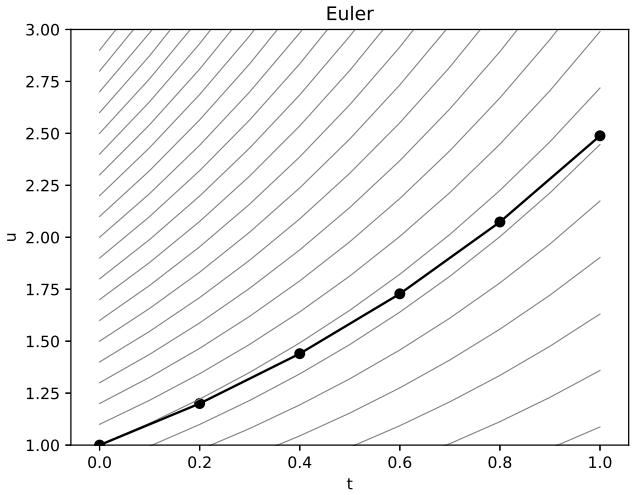


Figura 2-1: Aproximación usando Euler.

Para fijar ideas al respecto del tema, tomaremos como ejemplo el problema de valor inicial

$$\frac{\partial u}{\partial t} = u, \quad u(0) = 1. \tag{2-20}$$

En el tiempo, para mostrar el comportamiento de la solución, tomaremos cinco pasos a igual distancia de 0.2, contra las soluciones a este problema con distintas condiciones iniciales. La solución general a este problema está dada por

$$u = e^{t+C}, \tag{2-21}$$

con C una constante arbitraria, las soluciones exactas, así como la aproximada se muestran en la figura 2-1.

En este caso, emplear el método de Euler para aproximar la solución de la ecuación diferencial, se reduce a resolver la igualdad

$$u_{n+1} = u_n + (t_{n+1} - t_n)F(t_n), \tag{2-22}$$

que sustituyendo el valor de F en el tiempo t_n y empleando la notación Δ_n para representar la diferencia $t_{n+1} - t_n$ nos lleva a la ecuación

$$u_{n+1} = (1 + \Delta_n)u_n, \tag{2-23}$$

sustituyendo los valores nos lleva a la serie $1.0, 1.2, 1.44, 1.73, 2.07, \ldots$, valores que se muestran en la figura 2-1.

El método de Euler, no es el único esquema para integración numérica, sin embargo, es uno de los más sencillos de deducir, junto con él, podemos desarrollar métodos implícitos; un método implícito, es aquel que para calcular estados futuros, requiere resolver sistemas de ecuaciones que involucran tanto información del estado actual como valores futuros, o por calcular. Retomando el ejemplo de la partícula que se desplaza partiendo de un punto u_0 en t_0, si elegimos un tiempo posterior t_1 como en el método de Euler, es decir, suficientemente cercano a t_0 para que la velocidad no varíe significativamente con respecto de v_0, entonces v_1 es suficientemente cercano a v_0 con lo cual podemos decir que el cambio de posición de la partícula es aproximadamente igual al cambio en el tiempo multiplicado por la velocidad de la partícula (v_1).

Recordando que el desplazamiento de la partícula es gobernado por una ecuación diferencial, podemos conocer el valor de v_1 por medio de la ecuación $v_1 = F(u(t_1), t_1)$. Con lo cual, podemos desarrollar una aproximación de la solución en el tiempo t_1 por medio de

$$u_1 = u_0 + (t_1 - t_0)F(u_1, t_1), \tag{2-24}$$

ya que la única incógnita es la variable u_1, es posible resolver la ecuación para predecir el comportamiento de la partícula en el tiempo t_1. De esta forma, siguiendo el mismo procedimiento del método de Euler, podemos emplear el valor de u_1 para reconstruir una secuencia $u_0, u_1, u_2, u_3 \ldots$ que nos aproxime la solución de la ecuación diferencial.

De esta forma, si aplicamos el método de Euler hacia atrás para aproximar la solución de la ecuación 2-20, y sustituyendo el valor de F en la ecuación 2-24, tenemos que para el n-ésimo

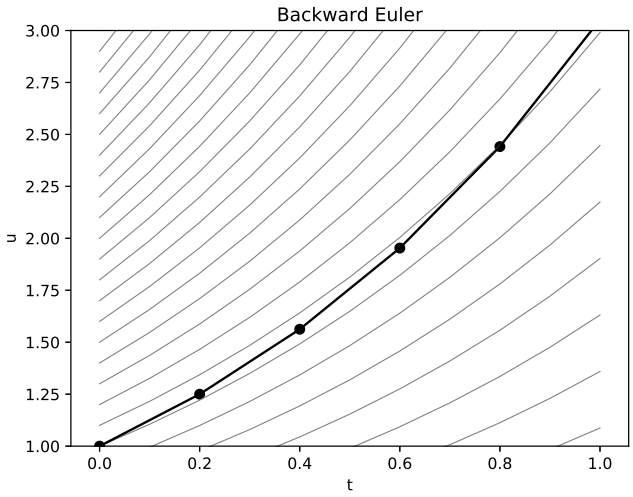


Figura 2-2: Aproximación usando Euler hacia atrás.

paso, la ecuación que debemos resolver es

$$u_{n+1} = u_n + (t_{n+1} - t_n)u_{n+1}, \tag{2-25}$$

la cual podemos resolver fácilmente al reescribir como

$$u_{n+1} = \frac{u_n}{1 - \Delta_n}. \tag{2-26}$$

Siguiendo este procedimiento secuencialmente, podemos reconstruir nuestra aproximación a la solución obteniendo la serie $1.0, 1.25, 1.56, 1.95, 2.44, 3.05, ...$ la cual se muestra en la gráfica 2-2. Tanto en la figura 2-1 como en 2-2 podemos ver en color gris claro el comportamiento de varias soluciones exactas que inician en un punto diferente y corresponderían al comportamiento que esperamos ver en nuestra solución aproximada. Debido al tamaño tan grande que hemos empleado para Δ_n es tan notorio el error que se acumula conforme nuestra aproximación va evolucionando, el cual podemos disminuir si disminuimos el paso de tiempo δ_n entre aproximaciones consecutivas.

Para aproximar el comportamiento de la partícula, tanto en Euler como en Euler hacia atrás

hemos partido de suponer que el margen de tiempo entre dos aproximaciones consecutivas es tan pequeño que no representa variación significativa para el comportamiento de la velocidad de la partícula, si la velocidad no varía significativamente, entonces podríamos tomar cualquier valor intermedio entre v_0 y v_1 para calcular nuestra siguiente aproximación a u_1; en particular, podríamos tomar el punto medio para dar origen al método de trapecios, de esta forma para aproximar u_1 debemos resolver la ecuación

$$u_1 = u_0 + (t_1 - t_0)\frac{1}{2}\left(F(u_0, t_0) + F(u_1, t_1)\right). \tag{2-27}$$

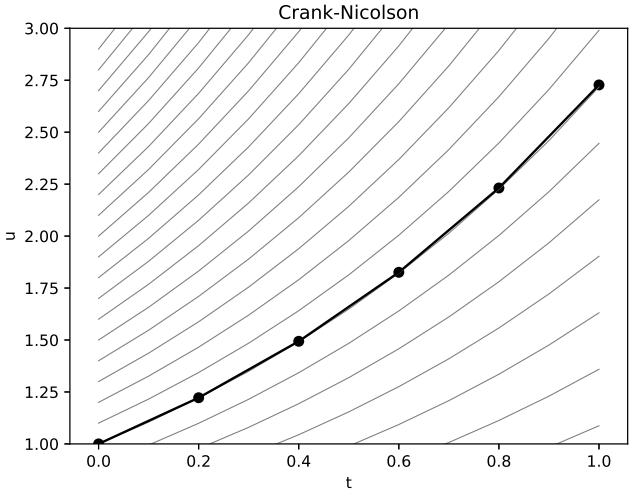


Figura 2-3: Aproximación usando el método de trapecios.

De manera análoga al procedimiento seguido en los métodos de Euler, para obtener una aproximación a la solución de mi ecuación diferencial, es requisito indispensable, resolver la ecuación 2-28 de manera iterativa para obtener una secuencia de resultados que nos aproximen la solución al problema de valores iniciales. De esta forma, si queremos aplicar este método al problema 2-20, al sustituir F en la ecuación 2-28 obtenemos para el paso n-ésimo

$$u_{n+1} = u_n + \Delta_n \frac{1}{2}\left(u_n + u_{n+1}\right), \tag{2-28}$$

que despejando adecuadamente, nos conduce a la ecuación

$$u_{n+1} = u_n \frac{1 + \frac{\Delta_n}{2}}{1 - \frac{\Delta_n}{2}}, \tag{2-29}$$

y en consecuencia, al resolverla, obtenemos la serie $1.0, 1.22, 1.49, 1.83, 2.23, 2.727, \ldots$, la cual difiere en el sexto término por menos de una centésima del valor real.

Introducción a convergencia

Comparando los resultados obtenidos por cada método numérico, de manera natural surge la duda sobre la precisión de cada uno para reproducir las características del modelo diferencial y que tan grande es el error resultante de emplear cada método. Una primer aproximación bastante burda para saber que tan bueno es un método con respecto a otro, es conocer el orden del método, el cual se obtiene, análogo a 1.2, de expresar las ecuaciones de nuestro método a partir de expansiones en series de Taylor y analizar el tamaño de el residuo; para el método de Crank-Nicolson, recordemos que su formula es

$$\frac{u_{n+1} - u_n}{\Delta_n} = \frac{f(u_{n+1}) + f(u_n)}{2} \tag{2-30}$$

esta ecuación surge de desarrollar la derivada de u en series de Taylor alrededor del punto $u^{n+\frac{1}{2}}$ y como resultado obtenemos un método de orden 2 en el tiempo. Sin embargo, esto no es suficiente para determinar el buen comportamiento de un método numérico.

En este sentido, uno desea en primer lugar, medir que tanto se aleja nuestra aproximación del valor exacto al transcurrir un paso de la aproximación, a esto es a lo que llamamos error local de truncamiento (ELT) y depende tanto del método de aproximación como del problema mismo que se está resolviendo. En el caso de la función exponencial que se revisó en la ecuación 2-20 y empleando el método de Crank-Nicolson, para calcular el error local de truncamiento es conveniente calcular la solución exacta $f(t)$ y sustituirla en el método numérico ($u_n = f(t_n)$) para calcular el error ($\tau_n = |u_{n+1} - f(t_{n+1})|$) que resulta un paso adelante en el tiempo. De

manera general, se puede suponer que la ecuación que pretendemos resolver es de la forma

$$u' = f(t, u). \tag{2-31}$$

En la gran mayoría de los casos, emplear la solución exacta de la ecuación diferencial no es posible, sin embargo, es posible realizar este cálculo pidiendo que la solución sea Lipschitz, es decir, para puntos cercanos, los cambios en la función deben estar acotados por la distancia entre puntos multiplicada por una constante llamada constante de Lipschitz, es decir

$$|f(x) - f(y)| \leq \lambda |x - y| . \tag{2-32}$$

Empleando esta suposición, es posible desarrollar el método para analizar el ELT, partiendo, de reescribir la expresión 2-30 como

$$u_{n+1} = u_n + \frac{\Delta_n}{2} \left[f(t_n, u_{n+1}) + f(t_n, u_n) \right], \tag{2-33}$$

tomando en cuenta que u puede representar todo un vector de aproximaciones. Si sustituimos nuestra solución exacta en la aproximación y restamos para calcular el error llegamos a la expresión

$$\tau_n = u(t_{n+1}) - \left\{ u(t_n) + \frac{\Delta_n}{2} \left[f(t_n, u_n) + f(t_{n+1}, u_{n+1}) \right] \right\}, \tag{2-34}$$

donde τ_n es el ELT en el paso n-ésimo; desarrollando en series de Taylor los términos $u(t_{n+1})$, $u(t_{n+1})$ y empleando la ecuación 2-31 llegamos a

$$\begin{aligned} \tau_n &= \left[u(t_n) + \Delta_n u'(t_n) + \frac{\Delta_n^2}{2} u''(t_n) + \mathcal{O}(\Delta_n^3) \right] \\ &- \left[u(t_n) + \frac{\Delta_n}{2} \left\{ u'(t_n) + \left[u'(t_n) + \Delta_n u''(t_n) + \mathcal{O}(\Delta_n^2) \right] \right\} \right] \\ &= \mathcal{O}(\Delta_n^3), \end{aligned} \tag{2-35}$$

esto nos lleva a concluir que existe una constante k tal que el ELT es menor que $k\Delta^3$.

Decimos que un método es consistente si el ELT tiende a 0 cuando refinamos nuestras aproximaciones, en este caso, cuando $\Delta_n \to 0$. La consistencia nos indica que nuestro método es una discretización razonable puntualmente al problema que deseamos resolver.

En este caso es fácil ver que Crank-Nicolson es consistente al calcular el límite de la ecuación 2-35 cuando Δ_n tiende a cero. Si bien, la consistencia es necesaria para saber que nuestro método nos será de utilidad para aproximar la solución a nuestro problema, requerimos de una herramienta más poderosa para mostrar que a largo plazo, nuestra aproximación seguirá siendo de utilidad para estudiar el comportamiento del método.

Esta herramienta la llamamos convergencia, si bien, la consistencia nos dice que en cada paso de cálculo, si tomamos pasos de tiempo suficientemente pequeños podremos hacer el error (ELT) tan pequeño como deseamos, la convergencia nos dice que este error se va a propagar de manera razonablemente lenta a través de varios pasos de cálculo y si los pasos de tiempo son suficientemente pequeños, este error global también lo podremos hacer tan pequeño como queramos.

Dicho lo anterior, definimos el error global como el error que se transmite y amplifica al emplear el método tras varios pasos sucesivos de cálculo, más el error que se genera en cada paso de cálculo. Partiendo de la ecuación 2-33, tomando la convención de usar $u(t_n)$ para los valores exactos de la solución en el tiempo t_n y haciendo todos los incrementos de tiempo Δ_n iguales entre si a una constante h, podemos expresar el error global en el paso de tiempo $n+1$ por medio de

$$e_{n+1,h} = e_n, h + \frac{h}{2}\left\{[f(t_n, y_n) - f(t_n, y(t_n))] + [f(t_{n+1}, y_{n+1}) - f(t_{n+1}, y(t_{n+1}))]\right\} + \mathcal{O}(h^3). \tag{2-36}$$

Tomando la norma de esta ecuación y empleando la desigualdad del triangulo, podemos expresar el error global por medio de

$$\|e_{n+1,h}\| \leq \|e_{n,h}\| + \frac{h}{2}\left\{\|f(t_n, y_n) - f(t_n, y(t_n))\| + \|f(t_{n+1}, y_{n+1}) - f(t_{n+1}, y(t_{n+1}))\|\right\} + \mathcal{O}(h^3), \tag{2-37}$$

de esta forma, si usamos la condición impuesta de que la solución sea Lipschitz, sabemos que existe una constante que llamaremos λ que nos indica que la diferencia entre dos pendientes cercanas es menor que λ veces la distancia entre estos puntos la cual corresponde a nuestro error global en los tiempos correspondientes. De igual manera, por lo visto en la ecuación 2-35, podemos reescribir el error global como

$$\|e_{n+1,h}\| \leq \|e_{n,h}\| + \frac{h\lambda}{2}\left\{\|e_{n,h}\| + \|e_{n+1,h}\|\right\} + kh^3, \tag{2-38}$$

desarrollando esta ecuación y tomando en cuenta las consideraciones adecuadas, podemos expresar el error global como una desigualdad de la forma

$$\|e_{n+1,h}\| \leq \left(\frac{1+\frac{h\lambda}{2}}{1-\frac{h\lambda}{2}}\right)\|e_{n,h}\| + \frac{k}{1-\frac{h\lambda}{2}}h^3; \tag{2-39}$$

desarrollando recursivamente esta desigualdad, podemos expresar el error global por medio de

$$\|e_{n+1,h}\| \leq \frac{c}{\lambda}\left[\left(\frac{1+\frac{h\lambda}{2}}{1-\frac{h\lambda}{2}}\right) - 1\right]h^2. \tag{2-40}$$

Notemos que el error global es de orden 2, lo cual es deseable ya que permite reducir el error de manera más rápida conforme h tiende a 0.

Tomemos en consideración que, si $0 < h\lambda < 2$ podemos seguir las desigualdades siguientes

$$\frac{1+\frac{h\lambda}{2}}{1-\frac{h\lambda}{2}} = 1 + \frac{h\lambda}{1-\frac{h\lambda}{2}} \tag{2-41}$$

$$\leq e^{\frac{h\lambda}{1-\frac{h\lambda}{2}}}, \tag{2-42}$$

con lo cual, podemos reescribir la desigualdad 2-40 como

$$\|e_{n+1,h}\| \leq \frac{kh^2}{\lambda}e^{\frac{T\lambda}{1-\frac{1}{2}h\lambda}}, \tag{2-43}$$

donde $T \geq nh$.

Retomando la convergencia, un método decimos que es convergente cuando para un tiempo T dado, el error global en el paso n lo podemos acotar de tal forma que, si hacemos decrecer h para que tienda a cero, nuestro error global va a decrecer tendiendo también a cero. Analizando la ecuación 2-43, es posible demostrar que nuestro método de aproximación es convergente. La convergencia, nos da una idea clara de que, refinando adecuadamente el método de aproximación a la solución de nuestro problema, podremos obtener una aproximación razonable con un error pequeño dependiente del tamaño de paso que se elija.

Si bien, partiendo de la convergencia, podemos observar que con tamaños de paso suficientemente pequeños, el error global se puede hacer muy pequeño, por contra parte, al hacer pasos de tiempo muy refinados, el poder de cómputo requerido será cada vez mayor, al requerir más pasos de cómputo para lograr el mismo objetivo. Debido a esto, en los métodos numéricos se busca lograr un equilibrio entre precisión y velocidad de cómputo, una de las opciones para lograr este objetivo consiste en usar métodos de alto orden tanto espaciales como temporales.

Cuando empleamos métodos de mayor orden, obtenemos beneficios en la reducción del poder de cómputo necesario para mantener el error a un mismo nivel, por ejemplo, al usar un método de primer orden, cuando uno duplica el número de pasos en el tiempo, obtiene una reducción a la mitad del error, mientras que, al emplear métodos de segundo orden bajo las mismas condiciones el error se parte a la cuarta parte del error original.

Nuevamente, la mejora no se obtiene gratuitamente, ya que al aumentar el orden de un método, este requiere más información para aproximar las soluciones en cada paso de tiempo, de esta forma, es necesario elegir un equilibrio adecuado en la elección del método para cada problema.

Capítulo 3

Métodos numéricos para la ecuación de flujo

Entrando en materia con la solución de la ecuación de Richards, previo a poder aplicar algoritmos adecuados para aproximar dicha solución, es necesario definir algunas características de la región que se pretende simular, estas características corresponden a dimensiones y geometrías físicas del medio donde se desea aproximar la solución, características mecánicas del suelo, condiciones de frontera e interfaz cuando van a interactuar dos o más medios.

Partiendo de estas características del problema, sigue plantear la forma en que se dividirá el dominio o espacio físico sobre el cual se desea aproximar la solución, esto se conoce como mallado y define tanto la distribución de los puntos que se desea observar como la relación que hay entre ellos en términos de vecindad.

Concluido el mallado, es necesario discretizar las ecuaciones en términos de las variables espaciales aplicando alguno de los métodos vistos en el capítulo 2, de tal forma que las ecuaciones que simulan el proceso de infiltración se muevan de un dominio continuo, a un sistema de ecuaciones algebraicas. Posteriormente, continúa la discretización temporal de las ecuaciones, que de igual manera a la discretización espacial, este paso nos aproxima las ecuaciones dependientes del tiempo por medio de un sistema discreto de ecuaciones algebraicas por resolver.

La ecuación de Richards es un problema no lineal, debido a esto, es conveniente tomar

medidas adicionales para disminuir los errores que se producen debido a los métodos de aproximación. Cada problema, tiene una forma diferente de abordarse, en el caso de la ecuación de Richards, este trabajo plantea un método adaptivo en el tiempo, el cual se describe en la sección correspondiente.

3.1. Región física

Antes de empezar a simular, es necesario definir de manera precisa que es lo que se desea estudiar, para llevar a cabo este análisis, se tomó en cuenta los procesos de infiltración en terraplenes, para lo cual, tomando en cuenta el trabajo de [Norambuena-Contreras *et al.*, 2012] con motivos de comparación se tomó en consideración un corte transversal a la carretera de un terraplén, el cual se eligió con una altura de cinco metros, de los cuales, dos metros en la base corresponden al suelo natural. Este corte tiene un ancho de nueve metros, considerando una distribución de tal forma que los primeros dos metros corresponden a la carpeta asfáltica la cual se considera impermeable, el siguiente metro corresponde al hombro de la carretera y los siguientes seis metros corresponden al talud del terraplén el cual empieza a la altura del asfalto y termina a ras del suelo natural.

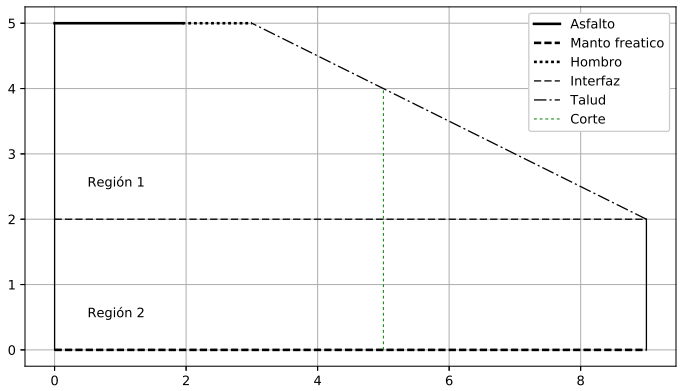


Figura 3-1: Geometría del terraplén.

En la figura 4-48 podemos observar gráficamente la distribución física del terraplén consi-

derado para la simulación. A una distancia de cuatro metros medidos horizontalmente desde la base del talud, se puede observar una línea punteada, esta línea corresponde a un corte vertical sobre el terraplén empleado para simulación y comparación de los datos numéricos.

Parte del objetivo de este trabajo consiste en desarrollar un método para aproximar la solución a la ecuación de Richards cuando existen dos medios físicos diferentes como podrían ser por ejemplo arcilla y limo, materiales que al ser diferentes entre sí, presentan diferentes características de permeabilidad y curvas de retención de agua. En la figura 4-48 podemos observar dos regiones delimitadas al interior del terraplén por una región llamada interfaz que se localiza verticalmente a dos metros sobre la base y corresponde a la separación entre el suelo natural y el material colocado para formar el terraplén.

3.2. Mallas

Una vez definido el espacio físico que se desea estudiar, es importante definir el tipo de mallas que se empleará para calcular las aproximaciones al problema. En la literatura existen diferentes tipos de mallas y malladores optimizados para diferentes tipos de problemas, cada uno de ellos presenta sus propias ventajas y desventajas a tomar en cuenta para cada problema que se desea estudiar.

Dependiendo del tipo de malla que se elija, el método para discretizar las ecuaciones variará ligeramente, las mallas no estructuradas permiten refinar el trabajo en zonas arbitrarias de manera relativamente sencilla a costa de estructuras de datos adicionales para mantener información de la misma como es la vecindad de los nodos. Por contra parte, las mallas estructuradas imponen restricciones adicionales a la adición de nodos eliminando la necesidad de estructuras adicionales para conservar información de la vecindad entre nodos.

Ejemplo de este último caso lo encontramos en las mallas lógicamente rectangulares, una malla rectangular la empleamos para dividir mediante una malla una región rectangular de tal forma que queda partida en rectángulos uniformes, generalmente los vértices de cada rectángulo que la subdividen son almacenados por medio de matrices donde cada entrada corresponde a

las coordenadas de un vértice y las entradas adyacentes corresponden a vértices adyacentes de la malla. De esta forma, sin necesidad de estructuras de datos adicionales podemos conservar la información de adyacencia o vecindad para todos los puntos de interés en la malla.

Una malla lógicamente rectangular parte de esta misma lógica para adecuarse a regiones o contornos que no necesariamente forman un rectángulo, en lugar de ello se establece un mapeo de forma tal que el contorno se asocia con un rectángulo y sobre este contorno se asocian lineas llamadas horizontales y verticales de tal forma que se crean cuadriláteros sobre toda la región, y sobre cada vértice sólo hay involucradas una linea horizontal y una vertical. De esta forma, aunque no necesariamente los puntos conservan una distribución regular como en una malla rectangular, podemos conservar la información de adyacencias entre los nodos de la malla.

Si bien, las mallas lógicamente rectangulares no siempre resultan ser óptimas para todos los problemas, en ocasiones su simpleza permite trabajar de manera sencilla los problemas de mallado. En el caso del presente trabajo, existe un mapeo muy sencillo entre el terraplén y un rectángulo, debido a lo cual se consideró entre otras razones como una propuesta inicial para generar el mallado del terraplén.

Cabe mencionar que un mallado rectangular para este terraplén no resulta en una condición óptima para la simulación, sin embargo se consideró continuar con este tipo de malla para generar a propósito una condición desfavorable que permita probar la robustez del método numérico propuesto ante mallas no óptimas.

Debido a la geometría que tiene el terraplén, la malla correspondiente al mismo, se genera en dos partes, de tal forma que la región dos es formada por una malla rectangular mientras que la región uno se genera por medio de interpolación transfinita a partir de una malla rectangular cuyo vértice superior derecho se ajusta para coincidir con la intersección entre el fin del hombro y el inicio del talud en el terraplén, y a partir de este punto se ajustan convenientemente todos los demás puntos involucrados, iniciando por los contornos y continuando por el interior de la región.

Para definir las dimensiones que tendrán las matrices que almacenarán la información correspondiente a cada malla, el código para generar el mallado recaba algunos datos correspondientes

al número de nodos que se espera en cada dirección y el número de capas que se emplearán para refinar las mallas en zonas de alto interés como es la región de interfaz (refI) y las regiones sujetas a infiltración como es el talud y el hombro (refV,refS). Posterior a esto, el código calcula el número de nodos requerido para la malla previo a las operaciones de refinado y elige adecuadamente el número de nodos que asignará a cada región de tal forma que ambas mallas sean relativamente uniformes.

```
# nodos horizontales y verticales de la malla
p = q = 41;


# capas asignadas para refinar
refV = 3*[1];
refS = 2*[1 1];
refI = 1*[1 1];


# numero de nodos verticales para la región 1
pm = round((p-sum(refS)-2*sum(refI))*h1/h2);
pm = trunc(Int,pm);
ph1 = pm + sum(refI);
```

En el código previo se observan las variables relacionadas al proceso de refinamiento el cual se explica más adelante, cada entrada de esta matriz indica una ejecución de refinamiento que actúa sobre las n capas más externas de la matriz generando una capa adicional que parte por la mitad a la anterior.

Si no tomáramos en cuenta los procesos de refinado de la malla se obtendría un resultado como el que se muestra en la figura 3-2, sin embargo, debido a las condiciones que se presentan en la región del hombro y del talud donde una condición de equilibrio es cambiada rápidamente hacia condiciones de saturación, para poder capturar los procesos involucrados en esta región se aumenta la densidad de nodos.

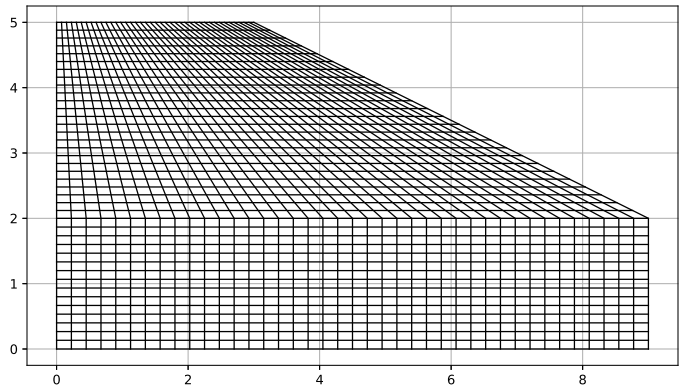


Figura 3-2: Malla sin refinamiento.

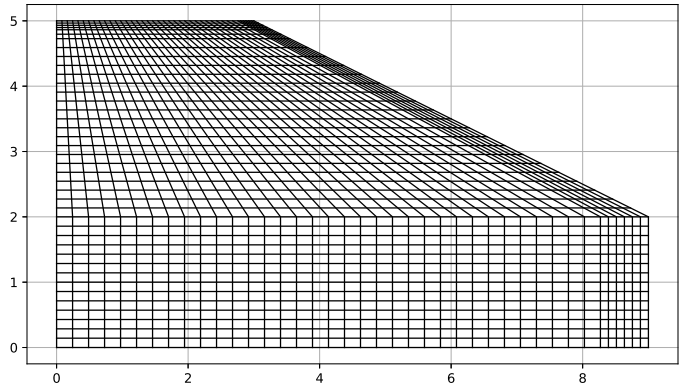


Figura 3-3: Malla refinada en el talud y el hombro.

Para llegar a esto, el código crea las matrices de coordenadas para cada región generando una malla rectangular sin tomar en cuenta las capas correspondientes a los refinamientos. Para generar los refinamientos, el código hace uso de las funciones refinaSup, refinaInf y refinaDer, las cuales toman como argumento la matriz rectangular y una matriz de instrucciones y regresan una matriz rectangular con filas o columnas adicionales que corresponden a las capas del refinamiento. Estas capas son generadas insertando una fila o columna adicional a la matriz original según corresponda e insertando en ellas las coordenadas del nodo que refina esta capa.

```
function refinaSup(x,y,n)
```

```
  x=x'; # La comilla indica la transposición de la matriz
  y=y';
  for i = n
    xp = x[1:(end-i),:];
    yp = y[1:(end-i),:];
    for j = i:-1:1
      xp = [xp; [0.5 0.5;0 1]*x[end-j:end-j+1,:]];
      yp = [yp; [0.5 0.5;0 1]*y[end-j:end-j+1,:]];
    end
    x = xp;
    y = yp;
  end
  return (x',y');
end
```

Finalmente, tras concluir las funciones de refinamiento, el código altera las coordenadas correspondientes al talud para generar su inclinación correcta para la malla final, una vez corregidas las coordenadas para el talud, el código continua alterando proporcionalmente todas las demás coordenadas para llegar al resultado final el cual concluye al unir las dos mallas en una sola. En la porción de código a continuación se muestra las instrucciones correspondientes a esta labor.

```
# Corrijo las fronteras
# Asigno las coordenadas en x del talud para la frontera
xs[:,end] = xs[:,end]*talud/base;
xs[end,:] = base-(ys[end,:]-h1)*(base-talud)/(h2-h1);

# Corrijo los interiores
# Asigno las coordenadas en x a los nodos del interior
```

```
for i = 2:size(xs,2)-1
xs[2:end-1,i] = xs[2:end-1,i]*xs[end,i]/xs[end,1]; end


# Pego las mallas
x=[xi xs[:,2:end]];
y=[yi ys[:,2:end]];
```

El código completo correspondiente a las funciones de generación de mallas se puede observar en el apéndice A.

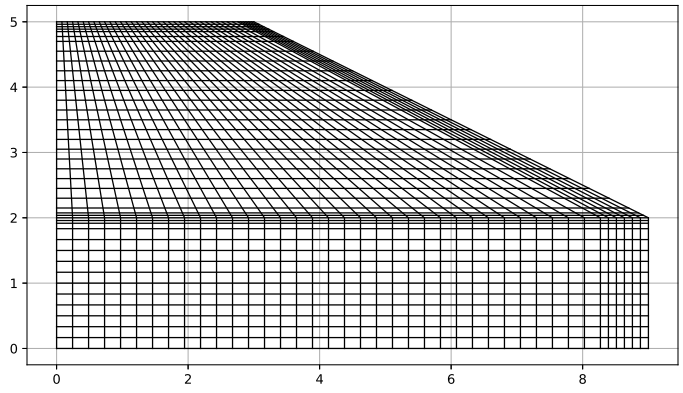


Figura 3-4: Malla refinada en el talud, el hombro y la región de interfaz.

En la figura 4-11 podemos observar la malla empleada para la simulación del proceso de infiltración suponiendo que la región uno como la dos tienen iguales características, por lo cual no es necesario tomar consideraciones especiales para la región de interfaz, por el contrario, si ambas regiones muestran características diferentes, para poder capturar de mejor manera los fenómenos que se presentan en la región de interfaz, es conveniente generar una malla más densa en esta zona, dando lugar a la malla que se observa en la figura 4-13, la cual fue empleada para probar el código en condiciones donde dos medios diferentes interactúan entre si.

3.3. Discretización espacial

Previo a proceder a la discretización espacial de las ecuaciones diferenciales, resulta conveniente ejemplificar dos posibles acercamientos. Nosotros pretendemos generar un operador al que podemos llamar L el cual deseamos que al aplicarlo a una variable u se comporte de manera similar a nuestra ecuación diferencial, es decir $Lu \approx f(u, u', \dots) = g(u)$. En este punto, tenemos en nuestras manos una elección, ya que las condiciones de frontera son algo conocido, tenemos la oportunidad de incluirlas o excluirlas de nuestra variable u vector.

En caso de excluirlas, dado que no podemos simplemente eliminar información, para conservarla debemos pasarlas al lado derecho de la ecuación y en consecuencia, nuestro operador L se convierte en una matriz cuadrada con tantos renglones como datos en nuestro vector u. Por el contrario, si se desea conservar la información de las fronteras en el vector u, la matriz de nuestro operador L resulta resulta de mayor tamaño al contener renglones nuevos que corresponden a las ecuaciones de la frontera, si bien esto requiere más memoria, resulta un poco más sencillo cambiar las condiciones de frontera por medio de pequeños cambios agregados por medio de un vector del lado derecho de la ecuación.

Para este trabajo se empleó el segundo enfoque con la intención de poder manipular fácilmente las condiciones de frontera correspondientes a las regiones donde se va a infiltrar el agua durante las simulaciones.

Tomando en cuenta este enfoque se crearon las funciones gammma y gammat, las cuales tienen como objetivo ayudar en el armado de las matrices que aproximan el operador

$$Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + F. \tag{3-1}$$

La primera de ellas, toma en cuenta el procedimiento mostrado en la sección 1.3 para calcular los pesos que se debe asignar a cada vecino de tal forma que se aproxime el operador buscado.

```
function gammma(x,y,p0,A,B,C,D,E,F)
  N = length(x);
  I = [1:(p0-1);(p0+1):N];
```

```
  dx = x[I]-x[p0];
  dy = y[I]-y[p0];
  M = [dx dy dx.*dx dx.*dy dy.*dy]';

  Gamma = pinv(M)*[D E 2*A B 2*C]';

  Gamma = [F-sum(Gamma);Gamma];
  Gamma = [Gamma[2:p0];Gamma[1];Gamma[(p0+1):N]];
end
```

Este código previo toma como argumentos las posiciones en x y y de los puntos cercanos al nodo en que nos interesa aproximar el operador, incluyendo las coordenadas del mismo, la posición de este en los vectores de coordenadas x y y está dado mediante la variable $p0$ mientras que las siguientes variables representan los coeficientes correspondientes a la ecuación que se desea aproximar. Como resultado obtenemos los pesos correspondientes a cada nodo para aproximar el operador siguiendo el algoritmo citado para el cálculo de los pesos referido en la sección 1.3.

Por su parte, la función gammat, es la encargada de ensamblar las matrices, esta función, toma como argumentos las matrices de coordenadas de una malla lógicamente rectangular, los coeficientes del operador que busca aproximar, los límites de la región sobre la cual aproximará el operador e información sobre las fronteras de esta región.

De esta forma, si deseamos calcular el operador sobre la n-ésima entrada, calculará el vector de pesos usando la función gammma y los datos resultantes los acomodará sobre la columna que corresponde a cada entrada en la n-ésima fila. Las coordenadas que pasa a la función gammma son una submatriz de 3x3 tomada de la matriz de coordenadas, esta submatriz está centrada en el punto que deseamos aproximar, en caso de que tiene como entrada central el nodo sobre el cual deseamos aproximar el operador, salvo que la submatriz quedara fuera de la región a la

cual le hemos limitado, en este caso se recorre quedando descentrada.

El código que se muestra a continuación muestra el ciclo central de cálculo de esta función para el nodo en la posición i, j.

```
# Calculo el centro de mi submalla.
i0 = min(Mr[2]-1,max( Mr[1]+1,i))
j0 = min(Nr[2]-1,max( Nr[1]+1,j))

# Recupero las coordenadas de mi submalla.
u = x[(i0-1):(i0+1),(j0-1):(j0+1)]
v = y[(i0-1):(i0+1),(j0-1):(j0+1)]

# Recupero la posición de mi nodo p0 sobre la submalla.
p0 = 5+(i-i0)+3*(j-j0)

# Calculo el vector de gammas.
k = gammma(u,v, p0, A,B,C,D,E,F)

# Recupero renglón y columna sobre mi matriz del operador.
p = (j-1)*M + i
p0 = (j0-1)*M + i0

# Armado de la matriz del operador.
K[p,((p0-1):(p0+1))-M] = [k[1] k[2] k[3]];
K[p,(p0-1):(p0+1)] = [k[4] k[5] k[6]];
K[p,((p0-1):(p0+1))+M] = [k[7] k[8] k[9]];
```

Una vez ensamblada la matriz de este operador, la función regresa la matriz resultante. El código completo de esta función se puede consultar en el apéndice A.

Una vez definido el proceso por medio del cual se generan las matrices, es conveniente hacer mención de algunas características de desempeño. La ecuación de Richards es una ecuación no lineal que depende continuamente de la presión de poro, aislando parte de este operador para hacer evidente esta dependencia, si u representa la presión de poro, parte de nuestro operador está dada por $Ku_{xx} + Ku_{yy}$, donde K es una función de permeabilidad que depende de la presión de poro, lo cual cambia continuamente los coeficientes de nuestro operador de Richards.

El proceso de cálculo de los operadores resulta altamente costoso debido a la inversión de matrices que se requiere, de igual manera, otros procesos alternos para el cálculo del operador se vuelven costosos si involucran la inversión de matrices. Una alternativa de menor costo a este problema consiste en calcular un operador alterno y escalarlo de acuerdo a las necesidades, empleando esta aproximación basta con calcular el operador $L \approx u_{xx} + u_{yy}$ y cuando necesito cambiarlo sólo hago operaciones sobre los renglones correspondientes de tal forma que $KL \approx Ku_{xx} + Ku_{yy}$.

Empleando este enfoque, el algoritmo crea operadores genéricos que después son usados para recalcular los operadores necesarios en cada paso de cómputo. A continuación se muestra el código encargado de generar los operadores genéricos.

```
# Operadores al interior de la malla
L2 = gammat(x,y, 1,0,1,0,0,0, [2 p-1],[2 q-1])
Lx = gammat(x,y, 0,0,0,1,0,0, [2 p-1],[2 q-1])
Ly = gammat(x,y, 0,0,0,0,1,0, [2 p-1],[2 q-1])


# Operador identidad para el interior y frontera respectivamente
I = gammat(x,y, 0,0,0,0,0,1, [2 p-1],[2 q-1])
Ie = speye(size(I,1)) - I


# Operador en fronteras Dirichlet
p0 = div(p,3)+1 # Punto donde inicia el hombro de la carretera
q0 = ph1 # Punto más bajo del talud
```

```
#q0 = div(q,3)


fd = gammat(x,y, 0,0,0,0,0,1, [1 p],[1 1]) # Base del terraplén
fd = fd + gammat(x,y, 0,0,0,0,0,1, [p0 p],[q q]) # Talud
fd = fd + gammat(x,y, 0,0,0,0,0,1, [p p],[q0 q]) # Hombro


#Operador en fronteras Neumann
fn = gammat(x,y, 0,0,0,-1, 0,0, [1 1],[2 q-1]) # Centro del terraplén
fn = fn + gammat(x,y, 0,0,0, 1, 0,0, [p p],[2 q0-1]) # Frontera derecha.
fn = fn + gammat(x,y, 0,0,0, 0, 1,0, [1 p0-1],[q q]) # Carpeta
```

Finalmente estos operadores son ensamblados para generar el operador final que aproxima a mi ecuación.

```
# Actualizo los operadores.
K = fK(zg)


mw = (max(fm(0.5*(z+zg)),1e-3)*g).*I;
L = K.*L2+(Lx*K).*Lx+(Ly*K).*Ly;


rhs = fd*min(z+1000*c0,0)+c0*g*Ly*(K+Kp)/2
```

Es necesario mencionar que para manejar varias regiones diferentes, el código se debe modificar para incluir las ecuaciones que modelan el comportamiento de la interfaz, así como manejar de forma diferente cada región, ya que entre regiones varían las ecuaciones de permeabilidad y del módulo de almacenamiento de agua. Para manejar esta diferencia, el código incluye algunas modificaciones como son

```
# Adiciones para manejar varias regiones
L2_R1 = gammat(x,y, 1,0,1,0,0,0, [2 p-1],[ph1 q-1], [1 p],[ph1 q])
Lx_R1 = gammat(x,y, 0,0,0,1,0,0, [2 p-1],[ph1 q-1], [1 p],[ph1 q])
```

```
Ly_R1 = gammat(x,y, 0,0,0,0,1,0, [2 p-1],[ph1 q-1], [1 p],[ph1 q])


L2_R2 = gammat(x,y, 1,0,1,0,0,0, [2 p-1],[2 ph1], [1 p],[1 ph1])
Lx_R2 = gammat(x,y, 0,0,0,1,0,0, [2 p-1],[2 ph1], [1 p],[1 ph1])
Ly_R2 = gammat(x,y, 0,0,0,0,1,0, [2 p-1],[2 ph1], [1 p],[1 ph1])


I_R1 = gammat(x,y, 0,0,0,0,0,1, [2 p-1], [ph1+1 q-1], [1 p],[1 q])
I_in = gammat(x,y, 0,0,0,0,0,1, [2 p-1], [ph1 ph1], [1 p],[1 q])
I_R2 = gammat(x,y, 0,0,0,0,0,1, [2 p-1], [2 ph1-1], [1 p],[1 q])
```

Modificaciones que a su vez modifican la forma en que se calculan los operadores. Para manejar las ecuaciones de la interfaz, se plantea una solución que parte de la ecuación de continuidad, viendo de esta manera, si tomamos un parche de superficie sobre la interfaz, el flujo que entra por una cara es igual al flujo que sale por la otra cara, de esta manera, la ecuación que modela la interfaz resulta ser $Ku_y|_{R1} - Ku_y|_{R2} = 0$.

```
L_R1 = I_R1*(K1.*L2_R1+(Lx_R1*K1).*Lx_R1+(Ly_R1*K1).*Ly_R1);
L_R2 = I_R2*(K2.*L2_R2+(Lx_R2*K2).*Lx_R2+(Ly_R2*K2).*Ly_R2);
L_in = I_in*(K1.*Ly_R1-K2.*Ly_R2)


L = L_R1+L_R2
mw = I_R1*mw1+I_R2*mw2
```

3.4. Discretización temporal

Una vez que hemos lidiado con la discretización espacial, sólo nos queda lidiar con la discretización en el tiempo, para este paso existen diferentes alternativas como se ha mostrado en la sección correspondiente. Este trabajo eligió discretizar empleando el método de Crank-Nicolson debido a las buenas características que presenta de estabilidad y a que es una discretización

de segundo orden en el tiempo, lo cual mejora los resultados al compararlo con métodos como Euler.

Visto de una manera general, nuestras ecuaciones diferenciales presentan la forma

$$u_t = f(u_{xx}, u_{xy}, u_{yy}, u_x, u_y, u) + g(u, t), \tag{3-2}$$

donde g corresponde a la parte no homogénea de la ecuación diferencial, en el caso de la ecuación de Richards existe una pequeña modificación adicional al incluir el módulo de almacenamiento de agua (m_w) como parte de la ecuación diferencial, de esta forma la ecuación se convierte en

$$m_w u_t = f(u_{xx}, u_{xy}, u_{yy}, u_x, u_y, u) + g(u, t). \tag{3-3}$$

Para ir desglosando poco a poco las ecuaciones, podemos reescribir el sistema anterior como

$$m_w u_t = s. \tag{3-4}$$

El método de Crank-Nicolson parte de hacer diferencias centradas medio paso adelante en el tiempo, de esta forma, al discretizar en el tiempo la ecuación anterior, requerimos evaluar tanto m_w como s medio paso adelante en el tiempo resultando

$$m_w|_{n+\frac{1}{2}} \left(\frac{u^{n+1} - u^n}{h} \right) = s|_{n+\frac{1}{2}}, \tag{3-5}$$

evidentemente no somos capaces de evaluar las funciones en estos puntos ya que no tenemos información de la presión de poro que nos sirva para evaluar estos puntos, sin embargo estos puntos se pueden interpolar a partir de los tiempo más cercanos los cuales podemos conocer, es decir, en los tiempo n y $n+1$, de esta forma obtenemos

$$\frac{m_w^{n+1} + m_w^n}{2} \left(\frac{u^{n+1} - u^n}{h} \right) = \frac{s^{n+1} + s^n}{2}. \tag{3-6}$$

La función que nos brinda el módulo de almacenamiento de agua es una función de la

presión de poro por lo cual lo podemos reescribir, de igual manera, sabemos que la función s nos representa la suma de un operador que depende de la presión de poro más una parte no homogénea, por lo cual podemos escribir la ecuación previa como

$$\frac{m_w(u^{n+1}) + m_w(u^n)}{2}\left(\frac{u^{n+1}-u^n}{h}\right) = \frac{L(u^{n+1})u^{n+1} + L(u^n)u^n}{2} + \frac{g(u^{n+1},t^{n+1}) + g(u^n,t^n)}{2}, \tag{3-7}$$

donde $L(u)$ hace explícito que el operador varía con la presión de poro y al ser aplicado a la variable u se aproxima mi ecuación diferencial siendo $L(u)u \approx f(u_{xx}, u_{xy}, u_{yy}, u_x, u_y, u)$.

Para poder resolver este sistema, adicionalmente vamos a hacer una linealización, la cual se corrige mediante un proceso adaptivo que se describe en la siguiente sección. Este paso consiste en crear una nueva variable $\hat{u}$ que emplearemos como sustituto de u^{n+1} de tal forma que la ecuación se vuelva lineal con respecto de u^{n+1} llegando al resultado

$$\frac{m_w(\hat{u}) + m_w(u^n)}{2}\left(\frac{u^{n+1}-u^n}{h}\right) = \frac{L(\hat{u})u^{n+1} + L(u^n)u^n}{2} + \frac{g(\hat{u},t^{n+1}) + g(u^n,t^n)}{2}. \tag{3-8}$$

Si bien esta fórmula presenta de manera breve el resultado de discretizar las ecuaciones por medio de Crank-Nicolson, aún hay un cambio más que podemos aprovechar para hacer más explícito el proceso necesario para resolverlo, de esta manera agrupando y despejando llegamos a

$$\left[m_w(\hat{u}) + m_w(u^n) - hL(\hat{u})\right]u^{n+1} = \left[m_w(\hat{u}) + m_w(u^n) + hL(\hat{u})\right]u^n + hg(\hat{u},t^{n+1}) + hg(u^n,t^n), \tag{3-9}$$

donde se puede observar la ecuación matricial que es necesario resolver para encontrar el vector de soluciones u^{n+1}.

Para un problema de valores iniciales, es usual definir las condiciones que cumple la frontera de manera independiente a las condiciones que cumple el interior, ya sea condiciones de Neumann o Dirichlet, estas son tratadas de manera un poco diferente a las condiciones del interior.

Decimos que una condición de frontera es de Dirichlet cuando imponemos condiciones sobre

el valor de una variable a lo largo de una región, un ejemplo de condición de Dirichlet sería decir que sobre cierta frontera la función solución debe cumplir con los valores de alguna otra función por ejemplo $u(x, y, t) = f(x, y)$.

Por otro lado, decimos que una condición es de Neumann cuando las restricciones impuestas actúan sobre las derivadas de la solución, es muy usual encontrar condiciones de Neumann limitando el flujo normal sobre una superficie que forma la frontera del problema, por ejemplo, si u_n representa la derivada normal sobre una superficie, podemos decir que nada fluye a través de esta con la condición $u_n = 0$.

Para manejar estas condiciones no existe una derivada temporal que nos permita manejar de forma directa las mismas ecuaciones que se vieron antes, sin embargo, partiendo de la deducción del método de Crank-Nicolson podemos desarrollar la discretización para las condiciones de frontera. El método es de orden dos debido a que emplea diferencias centradas medio paso adelante en el tiempo.

Siguiendo esta misma lógica, para implementar condiciones de frontera debemos plantearlas a medio paso de tiempo, por ejemplo si decimos que hay gradiente fijo en la dirección x y tenemos un operador L_n tal que $L_n u \approx u_x = k$ la condición que requerimos implementar está dada por

$$L_n u^{n+\frac{1}{2}} = k, \tag{3-10}$$

sin embargo, nuevamente nos enfrentamos al problema de que no tenemos información de nuestra variable u en medios pasos de tiempo. Para afrontar esta situación, nuevamente requerimos interpolar los valores de tiempo cercanos para obtener el valor de u que necesitamos, de esta manera la ecuación se convierte en

$$L_n \left(\frac{u^n + u^{n+1}}{2} \right) = k, \tag{3-11}$$

finalmente podemos reescribir esta ecuación por simplicidad llegando al resultado

$$\frac{L_n}{2} u^{n+1} = k - \frac{L_n}{2} u^n. \tag{3-12}$$

3.5. Paso adaptivo

En la sección previa se hizo mención a un proceso para linealizar las ecuaciones discretizadas, este proceso consiste en sustituir la variable u en el paso de tiempo futuro por una aproximación que hemos llamado $\hat{u}$, sustituyendo de forma adecuada para poder resolver el problema de manera sencilla.

Como se puede observar, realizar este cambio introduce nuevas complicaciones al problema de cálculo, si uno desea realizar una iteración de Crank-Nicolson debemos hacer que $\hat{u} \to u^{n+1}$, por otro lado, observemos un detalle que aparece al realizar este cambio.

Recordemos el método de Euler para una ecuación diferencial $u^{n+1} = u^n + hLu + hg(t,u)$, si cambiamos nuestro operador L por $L(u)$ para hacer evidente que éste operador puede cambiar en el tiempo, una iteración de Euler se puede ver como

$$u^{n+1} = u^n + hL(u)u + hg(t,u), \tag{3-13}$$

por otro lado, nuestro método empleando la notación de la ecuación anterior tiene la forma

$$u^{n+1} = u^n + \frac{hL(\hat{u})u + hL(u)u}{2} + \frac{hg(t,\hat{u}) + hg(t,u)}{2}, \tag{3-14}$$

si elegimos $\hat{u} = u$ evidentemente nuestro algoritmo se convierte en el algoritmo de Euler.

Esta observación nos hace evidente que el método puede variar los resultados en función del valor elegido para $\hat{u}$ el cual queremos finalmente igualarlo a u^{n+1} como meta para cada paso de cálculo. Este objetivo lo alcanzamos empleando tres estrategias importantes para buscar la convergencia del método a la solución.

La primer consideración que es necesario tomar, es la forma en que se elige inicialmente el valor de $\hat{u}$, ya que de este valor dependerá la calidad de nuestra primer predicción. No existe manera única de elegir dicho valor, sin embargo, tomando en cuenta la observación hecha a la ecuación 3-14, una opción bastante sencilla de implementar es efectivamente iniciar con $\hat{u} = u^n$.

La segunda consideración se encuentra en el momento que deseamos actualizar nuestras

predicciones, una adecuada actualización de nuestro valor $\hat{u}$ nos brinda un acercamiento veloz y confiable al valor de u^{n+1}. Una estrategia bastante sencilla de implementar consiste en actualizar $\hat{u}$ directamente con nuestra más reciente aproximación de u^{n+1} sin embargo, en la práctica, bajo ciertas condiciones este proceso puede generar errores considerables que impiden la convergencia a una buena aproximación.

Una estrategia alternativa y bastante confiable consiste en emplear una actualización theta

$$\hat{u}^* = \hat{u} + \theta(u^{n+1} - \hat{u}), \tag{3-15}$$

de esta forma, el algoritmo frena un poco el avance hacia la solución disminuyendo el riesgo de comportamientos oscilatorios. Una adecuada calibración de este parámetro puede brindar ventajas considerables para acelerar el proceso de cálculo.

Una vez que el algoritmo ha logrado la convergencia en el paso de tiempo actual, el tamaño de paso se incrementa al doble del paso anterior o un límite pre establecido, lo que resulte menor con la finalidad de recuperar la velocidad a la que avanza el método en el cálculo.

Mediante este proceso de corrección, el algoritmo genera una mejor aproximación en cada ciclo de cálculo, sin embargo, aún no hemos abordado las condiciones de paro que se emplean para decidir en que momento la aproximación es suficientemente buena. La primer dificultad que enfrentamos al querer calcular los errores de nuestra aproximación es que no tenemos contra que compararlo, ya que de antemano, no sabemos la solución exacta.

Para lidiar con esta dificultad, un método bastante empleado en algoritmos iterativos consiste en calcular el error entre aproximaciones sucesivas, de forma tal que cuando el error entre ellas cae por debajo de un umbral, se considera que el algoritmo ha concluido satisfactoriamente.

Otra consideración bastante común es emplear tanto errores relativos como absolutos para evaluar la condición de paro. Cuando evaluamos cosas muy pequeñas, una condición de error relativo resulta muy conveniente, por el contrario para cosas grandes, basta con un error absoluto para obtener buenos resultados. De esta forma, incluir condiciones que tomen en cuenta ambos

casos resulta muy conveniente, de esta forma, una condición de paro satisfactoria se da cuando

$$\|\hat{u} - \hat{u}^*\| < 2e + 2e\|\hat{u}^*\|, \tag{3-16}$$

donde e representa el error establecido como condición de paro.

Por si solo, este proceso no garantiza el tiempo de convergencia del algoritmo, ya que bajo ciertas circunstancias, el valor predicho de la aproximación puede oscilar alrededor de un punto aproximándose a este de manera muy lenta. Es por esta razón que se incluye una condición de adaptación en el tiempo para facilitar el proceso de cálculo, de esta manera, si tras un número fijo de aproximaciones no se ha llegado a la condición de paro establecida, el algoritmo corta por la mitad el paso de tiempo de forma que mejora la calidad de las predicciones alrededor de puntos difíciles, sin comprometer la velocidad de cálculo alrededor de otros puntos.

Capítulo 4

Simulaciones y resultados.

Debido al alto número de ecuaciones algebraicas que involucra aproximar la solución a una ecuación diferencial, en este caso la ecuación de Richards, para emplear un método de diferencias finitas es recomendable el uso de computadoras. Entre mayor sea la resolución que se desea obtener en el cálculo de las soluciones mayor es el poder de cómputo que se requiere para lograr este objetivo.

Uno de los mayores cuellos de botella se encuentra en el cálculo de los operadores de la ecuación diferencial, una forma bastante eficiente para disminuir este efecto es aprovechar la linealidad de los operadores diferenciales para facilitar el cálculo por medio del método descrito en la sección 3, por medio del cual es posible reducir al mínimo necesario el cálculo de los operadores por medio de la solución de sistemas matriciales.

Para el cómputo de datos con interés científico, existen una gran variedad de paquetes comerciales, uno de los más difundidos en ambientes académicos es Matlab[1], el cual tiene bastantes años en el mercado y cuenta con rutinas muy optimizadas para el cálculo en diversas disciplinas, entre las cuales se encuentra el álgebra lineal. Sin embargo, debido al costo de sus licencias, dentro de algunos ambientes académicos es difícil acceder a licencias de desarrollo.

De manera alternativa existen bastantes herramientas tanto libres como comerciales permiten el cómputo científico de manera sencilla para el usuario, algunas alternativas muy difundidas

[1] https://la.mathworks.com/

en áreas de investigación se encuentran en Fortran[2], C[3], Python[4], entre otros. Si bien, estas alternativas requieren un poco más de conocimiento en programación, resultan convenientes para reducir la cantidad de código final necesario para realizar diversas tareas de cómputo con fines científicos. Aunado a una reducción en el costo de licencias y en algunos casos aumento en la velocidad de ejecución del código.

4.1. Software, hardware y proceso de simulación

Inicialmente, el software presentado en este trabajo fue desarrollado empleando las herramientas de Matlab, sin embargo, con el tiempo se decidió probar otras alternativas como es Julia. Como se puede observar en la figura 4-1, Julia presenta un comportamiento consistentemente mas rápido que Matlab para distintas pruebas de velocidad realizadas con distintos lenguajes de programación.

Adicionalmente a la velocidad de cálculo, se decidió realizar pruebas del código empleando Julia debido a que es un lenguaje interpretado, es decir, el código no se compila sino que es directamente interpretado por las herramientas de Julia. Los resultados mostrados en esta tesis corresponden al código desarrollado en Julia versión v0.4.7 compilado para Linux Ubuntu 17.04 el cual se corrió sobre una computadora Dell inspiron N4110 con procesador intel core i5 y 4GB de memoria RAM.

Para realizar los comparativos del funcionamiento del algoritmo se realizaron varias pruebas las cuales se describen con mayor claridad y detalle en las siguientes secciones. Entre estas pruebas destacan la simulación empleando dos ecuaciones diferentes para describir el comportamiento del terraplén así como la simulación de dos materiales diferentes. También se realizó la simulación empleando dos medios de características diferentes dentro del mismo terraplén.

[2]LAPACK: http://www.netlib.org/lapack/, FORTRAN: https://www.fortran.com/compilers.html
[3]GNU Scientific Library: https://www.gnu.org/software/gsl/, gcc: https://gcc.gnu.org/
[4]SciPy: https://www.scipy.org/, Python: https://www.python.org/

Figura 4-1: Comparativo de tiempos de cálculo relativos a C para varios lenguajes de programación. Tomado de la página oficial de Julia.

4.2. Medios

Como se mencionó en el capítulo 2, una expresión usual para modelar la permeabilidad de un medio poroso está dada por la ecuación de Van Genuchten-Mualem, la cual, recordando la ecuación 2-10, está descrita por

$$k(\theta_n) = k_s\sqrt{\theta_n}\left(1-(1-\theta_n^{\frac{1}{m}})^m\right)^2, \tag{4-1}$$

a su vez, esta ecuación que depende del contenido normalizado de agua θ_n, puede variar dependiendo de la forma en que se modele esta última ecuación.

Como se mencionó previamente, en este trabajo se puede optar tanto por la ecuación de Van Genuchten para el contenido de agua como por la ecuación de Fredlund y Xing. Siguiendo la ecuación de Van Genuchten, la ecuación para el contenido de agua propuesta en la ecuación 2-11 es

$$\theta_n(s) = \frac{1}{((\alpha s)^n+1)^m}, \tag{4-2}$$

ecuación que a su vez da origen al módulo de almacenamiento de agua el cual está dado por

$$m_w = -mn\left(\theta_s-\theta_r\right)\frac{\alpha\left(\alpha s\right)^{n-1}}{\left((\alpha s)^n+1\right)^{m+1}}, \tag{4-3}$$

donde θ_s y θ_r corresponden al contenido volumétrico de agua saturado y residual respectivamente.

Tomando en cuenta estas tres ecuaciones previas, se realizó la simulación del terraplén descrito en la sección 3.1 usando mallas con refinamientos sólo en la región del hombro como se describe en la sección 3.2.

Evidentemente antes de proceder a la simulación del terraplén, antes es necesario definir las características del suelo en estudio, el terraplén de estudio pertenece a una autopista ubicada sobre suelo natural con un nivel freático en la base, lo cual impone condiciones de presión de poro cero en esta región.

Las condiciones iniciales se consideran en un estado estable donde la presión de poro inicial

está dada por la fórmula $u = -gy$, tomando en cuenta a u como la presión de poro. Para condiciones de simulación, se considera que las fronteras verticales así como la capa asfáltica presentan flujo nulo todo el tiempo. Mientras que la región del hombro y el terraplén se encuentran sometidos a una fina capa de agua que fija humedece la superficie y fija la presión de poro en 0.

Parámetro	Suelo
θ_r	0.04
θ_s	0.37
α(1/kPa)	0.89
n	1.57
k_s(m/día)	0.25

Tabla 4-1: Constantes del medio 1.

Las características descritas en la tabla 4-1 corresponden a las condiciones del modelo tomado en cuenta para el medio uno. Estas condiciones fueron tomadas en cuenta para cada una de las simulaciones como se describen en las secciones correspondientes.

Una forma de visualizar el comportamiento de cada medio es a través del estudio de las funciones que modifican a la ecuación diferencial que lo rige. En las figuras 4-2, 4-3 y 4-4 podemos observar como se modifican la permeabilidad, contenido de agua y módulo de almacenamiento respectivamente dependiendo de la presión de poro.

De estas gráficas podemos observar el rápido cambio que se presenta en las condiciones del terraplén entre las presiones de poro comprendidas entre los $0kPa$ y $-10kPa$, región donde es muy marcado el rápido incremento en la permeabilidad a la vez que se da un brusco descenso en el módulo de almacenamiento de agua. Todo esto se ve reflejado en cambios más rápidos del contenido volumétrico de agua cuando la presión de poro de encuentra alrededor de $-5kPa$.

Retomando el trabajo de Fredlund y Xing, una manera alternativa de modelar las ecuaciones que describen el comportamiento del contenido volumétrico de agua, es por medio de la

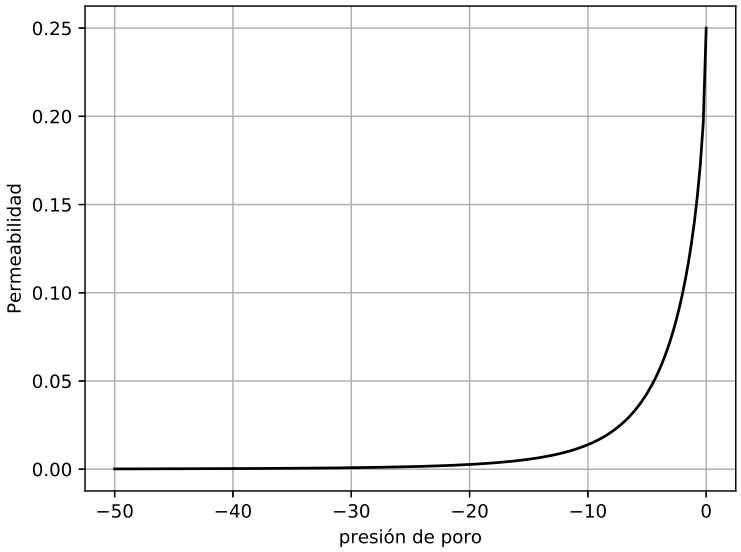


Figura 4-2: Gráfica de permeabilidad en función de la presión de poro, correspondiente al medio 1.

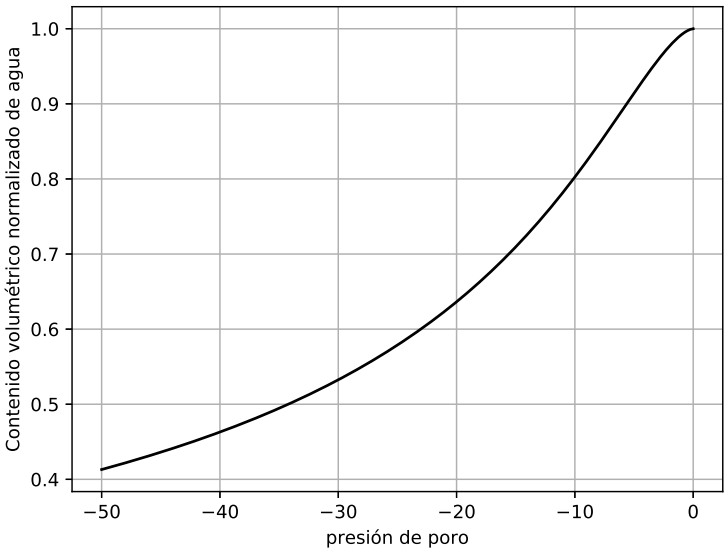


Figura 4-3: Gráfica normalizada del contenido volumétrico de agua en función de la presión de poro, correspondiente al medio 1.

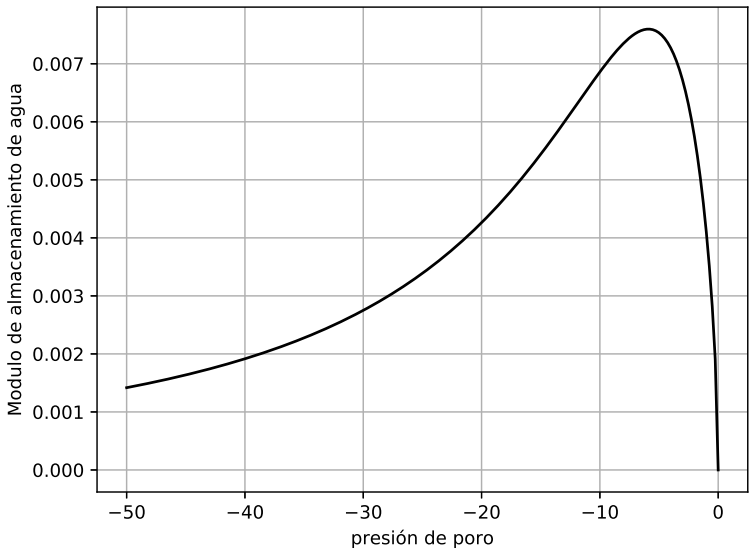


Figura 4-4: Gráfica del módulo de almacenamiento de agua como función de la presión de poro, correspondiente al medio 1.

expresión 2-12, la cual está descrita por medio de la ecuación

$$\theta_n(s) = \frac{1}{(ln(e + (\frac{s}{a_f})^{n_f}))^{m_f}}. \tag{4-4}$$

Así mismo, retomando la definición del módulo de almacenamiento de agua con la ecuación previa damos origen a la expresión

$$m_w = -\frac{s^{n_f-1}}{\alpha_f^{n_f}} \frac{m_f n_f}{e + (\frac{s}{\alpha_s})^{n_f}} \frac{\theta_s - \theta_r}{\left(\ln\left(e + \left(\frac{s}{\alpha_f}\right)^{n_f}\right)\right)^{m_f}}, \tag{4-5}$$

la cual en conjunto con las ecuaciones 4-4 y 4-1 nos permiten modelar de igual forma el comportamiento de un material poroso, de esta forma, en conjunto con los valores descritos en la tabla 4-2 y las mismas condiciones de frontera para el medio uno tenemos definidas las condiciones del modelo y la simulación del medio 1B.

Observando las gráficas correspondientes, podemos notar las mayores diferencias que se encuentran en la permeabilidad de los medios, siendo el primero un material tan poco permeable

Parámetro	Suelo
θ_r	0.0001
θ_s	0.4
α(1/kPa)	0.89
n	2.544
α_f	5
m_f	1
n_f	2
k_s(m/día)	0.864

Tabla 4-2: Constantes del medio 1B.

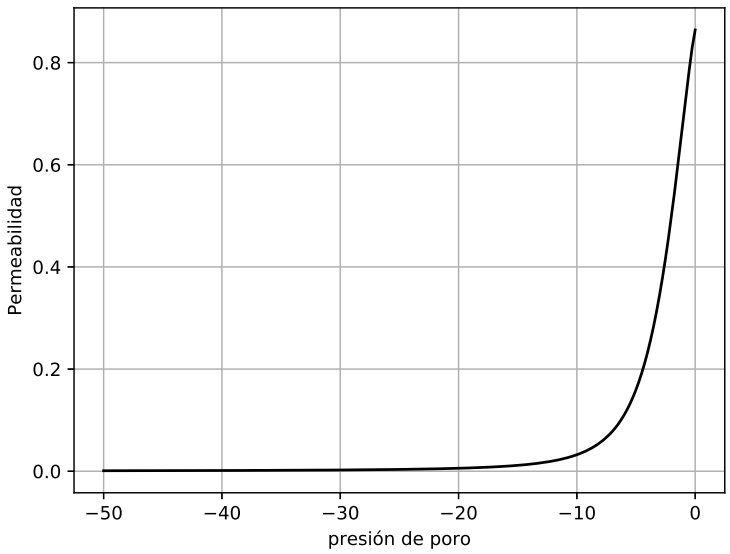


Figura 4-5: Gráfica de permeabilidad en función de la presión de poro, correspondiente al medio 1B.

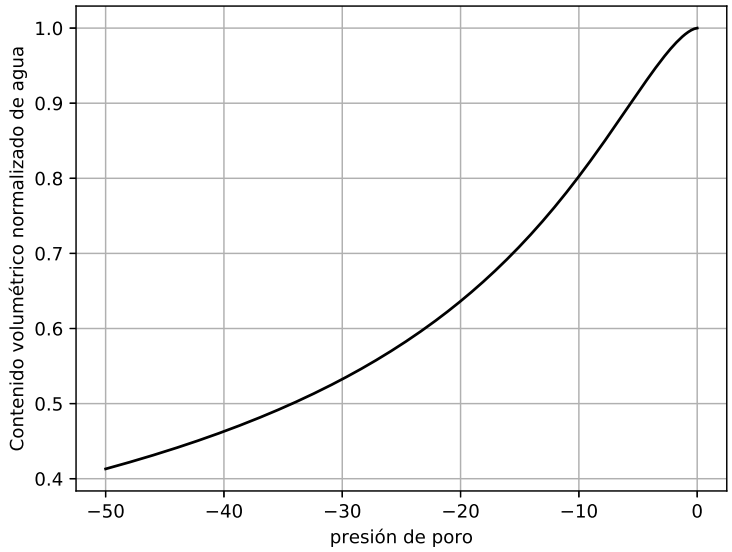


Figura 4-6: Gráfica normalizada del contenido volumétrico de agua en función de la presión de poro, correspondiente al medio 1B.

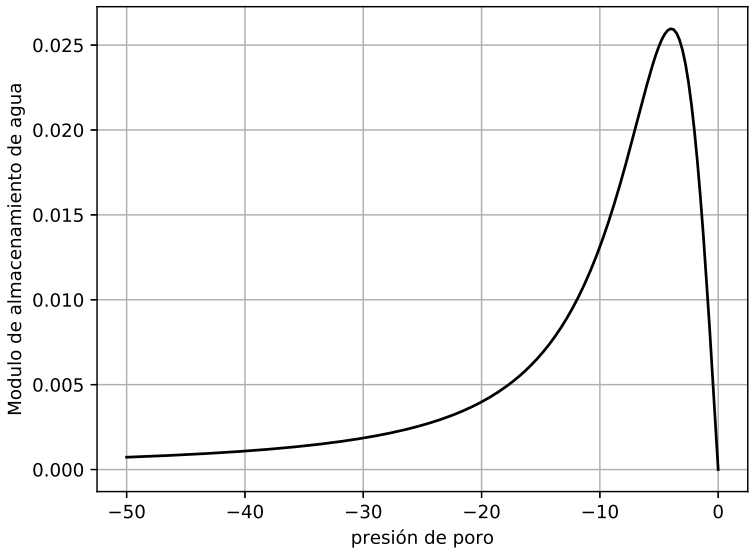


Figura 4-7: Gráfica del módulo de almacenamiento de agua como función de la presión de poro, correspondiente al medio 1B.

como un limo mientras que el segundo presenta significativamente mayor permeabilidad como una arena. Otra característica importante es la facilidad que presenta el segundo medio para des-saturar el material al permitir un menor contenido volumétrico de agua para llegar a un contenido residual, esto se debe a la mayor presencia de huecos de aire y mayor dificultad que presenta para permitir la conducción capilar.

Para estudiar las características que presenta el algoritmo ante medios diferentes que interactúan entre si, se modeló un tercer medio, de forma que se puedan estudiar estas interacciones. Para este estudio se empleó el modelo de Van Genuchten con un medio de características menos permeables que permitan observar los cambios que se producen al introducir esta interfaz.

Parámetro	Suelo
θ_r	0.054
θ_s	0.355
α(1/kPa)	0.00102
n	1.43
k_s(m/día)	$4.32e^{-6}$

Tabla 4-3: Constantes del medio 2.

Las características con que se dotó a este medio están descritas en la tabla 4-3 donde se puede notar el significativo cambio en la permeabilidad, correspondiente a una arcilla por sus características de permeabilidad y contenido residual de agua principalmente.

En las gráficas correspondientes a este medio se puede observar el suave cambio que producen estas constantes en las presiones de poro que se manejan como condición inicial para estas pruebas.

4.3. Mallas

Como se mencionó en la sección 2, la calidad de las mallas es un tema bastante discutido y sin consenso, adicionalmente, se dijo en la sección previa que las mallas empleadas para este trabajo corresponden a mallas lógicamente rectangulares debido a la facilidad que implica su manejo y la facilidad que tienen para generarse dentro del ambiente escolar. Si bien, la calidad

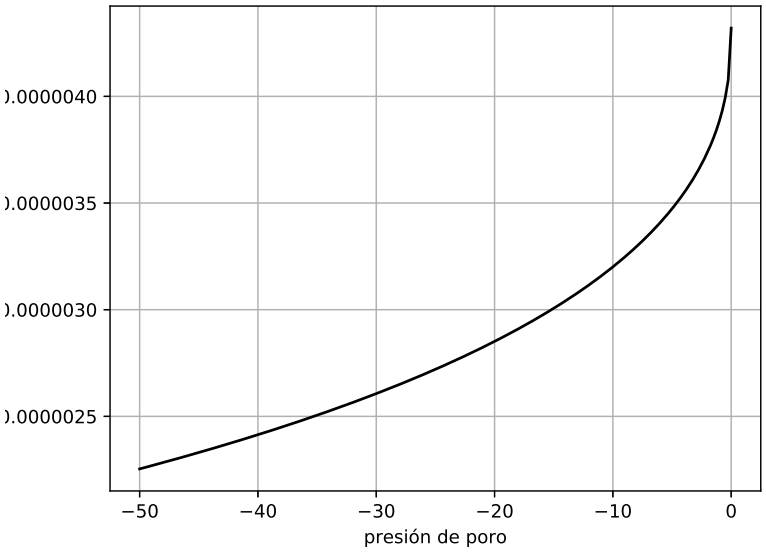


Figura 4-8: Gráfica de permeabilidad en función de la presión de poro, correspondiente al medio 2.

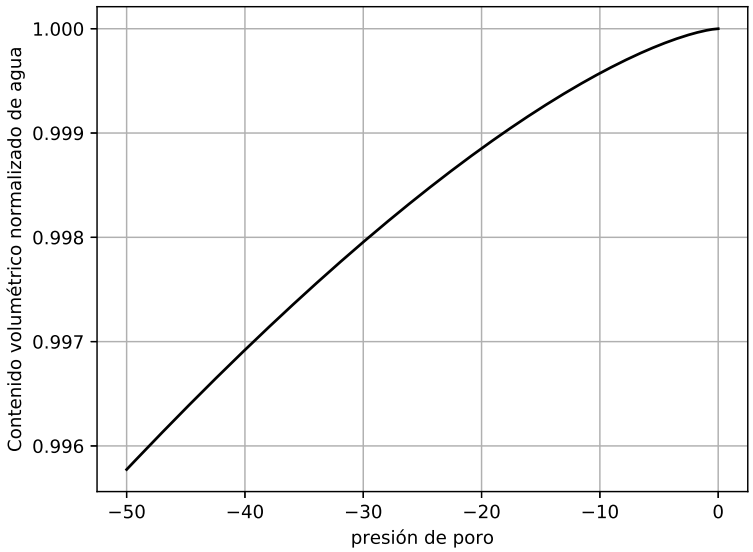


Figura 4-9: Gráfica normalizada del contenido volumétrico de agua en función de la presión de poro, correspondiente al medio 2.

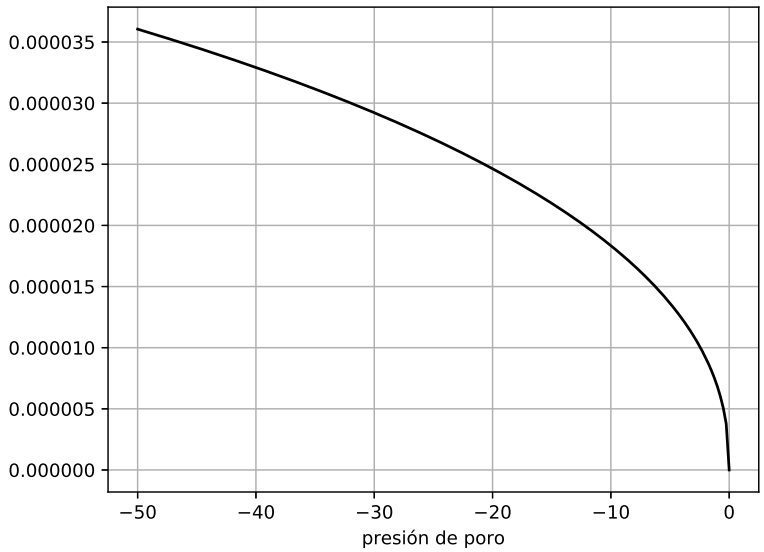


Figura 4-10: Gráfica del módulo de almacenamiento de agua como función de la presión de poro, correspondiente al medio 2.

de las mallas generadas para este trabajo no son óptimas, resultan convenientes para mostrar la estabilidad o robustez del método ante condiciones adversas, así como mostrar que aún en condiciones de baja calidad los resultados pueden ser buenos para la simulación.

Para este trabajo se realizaron simulaciones bajo diversas condiciones, las primeras de ellas fueron realizadas en condiciones uniformes con un sólo medio, en ellas se empleó la malla descrita en la figura 4-11 empleando hasta tres refinamientos diferentes de la malla para motivos de comparación, en la figura se muestra la malla correspondiente a 41 divisiones en cada eje, generando como resultado mallas de 41×41, 81×81 y 161×161 puntos.

La gráfica correspondiente a la calidad de estas mallas se puede ver en la figura 4-12 donde se observa por medio de curvas de nivel la disminución que presenta la calidad al acercarse en dirección al talud a una altura de 4.5 metros y por el contrario, la calidad mejora considerablemente al acercarse a la región de frontera un metro por debajo del asfalto así como al ubicarse en la región dos, la parte que corresponde al suelo natural.

Por otro lado, ya que deseamos estudiar el comportamiento del método ante medios no uniformes donde interaccionan dos medios se realizaron simulaciones empleando la malla descrita

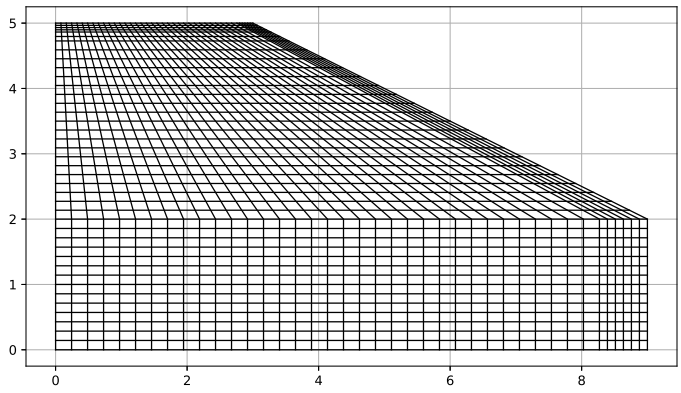


Figura 4-11: Malla empleada para simulaciones en medio único.

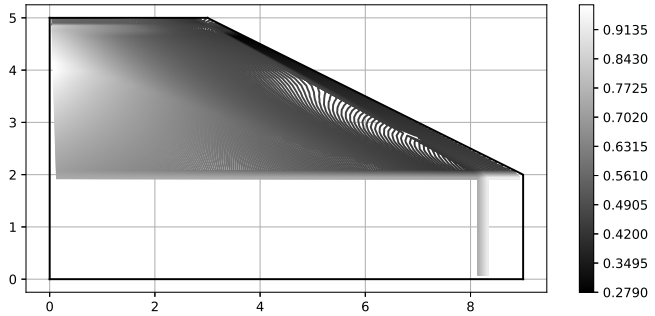


Figura 4-12: Calidad de la malla resultante en curvas de nivel.

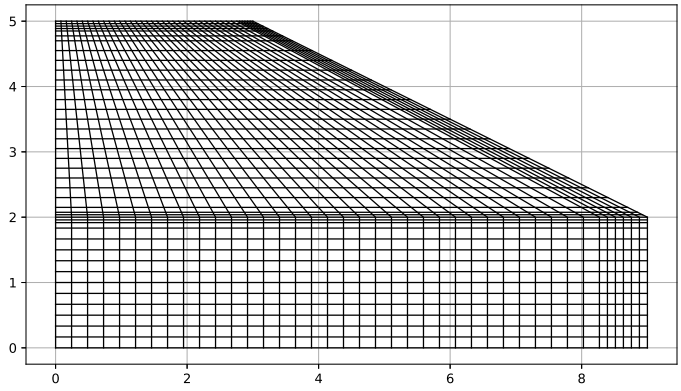


Figura 4-13: Malla empleada para simulaciones en dos medios.

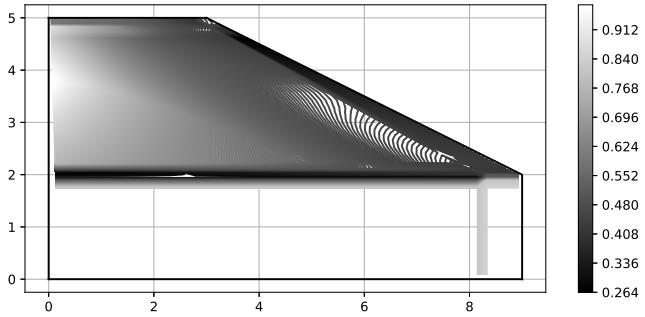


Figura 4-14: Calidad de la malla resultante mostrada mediante curvas de nivel.

en la figura 4-13 donde se puede observar los dos niveles adicionales de refinamiento que se incluyen para mejorar los resultados de las simulaciones en la interfaz entre la región uno y la región dos, en este caso sólo se realizaron simulaciones con la malla de 41×41 puntos. El comportamiento de la calidad en esta malla se observa similar al caso descrito para un medio, el cual se puede observar en la figura 4-14, donde se muestra de manera evidente el cambio que surge entre las dos mallas debido al refinamiento adicional que tienen.

Si bien, se observa en ambos casos que los refinamientos producen una disminución de la calidad, estos son necesarios para capturar los fenómenos que se producen en las regiones de interfaz. Al encontrarse alargadas en una dirección, estas mallas no capturan de manera óptima los fenómenos que se propagan a lo largo de esta dirección, sin embargo, al estar compactadas en la otra dirección, los cambios que se propagan sobre esta dirección se pueden capturar de mejor manera, en la sección siguiente se detallan cuales son estas características que se desean estudiar de mejor manera y que nos motivan a inducir estos refinamientos.

4.4. Medio 1

Como se describió en los apartados previos, en cada medio se realizaron varias pruebas para verificar el funcionamiento de los métodos. En la figura 4-15 podemos observar el comportamiento inicial del terraplén tras 14.4 minutos de someterse a una fina capa de agua en la zona

del terraplén y hombro las cuales son capaces de absorber esta humedad. Pese a que se da un muy rápido proceso de absorción en éstas áreas, aún es muy superficial el área afectada debido a la baja permeabilidad del medio.

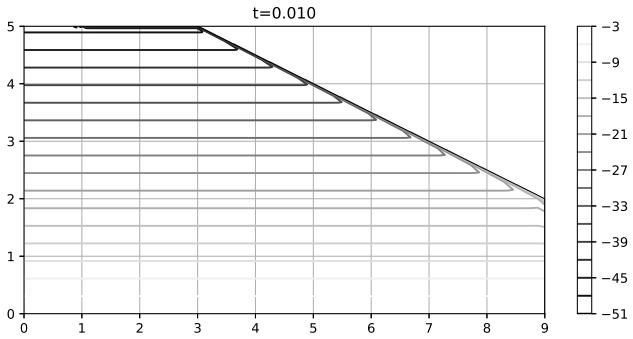


Figura 4-15: Presión de poro en el tiempo, malla de 41×41 en el medio 1. Tiempo medido en días.

Por su parte, en las figuras 4-16 y 4-17 se observa el comportamiento de la malla de 41×41 puntos donde se puede apreciar el proceso de difusión de la humedad, el cual evidentemente no es lineal y se da de manera lenta en las zonas con menor presión de poro, mientras que se produce de manera acelerada cuando la presión esta por encima de los $-10kPa$.

En este punto, el efecto del rápido cambio en la humedad ya no es perceptible, ya que tras 7.2 y 16.8 horas el efecto que predomina es la difusión, así como un débil efecto de advección producido por la gravedad.

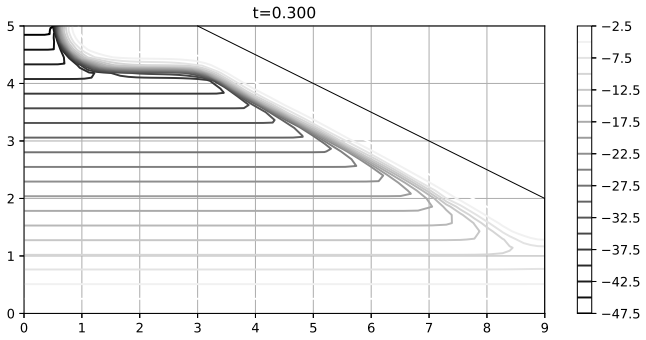


Figura 4-16: Presión de poro en el tiempo, malla de 41×41 en el medio 1. Tiempo medido en días.

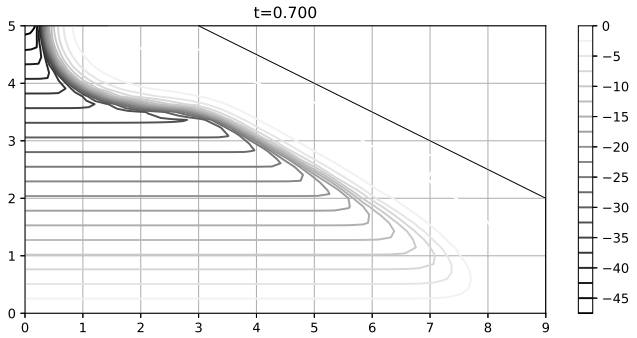


Figura 4-17: Presión de poro en el tiempo, malla de 41×41 en el medio 1. Tiempo medido en días.

En las figuras 4-18 y 4-19 observamos el efecto de estudio que se esta buscando por medio de las simulaciones, transcurridas 24 y 36 horas respectivamente a partir del inicio del fenómeno, vemos como la humedad se está minando hacia el interior de la carretera.

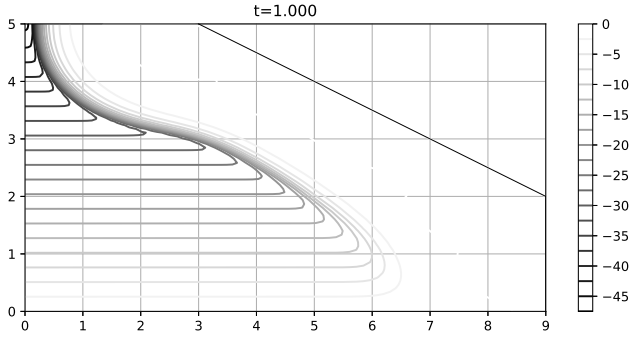


Figura 4-18: Presión de poro en el tiempo, malla de 81×81 en el medio 1. Tiempo medido en días.

Estudiando las principales funciones que intervienen en el comportamiento de la difusión de humedad dentro del terraplén, en la figura 4-20 el comportamiento de la permeabilidad tras 16.4 horas de infiltración, se observa como en las zonas de menor presión ésta permeabilidad es baja dando un cambio brusco algunos pascales antes de saturarse. En la figura 4-20 se observa el comportamiento del módulo de almacenamiento de agua el cual cae rápidamente a cero en las zonas donde se ha alcanzado la saturación, mientras que en la figura 4-21 se observa el comportamiento del contenido volumétrico de agua.

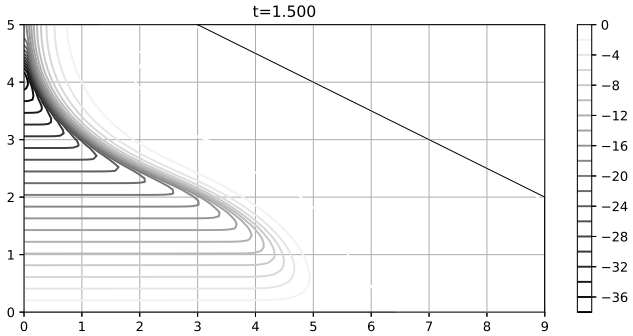


Figura 4-19: Presión de poro en el tiempo, malla de 81×81 en el medio 1. Tiempo medido en días.

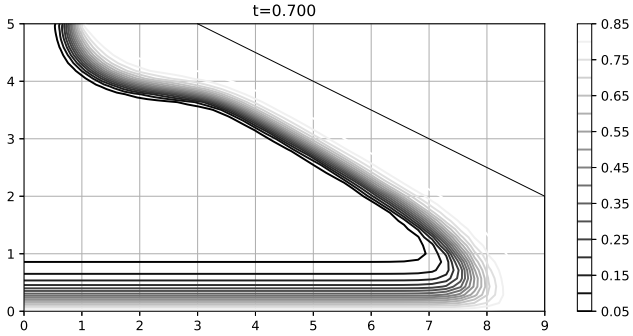


Figura 4-20: Gráfica de permeabilidad en el tiempo $t = 0.7$.

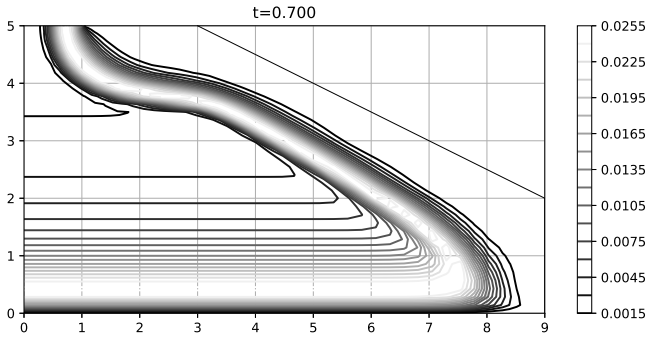


Figura 4-21: Gráfica del módulo de almacenamiento de agua en el tiempo $t = 0.7$.

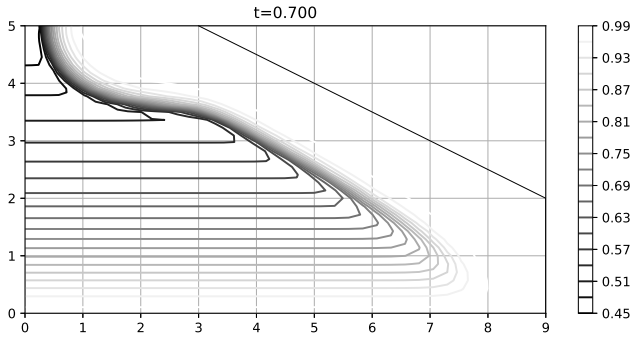


Figura 4-22: Gráfica de contenido volumétrico de agua normalizada en el tiempo $t = 0.7$.

Para poder comparar numéricamente los resultados obtenidos se realizó un corte vertical del terraplén a cinco metros, sobre este corte se interpolaron doscientos puntos en donde está reflejada la presión de poro para distintos tiempos de simulación. En la figura 4-23 se observa el comportamiento de esto puntos en una gráfica de presión en el eje x contra altura en el eje y graficando en negro los resultados simulando con un paso de tiempo máximo de $h = 0.025$ días mientras que en gris se graficó con un paso del doble de resolución.

Visualmente ambas gráficas se corresponden sin error, sin embargo para mostrar realmente la diferencia entre ambas resoluciones, en la figura 4-24 se muestra el error entre simulaciones sucesivas que varían por mitad en la resolución temporal. Similar a la gráfica anterior, se puede observar en negro la diferencia entre la primer y segunda resolución en valor absoluto, mientras que en color gris, se observa amplificado 4 veces el error entre la segunda y la tercer resolución.

Como se observa en las gráficas, en la parte superior del terraplén se concentran las mayores variaciones debido a la calidad de la malla, mientras que en las partes inferiores el error disminuye a la cuarta parte al refinar en el tiempo.

Para disminuir el efecto de la interpolación en la visualización del error, se eligieron 3 puntos cercanos al corte a alturas de 2, 2.5 y 3 metros, los cuales se graficaron con respecto al tiempo para dos resoluciones temporales, de esta manera se puede obtener una mejor imagen del error. En la figura 4-26 observamos el comportamiento del error, el cual tiende a disminuir alrededor de 4 veces salvo el área cercana al talud donde la calidad de la malla es menor.

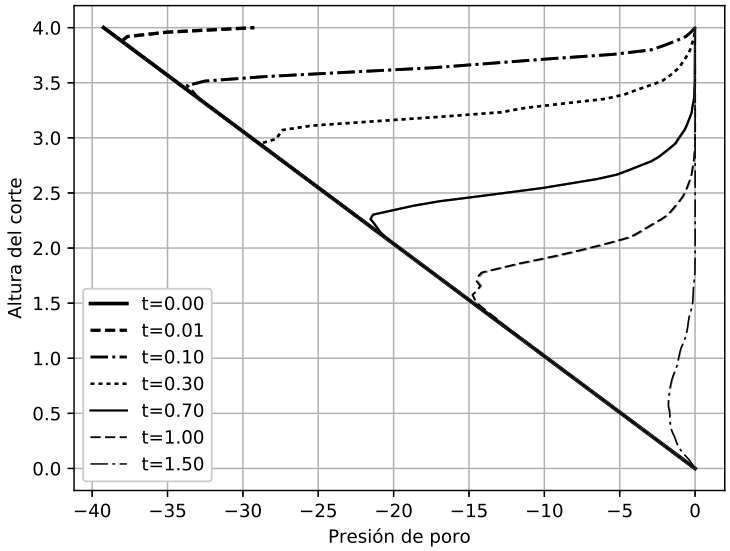


Figura 4-23: Corte vertical de la malla a 5 metros. Tomado por interpolación lineal.

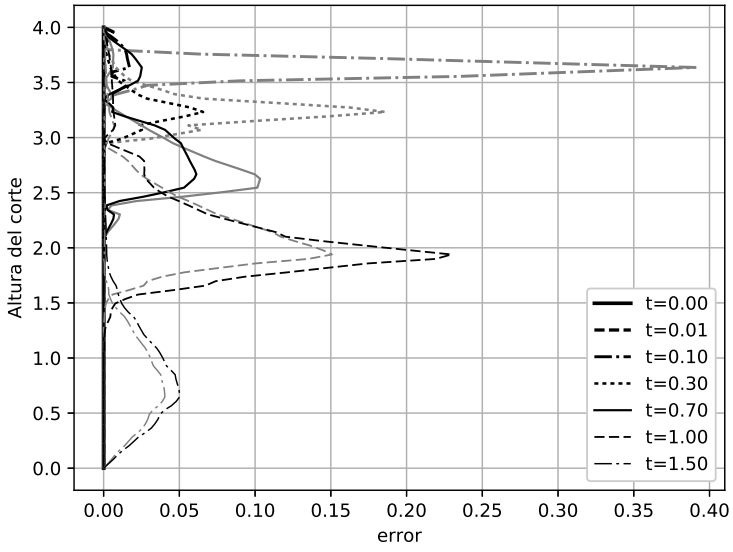


Figura 4-24: Error entre diferentes refinamientos temporales sobre el corte vertical a 5 metros. En negro se muestra el error $|u_h - u_{\frac{h}{2}}|$, en gris $4|u_{\frac{h}{2}} - u_{\frac{h}{4}}|$.

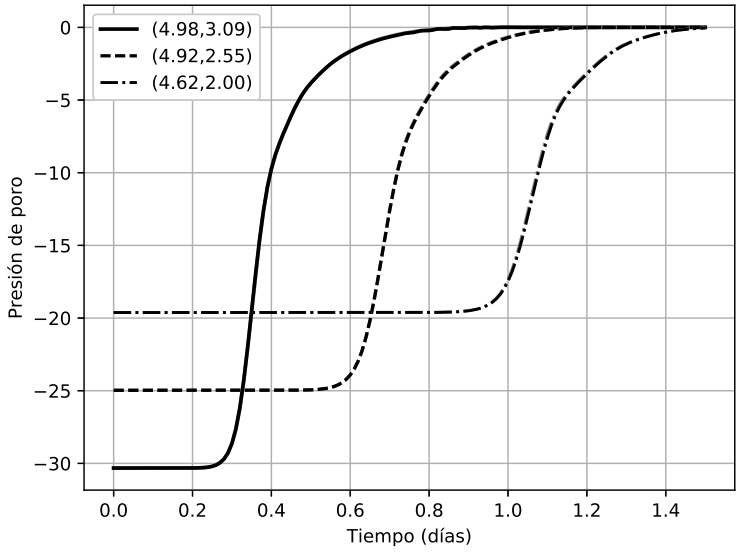


Figura 4-25: Seguimiento en el tiempo de varios puntos sobre la malla.

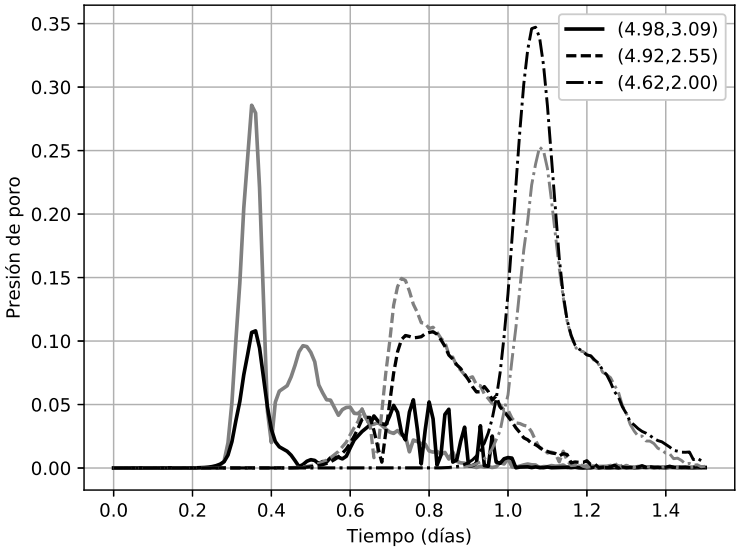


Figura 4-26: Seguimiento del error en el tiempo para varios puntos de la malla ante diferentes refinamientos temporales. En negro se muestra el error $|u_h - u_{\frac{h}{2}}|$ mientras que en gris $4|u_{\frac{h}{2}} - u_{\frac{h}{4}}|$.

4.5. Medio 1B

Como se describió en secciones anteriores, para verificar el comportamiento numérico del algoritmo, se puso a prueba ante diferentes medios y ante diferentes ecuaciones, de esta forma modelando el contenido volumétrico de agua por medio de la ecuación de Fredlund y Xing llegamos a las ecuaciones del medio B, el cual como sabemos, debido a las condiciones del suelo que se está modelando se espera una mayor permeabilidad. En la figura 4-27 se observa la simulación en la malla de 41×41 puntos para un tiempo de 14.4 minutos, aún no es muy perceptible a simple vista el efecto de la permeabilidad, sólo se observan los efectos de las condiciones de frontera.

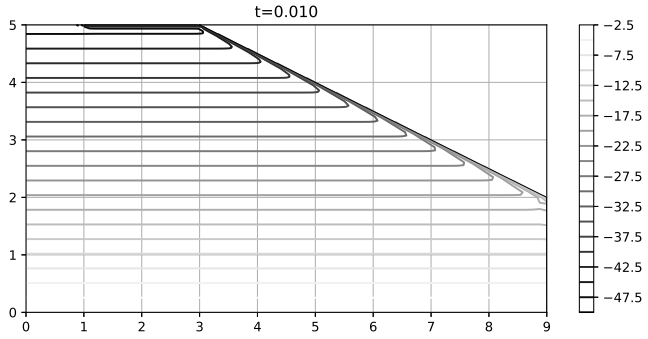


Figura 4-27: Presión de poro en el tiempo, malla de 41×41 en el medio 1B. Tiempo medido en días.

En las figuras 4-28 y 4-29 podemos observar el comportamiento de la infiltración, como se esperaba es más rápido que en el medio 1; estas figuras muestran el comportamiento en la malla de 41×41 puntos tras 2.4 y 7.2 horas respectivamente.

Las mallas de 81×81 puntos, muestran el mismo comportamiento que las mallas anteriores, mostrando claramente el efecto de la infiltración mayor al medio 1, lo cual permite saturar el terraplén con mayor velocidad.

Algo que es necesario recalcar es la mayor velocidad con que el algoritmo logra converger a la solución de esta segunda ecuación al requerir refinar menor cantidad de veces el tiempo por medio del paso adoptivo. Esto se traduce en menor número de evaluaciones de las matrices con

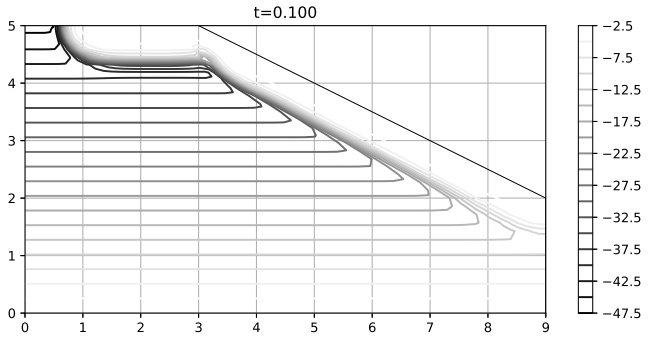


Figura 4-28: Presión de poro en el tiempo, malla de 41×41 en el medio 1B. Tiempo medido en días.

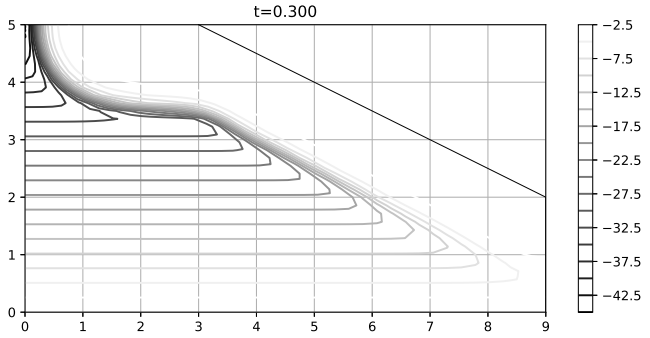


Figura 4-29: Presión de poro en el tiempo, malla de 41×41 en el medio 1B. Tiempo medido en días.

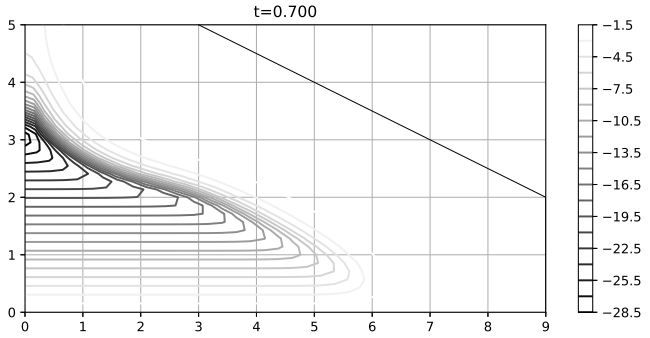


Figura 4-30: Presión de poro en el tiempo, malla de 81×81 en el medio 1B. Tiempo medido en días.

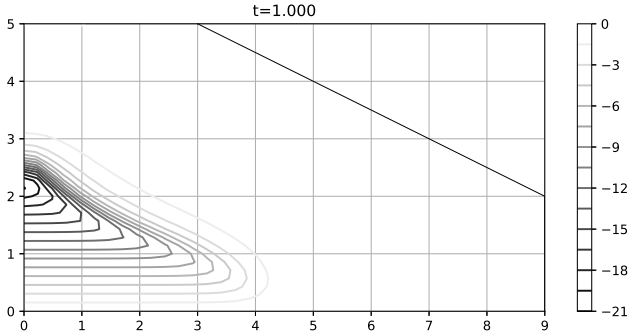


Figura 4-31: Presión de poro en el tiempo, malla de 81×81 en el medio 1B. Tiempo medido en días.

lo cual se llega a la solución en menos pasos.

En las figuras 4-32, 4-33 y 4-34 respectivamente, podemos observar la permeabilidad, el módulo de almacenamiento de agua y el contenido volumétrico de agua. En estas gráficas se puede percibir más notoriamente el cambio que producen las diferentes condiciones del medio, presentando un gradiente más suave tanto en la permeabilidad como en el módulo de almacenamiento.

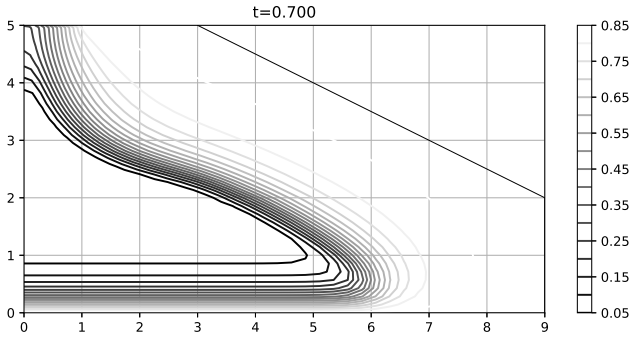


Figura 4-32: Gráfica de permeabilidad en el tiempo $t = 0.7$.

Nuevamente, para realizar una adecuada comparación numérica de los resultados, es conveniente comparar los errores que se producen, de nuevo, debido a que no existe un solución analítica a esta ecuación se comparan los resultados entre diferentes refinamientos. Para llevar a cabo esta comparación, se tomó nuevamente un corte vertical a 5 metros sobre el cual se interpolaron de manera lineal 200 puntos a igual distancia, los cuales se comparan entre distintos

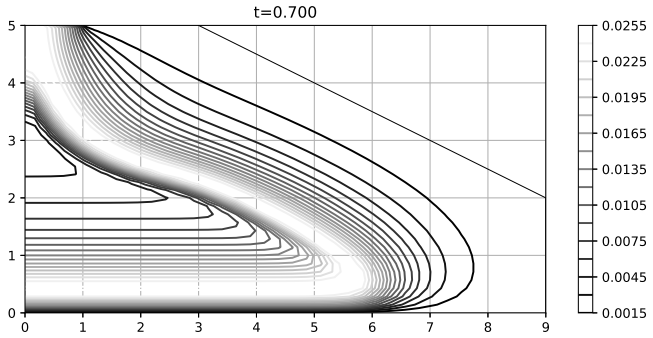


Figura 4-33: Gráfica del módulo de almacenamiento de agua en el tiempo $t = 0.7$.

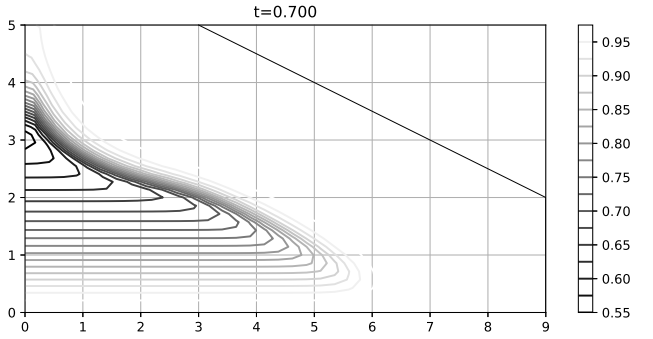


Figura 4-34: Gráfica de contenido volumétrico de agua normalizada en el tiempo $t = 0.7$.

refinamientos.

En la figura 4-35 observamos dos cortes entre diferentes refinamientos temporales con un tamaño de paso $h = 0.05$ en negro y $\frac{h}{2} = 0.025$ en gris. Visualmente la diferencia es nula por lo cual requerimos calcular los errores entre aproximaciones sucesivas, con lo cual en la figura 4-36 se observan graficados los errores $|u_h - u_{\frac{h}{2}}|$ en color negro y $4|u_{\frac{h}{2}} - u_{\frac{h}{4}}|$ en color gris, lo cual hace evidente las complicaciones que presenta el método ante las condiciones de frontera y la calidad de la malla; complicaciones que se van disipando conforme transcurre la simulación y la región de mayores cambios se empieza a acercar a una malla da mayor calidad.

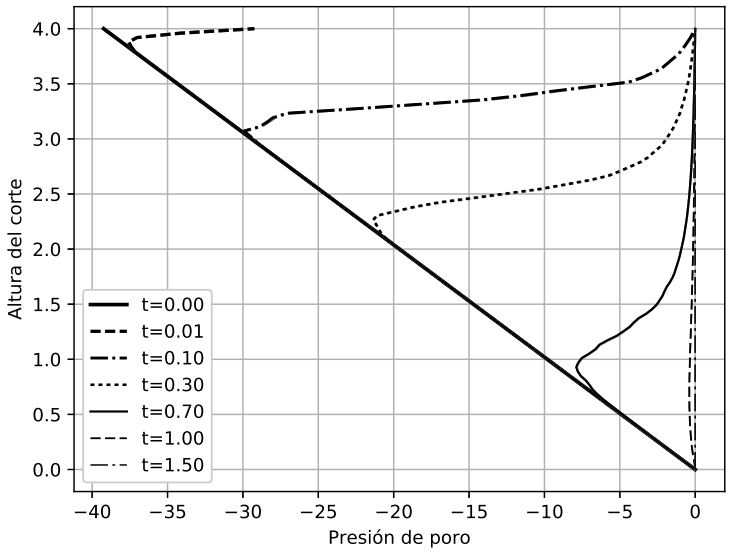


Figura 4-35: Corte vertical de la malla a 5 metros. Tomado por interpolación lineal.

Similar a las observaciones que se realizaron en las figuras antes mencionadas, se eligieron tres puntos cercanos al corte vertical para comparar sus resultados sin emplear interpolaciones, estos puntos están graficados en la figura 4-37 mostrando en el eje de las x el tiempo y en el otro eje la presión de poro nuevamente para dos resoluciones diferentes.

Para observar los errores entre aproximaciones, se graficó en la figura 4-38 los errores correspondientes, de aquí se observa que similar a la figura 4-36, los errores son grandes en las regiones cercanas al talud mientras que estos disminuyen a conforme transcurre el tiempo,

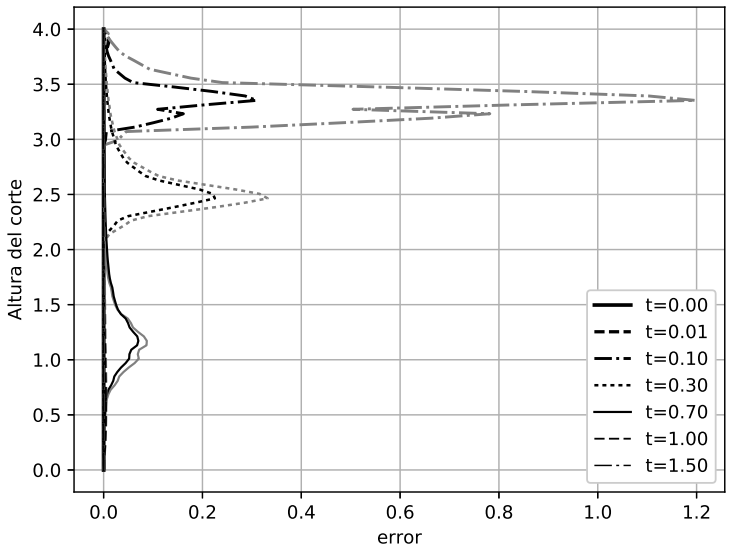


Figura 4-36: Error entre diferentes refinamientos temporales sobre el corte vertical a 5 metros. En negro se muestra el error $|u_h - u_{\frac{h}{2}}|$, en gris $4|u_{\frac{h}{2}} - u_{\frac{h}{4}}|$.

siendo nuevamente de una cuarta parte con respecto a las aproximaciones previas.

4.6. 2 medios uniformes

Uno de los puntos importantes en la simulación es el estudio de la interacción entre dos medios, para poder analizar y comparar entre si el comportamiento de dos medios, una herramienta de gran utilidad es comparar la solución en un medio y comparar los resultados obtenidos en dos medios iguales contra esta solución ya estudiada.

De esta forma, se implementaron las ecuaciones correspondientes a la región de interfaz dotando a la región uno como la región dos de ecuaciones idénticas que los rigen, en este caso ambas regiones se simularon con la ecuación de Van Genuchten-Mualem y se dotaron de las mismas propiedades correspondientes al medio 1. Así la aproximación resultante puede ser comparada directamente con la aproximación resultante de la simulación del medio 1.

En la figura 4-39 observamos el resultado de esta aproximación comparando el corte vertical

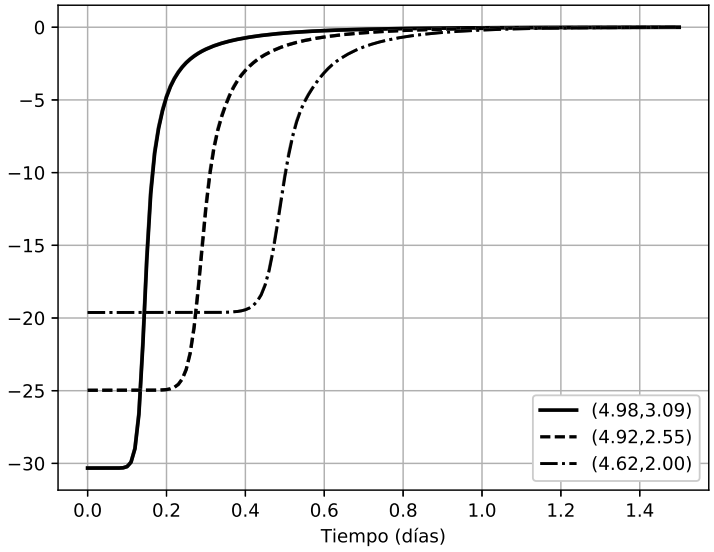


Figura 4-37: Seguimiento en el tiempo de varios puntos sobre la malla.

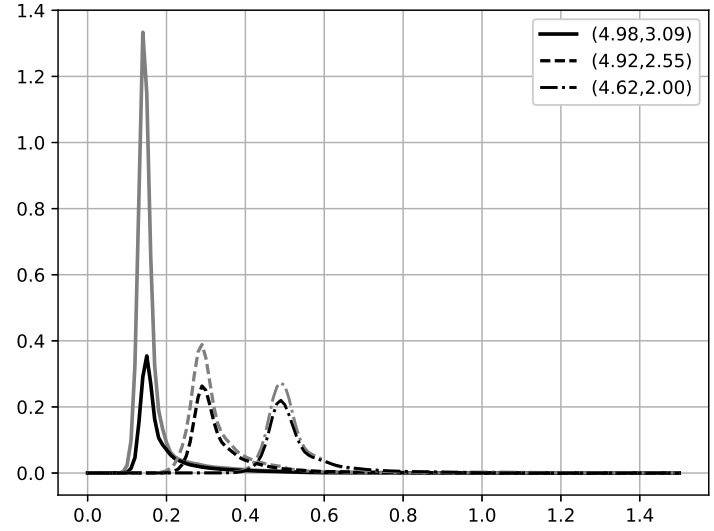


Figura 4-38: Seguimiento del error en el tiempo para varios puntos de la malla ante diferentes refinamientos temporales. En negro se muestra el error $|u_h - u_{\frac{h}{2}}|$ mientras que en gris $4|u_{\frac{h}{2}} - u_{\frac{h}{4}}|$.

que hemos empleado, el cual se tomó en siete momentos del tiempo. A diferencia de gráficas anteriores, en esta gráfica podemos observar algunas diferencias mayores, las cuales en gran manera se deben a la malla, la cual ahora se encuentra ubicada en puntos ligeramente diferentes.

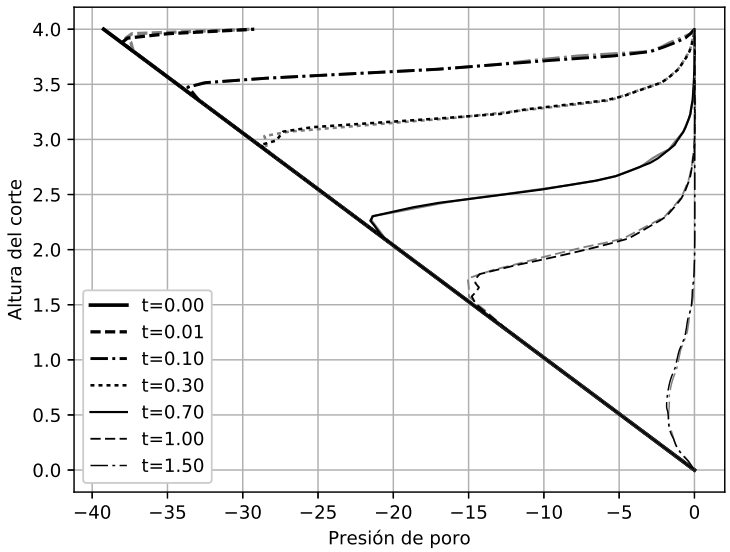


Figura 4-39: Comparativo entre dos medios de iguales características. En negro la solución empleando un sólo medio para el terraplén completo; en gris la solución empleando dos medios de iguales características.

Nuevamente, un método visual no es del todo válido para comparar resultados, por lo que al igual que en comparaciones anteriores se procede a revisar numéricamente el valor absoluto del error entre ambas aproximaciones, de esta manera en la figura 4-40 se observa el resultado de esta comparación. Se observa el rápido efecto de las condiciones de frontera que se va disipando poco a poco, una vez transcurrido el efecto transitorio de estas condiciones el error del uno por ciento.

Nuevamente para disipar el efecto de la interpolación se eligieron cuatro puntos sobre la malla los cuales se graficaron para observar el error entre simulaciones a lo largo del tiempo. Debido a que las mallas ahora son diferentes entre si, se eligieron los puntos dentro de la región

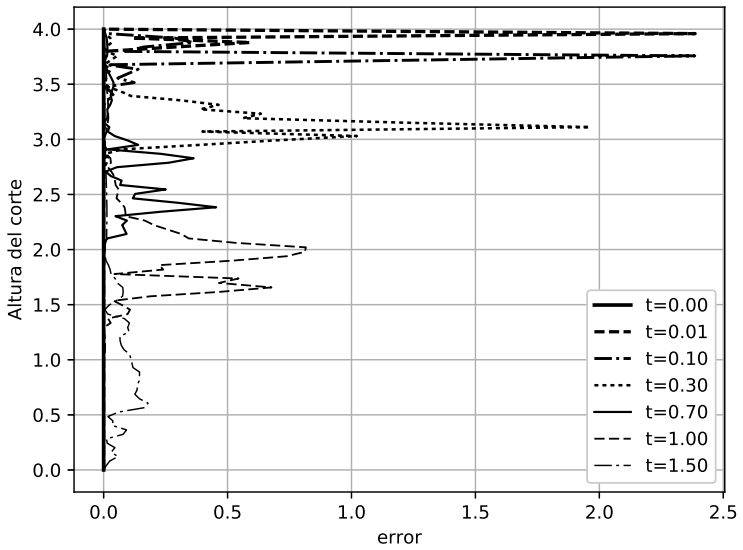


Figura 4-40: Error entre soluciones de uno y dos medios sobre el corte vertical.

de interfaz, la cual si coincide en ambas mallas. En la figura 4-41 se observa el comportamiento en el tiempo del error, el cual tiende a cero tras una fase de transición en todos los puntos.

4.7. Dos medios

Por último, para observar el comportamiento del algoritmo ante dos medios radicalmente diferentes, se simuló con la ecuación de Van Genuchten el comportamiento dotando a la región uno de las propiedades del medio 1 mientras que se dotó a la región dos con las propiedades descritas para el medio 2.

Se incluyeron las condiciones iniciales y de frontera que se simularon en las aproximaciones anteriores y se dejó correr el algoritmo para simular día y medio de infiltración. El resultado de este proceso se puede observar en las figuras de la 4-42 a la 4-45, donde se aprecia el efecto del material poco permeable cambiando la dinámica de la infiltración para la región uno a su vez que ésta misma se empieza a saturar en una región muy estrecha cercana a la región de interfaz.

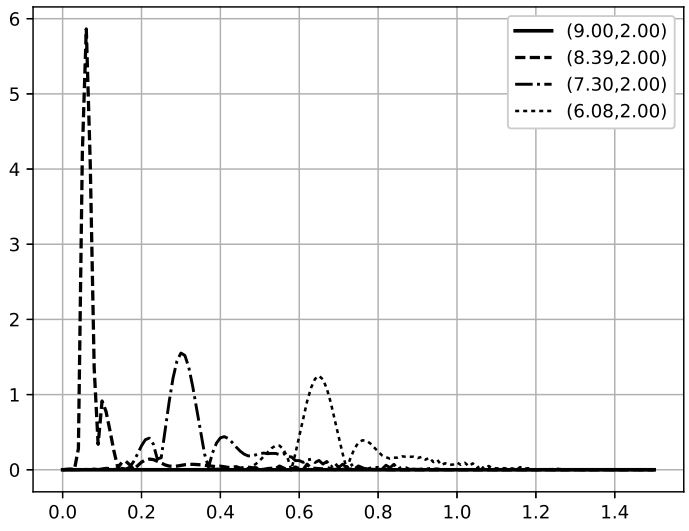


Figura 4-41: Error observado a lo largo del tiempo para algunos puntos en la región de interfaz.

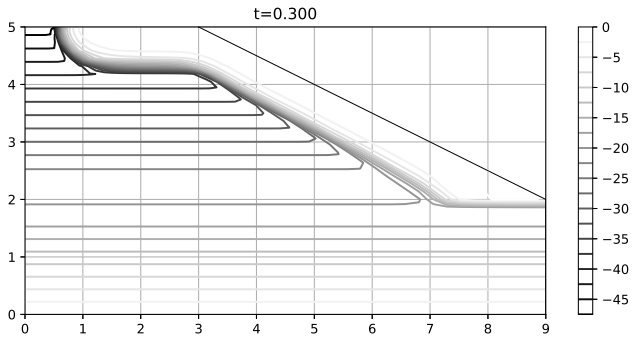


Figura 4-42: Simulación en dos medios diferentes en $t = 0.3$

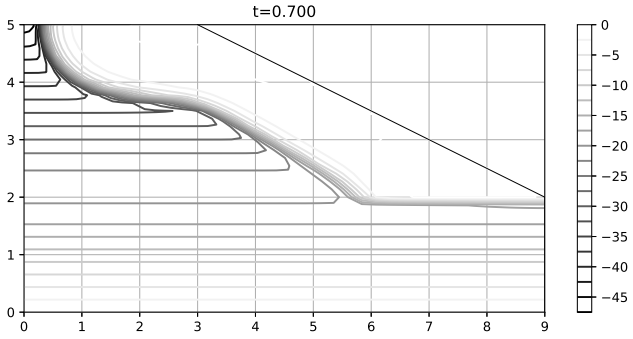


Figura 4-43: Simulación en dos medios diferentes en $t = 0.7$

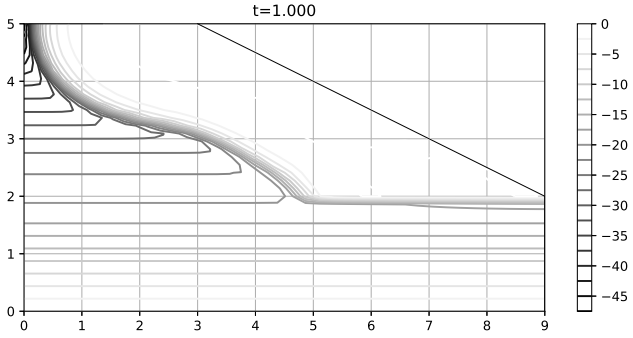


Figura 4-44: Simulación en dos medios diferentes en $t = 1.0$

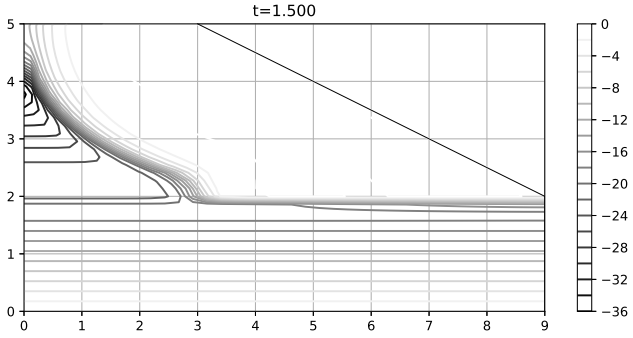


Figura 4-45: Simulación en dos medios diferentes en $t = 1.5$

Para observar el efecto de la ecuación con que se modeló la región de interfaz, se realizó nuevamente un corte para estudiar el comportamiento de la presión de poro a lo largo del corte en varios momentos de tiempo, en la figura 4-46 se puede observar los resultados de este proceso.

En este corte se puede observar el comportamiento esperado del proceso de infiltración, salvo en un momento antes de saturarse la región aledaña a la interfaz, en este punto se observa un descenso de la presión de poro, lo cual no corresponde al comportamiento esperado y podría deberse a la malla, lo cual requiere un poco más de estudio.

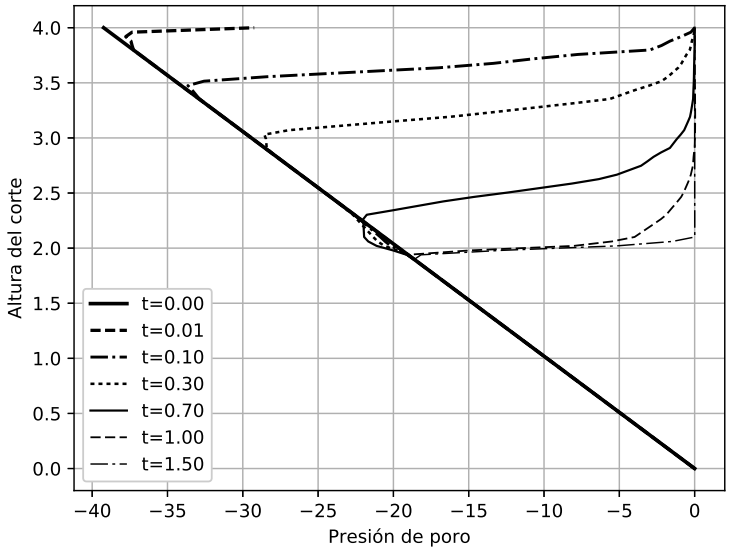


Figura 4-46: Corte realizado sobre la solución de la ecuación con dos medios de características diferentes.

4.8. Comparativo

Finalmente, para validar los resultados y tener algo con que compararlos, se simuló la ecuación de Richards en los medios 1 y 1B usando Flexpde 6 (PDE solutions Inc) ©. Este software usa elementos finitos triangulares y mallas adaptivas para minimizar el error en las regiones de alto gradiente.

Una comparación visual entre los resultados muestra una gran similitud entre ambas apro-

ximaciones. En la figura 4-47 se observa la simulación para el medio 1 el cual corresponde a la ecuación de Van Genuchten. En negro se observa la simulación realizada con el algoritmo propuesto, mientras que en gris se observa la simulación de Flexpde.

Por su parte, la figura 4-48 muestra los resultados obtenidos mediante la simulación para el medio 1B, donde de igual manera se observa gran similitud con los resultados obtenidos por medio del algoritmo propuesto.

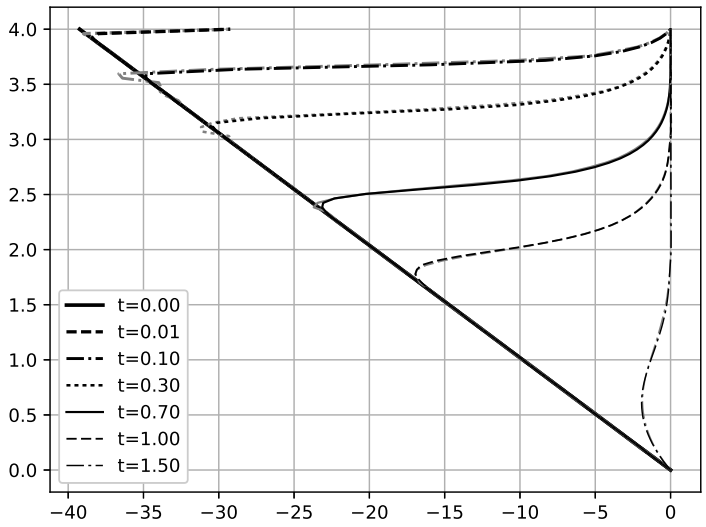


Figura 4-47: Comparativo entre la solución encontrada por flexPde y el algoritmo propuesto para la ecuación del medio 1 (Van Genuchten-Mualem). En negro la solución del algoritmo propuesto con la malla de 161x161 puntos; En gris la solución encontrada por flexPde usando mallas adaptivas

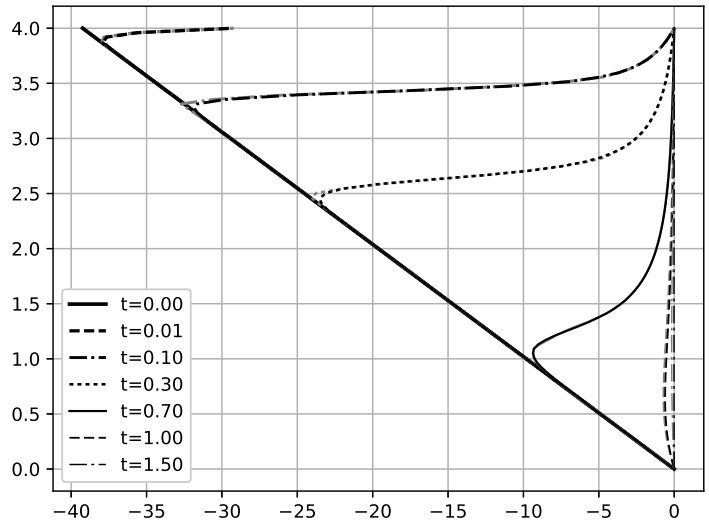


Figura 4-48: Comparativo entre la solución encontrada por flexPde y el algoritmo propuesto para la ecuación del medio 1b (Fredlund y Xing). En negro la solución del algoritmo propuesto con la malla de 161x161 puntos; En gris la solución encontrada por flexPde usando mallas adaptivas

Capítulo 5

Conclusiones

En este trabajo se presenta una implementación del algoritmo de Crank-Nicolson el cual ha sido modificado para aproximar soluciones a ecuaciones diferenciales no lineales. El método propuesto en este trabajo, implementa controles adaptivos para modificar el paso en el tiempo favoreciendo la convergencia del mismo.

El algoritmo propuesto implementa un control de error iterativo para mejorar el resultado de las aproximaciones. De igual manera, se implementa un método para el cálculo de los operadores de forma optimizada reduciendo el número de evaluaciones de la submatriz, la cual permite el cálculo de los coeficientes de peso que se asignan a cada nodo.

Por medio del método propuesto en este trabajo, se resuelve la ecuación de Richards en un terraplén carretero sujeto a infiltración de humedad en el talud y un manto freático en la base. Se realizan varias pruebas para observar y comparar el comportamiento de la solución.

5.1. Medio uniforme

Comparando los resultados observados en la sección 4.4, se observa el comportamiento del método de acuerdo a lo esperado y que se confirma en la figura 4-47 al coincidir con los resultados obtenidos mediante flexPde. Cabe mencionar que a diferencia de este último, el algoritmo propuesto en este trabajo evita la presencia de oscilaciones espurias, las cuales

se observan en la figura 4-47 en los picos de presión negativa por debajo de las condiciones iniciales.

Por otro lado, también podemos observar en la figura 4-24 la tendencia que tiene el error a lo largo del tiempo, las regiones donde la calidad de la malla resulta óptima muestran una tendencia de disminuir el error a la cuarta parte cuando el tiempo se refina a la mitad, sin embargo, en las regiones cercanas al talud, se observa un error que no disminuye en la misma proporción, principalmente debido a la calidad de las mallas que contienen celdas muy elongadas en esta región.

En la figura 4-26 se puede observar un comportamiento similar a lo largo del tiempo, donde, en las regiones de menor calidad el error decrece lentamente al refinar el tiempo, mientras que en las zonas de mayor calidad el error tiende a decrecer a la cuarta parte.

Así mismo, al pasar a la sección 4.5, las figuras correspondientes muestran un comportamiento acorde a lo esperado para un fenómeno de infiltración y que se comprueba al compararlo con los resultados obtenidos mediante flexPde como se observa en la figura 4-48. Bajo las condiciones de la ecuación correspondiente, flexPde muestra menores oscilaciones espurias al aproximar la solución con la ecuación de Fredlund y Xing, teniendo diferencias pequeñas con los resultados del algoritmo en las zonas de menor presión sin afectar el resultado general de la dinámica.

Para un adecuado análisis de los fenómenos que se dan en esta región se requeriría de un estudio experimental para evaluar cual es el comportamiento más preciso de la solución.

Las figuras 4-36 y 4-38 confirman un comportamiento similar en las regiones de mala calidad donde el error mantiene una proporción similar al refinar temporalmente mientras que en las regiones de mejor calidad este error disminuye por un cuarto.

En conjunto, estas pruebas nos muestran un método de aproximación a la solución capaz de llegar a la misma a pesar de la mala calidad de las mallas y la no linealidad del problema. Los problemas de aproximación en estas inadecuadas regiones se pueden disminuir por medio de mallas no estructuradas diseñadas adecuadamente para cada terraplén.

5.2. Medios con interfaz

En la figura 4-39 podemos observar comportamientos similares de las dos soluciones, las cuales difieren principalmente en la región de interfaz, la cual en consecuencia nos induce a emplear mallas ligeramente diferentes como se ha indicado en las secciones correspondientes. Estos cambios en la malla producen alteraciones en las interpolaciones las cuales se ven reflejadas en la dinámica de la solución que se observa en la figura previamente mencionada, por medio de atrasos y adelantos en distintas regiones de la solución como resultado de la interpolación.

A una altura de dos metros, se puede observar el comportamiento de la solución al emplear condiciones de flujo sobre la interfaz en comparación con la solución para un sólo medio, en esta región y nodos vecinos se observa un ligero atraso en el comportamiento de la solución, el cual se debe a la resolución de las mallas.

En la figura 4-40 se observa el comportamiento descrito del error debido a la interpolación, mostrando un error relativamente alto en las regiones donde las mallas difieren más entre si, y un error más bajo en la región dos donde las mallas se parecen más entre si.

5.3. Conclusiones

El algoritmo presentado en esta tesis muestra resultados satisfactorios para aproximar la solución a la ecuación de Richards los cuales son comparables con los resultados obtenidos por medio de software comercial. Los resultados fueron comparados con FlexPde y se observan coincidencias bastante buenas, sin embargo poder validar los resultados obtenidos por medio de ambos métodos requiere de experimentación.

El algoritmo propuesto para controlar el error presenta buenos resultados para favorecer la convergencia del método. El número de iteraciones finales que se realizan son del mismo orden a las iteraciones requeridas por FlexPde para llegar al resultado.

Como se puede observar el algoritmo nos da un buen resultado para aproximar la solución a la ecuación de Richards, manejando la no linealidad del problema de manera adecuada siendo un algoritmo de segundo orden en el tiempo. El algoritmo muestra un buen comportamiento

para manejar las condiciones de frontera cambiantes debido al algoritmo de manejo de errores.

5.4. Por hacer

Como se ha observado en este trabajo, el algoritmo propuesto funciona para modelar la ecuación de Richards y en general para modelar ecuaciones diferenciales parciales, sin embargo, como todo trabajo en desarrollo hay cosas que a futuro se podrían mejorar tomando como base este algoritmo.

Desde el punto de vista matemático, se podría probar que el algoritmo iterativo para corregir la aproximación es una contracción. Si se demuestra esta afirmación, se pueden sentar las bases para determinar en que intervalo es una contracción y optimizar de manera adecuada el proceso de convergencia.

El método para actualizar cada nueva aproximación se ha elegido de manera empírica para buscar la convergencia del método, sin embargo distintas estrategias se pueden seguir a futuro para optimizar el algoritmo. Algunas estrategias posibles a estudiar consisten en analizar el error para elegir la forma en que se actualizará el nuevo valor de la aproximación a la solución. Posiblemente las ideas mas fuertes para optimizar esta actualización de la aproximación surjan una vez que se haya demostrado que este algoritmo es una contracción.

Otro punto que se puede estudiar es la elección de paso del tiempo, en este trabajo se ha optado por tomar en cuenta el número de ejecuciones para elegir cuando se refinará el tiempo, sin embargo existen mas caminos que se pueden seguir en la optimización del método. Entre otras opciones se puede tomar en cuenta un análisis del error para elegir tamaños de paso óptimos para el algoritmo.

Desde el punto de vista físico del problema de infiltración hay dos áreas que se pueden estudiar, una de ellas consiste en verificar experimentalmente los resultados. Por otro lado, una parte de gran interés para la ciencia consiste en estudiar tanto físicamente como numéricamente el comportamiento de la infiltración entre dos medios de características diferentes.

Apéndice A

Códigos

A.1. Referentes a las matrices

A.1.1. Cálculo de gammas

Los siguientes códigos se encargan del cálculo de los pesos para establecer las matrices que aproximen los operadores necesarios.

apendices/gammat.jl

```
include("gammma.jl")

function gammat(x,y, A,B,C,D,E,F, m,n, Mr=[1,Inf],Nr=[1,Inf])
    #%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    # Esta funcion está pensada en calcular las matrices correspondientes al
    # operador A d2/dx2 + B d2/dxdy + C d2/dy 2 + Dd/dx + E d/dy + F
    # en una matriz lógicamente rectangular cuyas coordenadas estan dadas
    # en las variables x, y
    #
    # m, n son vectores me dan las coordenadas minimas y máximas de cada nodo
    # a calcular sobre los ejes x, y respectivamente, cada variable es un
    # vector de la forma [min,max]
    #
    # Mr, Nr son vectores opcionales para delimitar las fronteras de la región,
    # esto será empleado para poder calcular correctamente las derivadas no
    # centradas en las regiones de interfaz
```

```
    #%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    (M,N) = size(x); # Obtengo mis máximos globales
    K = spzeros(length(x),length(x)); # Preparo mi matriz operador

    # Verifico mis regiones
    Mr = [Int( max( Mr[1], 1)),Int( min( Mr[2], M))]
    Nr = [Int( max( Nr[1], 1)),Int( min( Nr[2], N))]

    for i = m[1]:m[2]
        for j = n[1]:n[2]
            # Calculo el centro de mi submalla.
            i0 = min(Mr[2]−1,max( Mr[1]+1,i))
            j0 = min(Nr[2]−1,max( Nr[1]+1,j))

            # Recupero las coordenadas de mi submalla.
            u = x[(i0−1):(i0+1),(j0−1):(j0+1)]
            v = y[(i0−1):(i0+1),(j0−1):(j0+1)]
            # Recupero la posición de mi nodo p0 sobre la submalla.
            p0 = 5+(i−i0)+3∗(j−j0)
            # Calculo el vector de gammas.
            k = gammma(u,v, p0, A,B,C,D,E,F)

            # Recupero renglon y columna sobre mi matriz del operador.
            p = (j−1)∗M+i
            p0 = (j0−1)∗M+i0

            # Armado de la matriz del operador.
            K[p,((p0−1):(p0+1))−M] = [k[1] k[2] k[3]];
            K[p,(p0−1):(p0+1)] = [k[4] k[5] k[6]];
            K[p,((p0−1):(p0+1))+M] = [k[7] k[8] k[9]];
        end
    end
    return K
end
```

apendices/gammma.jl

```
function gammma(x,y,p0,A,B,C,D,E,F)
    #%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    # Calcula el vector de gammas para el esquema de
    # diferencias finitas que aproxima:
    # Au_xx+Bu_xy+Cu_yy+Du_x+Eu_y+F
    # en el punto P_0
    #%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    N=length(x);
    I=[1:(p0-1);(p0+1):N];
    dx=x[I]-x[p0];
    dy=y[I]-y[p0];
    M=[dx dy dx.*dx dx.*dy dy.*dy]';
    Gamma=pinv(M)*[D E 2*A B 2*C]';
    Gamma=[F-sum(Gamma);Gamma];
    Gamma = [Gamma[2:p0];Gamma[1];Gamma[(p0+1):N]];
end
```

A.1.2. Cálculo de las mallas

Los códigos presentados a continuación son los responsables de llenar mis estructuras de datos para establecer la malla de mi problema de estudio.

apendices/malla.jl

```
using PyPlot
include("refina.jl");
isdefined(:p) || (p = q = 41)

## Valores iniciales
base = 9;
h1 = 2;
h2 = 5;
talud = 3;
pavimento = 2; #Region impermeable de la frontera superior (norte)
refV = 3*[1];
refS = 2*[1 1];
refI = 1*[1 1];

isdefined(:miMalla) || (miMalla = 2)
if miMalla < 2
    refI = 0*refI;
```

```
end
if miMalla < 1
    refV = 0*refV;
    refS = 0*refS;
end

pm = round((p-sum(refS)-2*sum(refI))*h1/h2);
pm = trunc(Int,pm);
ph1 = pm + sum(refI);

## Creo una malla inicial en dos partes
xi = repmat(linspace(0,base,p-sum(refV)), 1,pm)
yi = repmat(linspace(0,h1,pm), 1,p-sum(refV))'
xs = repmat(linspace(0,base,p-sum(refV)), 1,p-sum(refS)-2*sum(refI)-pm+1)
ys = repmat(linspace(h1,h2,p-sum(refS)-2*sum(refI)-pm+1), 1,p-sum(refV))'

#frontera sup
(xs,ys)=refinaSup(xs,ys,refS);

# Agrego refinamientos en fronteras internas
(xi,yi) = refinaSup(xi,yi,refI);
(xs,ys) = refinaInf(xs,ys,refI);

# Agrego refinamientos en frontera derecha
(xi,yi) = refinaDer(xi,yi, refV);
(xs,ys) = refinaDer(xs,ys, refV);

# Corrijo las fronteras (Ajusto al talud)
xs[:,end] = xs[:,end]*talud/base;
xs[end,:] = base-(ys[end,:]-h1)*(base-talud)/(h2-h1);

# Corrijo los interiores (Reparto los nodos al interior)
for i = 2:size(xs,2)-1
    xs[2:end-1,i] = xs[2:end-1,i]*xs[end,i]/xs[end,1]; end

# Pego las mallas
x=[xi xs[:,2:end]];
y=[yi ys[:,2:end]];

cla()
```

```
plot(x,y);
plot(x',y');
```

apendices/refina.jl

```
function refinaSup(x,y,n)
    x=x';
    y=y';
    for i = n
        xp = x[1:(end−i),:];
        yp = y[1:(end−i),:];
        for j = i:−1:1
            xp = [xp; [0.5 0.5;0 1]∗x[end−j:end−j+1,:]];
            yp = [yp; [0.5 0.5;0 1]∗y[end−j:end−j+1,:]];
        end
        x = xp;
        y = yp;
    end
    return (x',y');
end

function refinaInf(x,y,n)
    x=x';
    y=y';
    for i = n
        xp = x[(i+1):end,:];
        yp = y[(i+1):end,:];
        for j = i:−1:1
            xp = [[1 0;0.5 0.5]∗x[j:j+1,:]; xp];
            yp = [[1 0;0.5 0.5]∗y[j:j+1,:]; yp];
        end
        x = xp;
        y = yp;
    end
    return (x',y');
end

function refinaDer(x,y,n)
    x=x';
    y=y';
```

```
    for i = n
        xp = x[:,1:(end−i)];
        yp = y[:,1:(end−i)];
        for j = i:−1:1
            xp = [xp x[:,end−j:end−j+1]∗[0.5 0.5;0 1]'];
            yp = [yp y[:,end−j:end−j+1]∗[0.5 0.5;0 1]'];
        end
        x = xp;
        y = yp;
    end
    return (x',y');
end
```

A.2. Referentes al método

A.2.1. Medio único

Este código es el encargado de aproximar la solución empleando un solo medio en toda la región del terraplén.

apendices/CN3.jl

```
##############################################################
# Esta versión del codigo implementa la primer version del Crank−Nicolson para
# resolver la ecuación de Richards empleando mis modificaciones para cosas no
# lineales. En esta versión ya se está implementando el paso adaptivo, el cual
# se podría mejorar en alguna versión posterior.

if (isdefined(:simMedio) && simMedio == 2)
   simMed = "medio1B.jl"
   simMedio = 1
else
   simMed = "medio1.jl"
end

include("gammat.jl") # Cálculo de la matriz del operador
include(simMed) # Constantes y funciones para el medio1
                        # funciones K y mw
funk(u) = funk1(u);
```

```
funmw(u) = funmw1(u);

##########################################################################
# Este bloque se dedica a cargar las condiciones de mi malla.
println("generando_malla");
include("malla.jl") # malla.jl carga todo lo necesario de la malla.

##########################################################################
# Este bloque me va a generar las condiciones necesarias para generar los
# operadores que modelan mi ecuación.
println("generando_los_operadores");

# Operadores al interior de la malla
L2 = gammat(x,y, 1,0,1,0,0,0, [2 p−1],[2 q−1])
Lx = gammat(x,y, 0,0,0,1,0,0, [2 p−1],[2 q−1])
Ly = gammat(x,y, 0,0,0,0,1,0, [2 p−1],[2 q−1])

# Operador identidad para el interior y frontera respectivamente
I = gammat(x,y, 0,0,0,0,0,1, [2 p−1],[2 q−1])
Ie = speye(size(I,1)) − I

# Operador en fronteras Dirichlet
p0 = div(p,3)+1 # Punto donde inicia el hombro de la carretera
q0 = ph1 # Punto mas bajo del talud
#q0 = div(q,3)

fd = gammat(x,y, 0,0,0,0,0,1, [1 p],[1 1]) # Base del terraplen
fd = fd + gammat(x,y, 0,0,0,0,0,1, [p0 p],[q q]) # Talud
fd = fd + gammat(x,y, 0,0,0,0,0,1, [p p],[q0 q]) # Hombro

#Operador en fronteras Neumann
fn = gammat(x,y, 0,0,0,−1, 0,0, [1 1],[2 q−1]) # Centro del terraplen
fn = fn + gammat(x,y, 0,0,0, 1, 0,0, [p p],[2 q0−1]) # Frontera derecha.
fn = fn + gammat(x,y, 0,0,0, 0, 1,0, [1 p0−1],[q q]) # Carpeta

##########################################################################
# Este bloque carga las condiciones iniciales y de frontera.
println("condiciones_iniciales_y_de_frontera");

#Algunas constantes y definiciones
```

```
isdefined(:Tmax) || (Tmax=0.5)
g = 9.81 # Peso específico del agua !!!Revisar nombre
v = 0.1 # Velocidad, ya no es necesario !!!Revisar
c = 0.5 # Coeficiente cfl, no necesario !!!Revisar

fK(u) = ones(size(u))*v # A borrar en el futuro
fm(u) = ones(size(u))*1 # A borrar en el futuro

fK(u) = funk(u) # Conductividad
fm(u) = funmw(u) # Modulo de almacenamiento

dx= dy= 1/(p-1)
dt= c*dx*dy/v # Intento previo de elegir el paso de tiempo
dt=0.01; # si no se implementa se puede borrar pronto.

#Condiciones iniciales
z = -g*y # Presion de poro inicial
z = reshape(z, length(x),1)

# Calcúlo mis operadores
K = fK(z)
mw= (min(fm(z),1e-3)*g).*I;
L = K.*L2+(Lx*K).*Lx+(Ly*K).*Ly;

# right hand side: contiene las cosas no
# linealizables y condiciones de frontera.
rhs=sparse(fd*z)+g*Ly*K

# Impresion inicial de datos
#fig = figure(figsize=(9,4))
include("muestra.jl")
muestra(x,y,z,0.0);

##########################################################
# Ciclo principal de cálculo: Inicialización
println("Iniciando_ciclo_de_computo");
zg = zf = z; # Valores de presión de poro,
Lp = L; # Operadores de cálculo
Kp = K; # Conductividad
mp = mw; # Módulo de almacenamiento
```

```
Dt = 0.01; # Resolución para guardar datos
t = 0; # Mi tiempo inicial

sumCnta = 0; # Suma contador A
matCnta = [0]; # Resultados parciales contador A

Dtmax = Dt; # Paso de tiempo maximo
isdefined(:DeltaT) && ( Dtmax = DeltaT)
c0=Dt # paso en el tiempo

Zs=[z]
Ts=[0]

##########################################################################
# Ciclo principal de cálculo
for tf = Dt:Dt:Tmax
    cnta=0;
    tic()
    while t < tf
        # Calculo el paso en el tiempo
        #dt = c*dx*dy/maximum(I*(K./(max(fm(z),1e-3)*g)))
        c0 = min(tf-t,2*c0,Dtmax); # Prueba

        cnt = 20; # Contador de iteraciones
        errp = 0; # Error previo, inicialización

        while true
            # Actualizo mis contadores.
            cnta += 1;

            # Actualizo los operadores.
            K = fK(zg)
            #mw= fm(zg)*g
            mw= (max(0.5*(fm(z)+fm(zg)),1e-3)*g).*I;
            L = K.*L2+(Lx*K).*Lx+(Ly*K).*Ly;

            #rhs = min(Ie*z,.1)
            ###rhs = fd*min(z+100*c0,0)+g*Ly*(K+Kp)/2
            #rhs = fd*min(z+1000*c0,0)+c0*g*Ly*(K+Kp)/2
            rhs = min(1000*c0,-fd*z)+c0*g*Ly*(K+Kp)/2 #tmp
```

```
    # Algoritmo base
    ###zf = (mw.*I+fn+fd-c0*L/2)\((mp.*I+c0*Lp/2)*z+rhs);
    zf = (mw-fn/2+fd-c0*L/2)\((mw+fn/2+fd+c0*Lp/2)*z+rhs);

    # Reviso la aproximacion
    # err = maxabs(zg-zf);
    # err =maxabs(zg-zf)-maxabs(zg)*2e-3;
    err = norm(zg-zf);
    if err < 2e-3 + norm(zg)*2e-3
        break;
    end

    # Si aún no converge reviso mis criterios.
    if err > errp
        cnt = cnt-10;
    else
        cnt = cnt-1;
    end

    if cnt > 0
        #println(err)
        #zg = zg+(zf-zg)*0.8;
        zg = zg+(zf-zg)*0.7071;
        errp=err;
        continue;
    end

    println("Refinando:")
    zg = (zg+z)/2;
    c0 = c0/2;
    cnt = 20;
    errp= err
end

# Resultados validos
Lp = L;
Kp = K;
z = zg = zf;
#zg = z +(zf-z)*2 # Prueba
```

```
        #z = zf # Prueba
        t = t+c0;
    end

    #Impresion de datos
    println(cnta)
    sumCnta=sumCnta+cnta;
    matCnta=[matCnta;sumCnta]
    toc()

    Zs=[Zs z]
    Ts=[Ts t]
    muestra(x,y,z,t);
end

println(sumCnta)
```

A.2.2. Dos medios

Este código presenta modificaciones para poder aproximar la solución al problema en dos medios de características diferentes.

apendices/CNvm.jl

```
##########################################################
# Esta version del codigo esta modificada para empezar a trabajar con dos medios
# y la ecuacion de Richards
# -----Modificaciones en proceso

include("gammat.jl") # Cálculo de la matriz del operador
include("medio1.jl") # Constantes y funciones para el medio1
                     # funciones K y mw
fK1(u) = funk1(u) # Conductividad
fm1(u) = funmw1(u) # Modulo de almacenamiento

if( isdefined(:variosMedios) && !variosMedios)
  fK2(u) = funk1(u) # Conductividad
  fm2(u) = funmw1(u) # Modulo de almacenamiento
else
```

```
  include("medio2.jl") # Constantes y funciones para el medio2
                       # funciones K y mw
  fK2(u) = funk2(u) # Conductividad
  fm2(u) = funmw2(u) # Modulo de almacenamiento
end

##########################################################
# Este bloque se dedica a cargar las condiciones de mi malla.
include("malla.jl") # malla.jl carga todo lo necesario de la malla.



##########################################################
# Este bloque me va a generar las condiciones necesarias para generar los
# operadores que modelan mi ecuación.

# Operadores al interior de la malla
L2 = gammat(x,y, 1,0,1,0,0,0, [2 p−1],[2 q−1])
Lx = gammat(x,y, 0,0,0,1,0,0, [2 p−1],[2 q−1])
Ly = gammat(x,y, 0,0,0,0,1,0, [2 p−1],[2 q−1])
# Adiciones para manejar varias regiones
L2_R1 = gammat(x,y, 1,0,1,0,0,0, [2 p−1],[ph1 q−1], [1 p],[ph1 q])
Lx_R1 = gammat(x,y, 0,0,0,1,0,0, [2 p−1],[ph1 q−1], [1 p],[ph1 q])
Ly_R1 = gammat(x,y, 0,0,0,0,1,0, [2 p−1],[ph1 q−1], [1 p],[ph1 q])

L2_R2 = gammat(x,y, 1,0,1,0,0,0, [2 p−1],[2 ph1], [1 p],[1 ph1])
Lx_R2 = gammat(x,y, 0,0,0,1,0,0, [2 p−1],[2 ph1], [1 p],[1 ph1])
Ly_R2 = gammat(x,y, 0,0,0,0,1,0, [2 p−1],[2 ph1], [1 p],[1 ph1])

# Operador identidad para el interior y frontera respectivamente
I = gammat(x,y, 0,0,0,0,0,1, [2 p−1],[2 q−1])
Ie = speye(size(I,1)) − I
# Adiciones para manejar varias regiones
I_R1 = gammat(x,y, 0,0,0,0,0,1, [2 p−1], [ph1+1 q−1], [1 p],[1 q])
I_in = gammat(x,y, 0,0,0,0,0,1, [2 p−1], [ph1 ph1], [1 p],[1 q])
I_R2 = gammat(x,y, 0,0,0,0,0,1, [2 p−1], [2 ph1−1], [1 p],[1 q])

# Operador en fronteras Dirichlet
p0 = div(p,3)+1 # Punto donde inicia el hombro de la carretera
q0 = ph1 # Punto mas bajo del talud
```

```
fd = gammat(x,y, 0,0,0,0,0,1, [1 p],[1 1]) # Base del terraplen
fd = fd + gammat(x,y, 0,0,0,0,0,1, [p0 p],[q q]) # Talud
fd = fd + gammat(x,y, 0,0,0,0,0,1, [p p],[q0 q]) # Hombro

#Operador en fronteras Neumann
fn = gammat(x,y, 0,0,0,−1, 0,0, [1 1],[2 q−1]) # Centro del terraplen
fn = fn + gammat(x,y, 0,0,0, 1, 0,0, [p p],[2 q0−1]) # Frontera derecha.
fn = fn + gammat(x,y, 0,0,0, 0, 1,0, [1 p0−1],[q q]) # Carpeta

##############################################################
# Este bloque carga las condiciones iniciales y de frontera.

#Algunas constantes y definiciones
isdefined(:Tmax) || (Tmax=0.5)
g = 9.81 # Peso específico del agua !!!Revisar nombre
v = 0.1 # Velocidad, ya no es necesario !!!Revisar
c = 0.5 # Coeficiente cfl, no necesario !!!Revisar

dx= dy= 1/(p−1)
dt= c∗dx∗dy/v # Intento previo de elegir el paso de tiempo
dt=0.01; # si no se implementa se puede borrar pronto.

#Condiciones iniciales
z = −g∗y # Presion de poro inicial
z = reshape(z, length(x),1)

# Calcúlo mis operadores
# Calculos por region
K1 = fK1(z)
mw1 = (max(fm1(z),1e−3)∗g).∗I;
L_R1 = I_R1∗(K1.∗L2_R1+(Lx_R1∗K1).∗Lx_R1+(Ly_R1∗K1).∗Ly_R1);
K2 = fK2(z)
mw2 = (max(fm2(z),1e−3)∗g).∗I;
L_R2 = I_R2∗(K2.∗L2_R2+(Lx_R2∗K2).∗Lx_R2+(Ly_R2∗K2).∗Ly_R2);
L_in = I_in∗(K1.∗Lx_R1−K2.∗Lx_R2)

L = L_R1+L_R2
mw = I_R1∗mw1+I_R2∗mw2

# right hand side: contiene las cosas no
```

```
# linealizables y condiciones de frontera.
gKy = g*I_R1*Ly_R1*K1+g*I_R2*Ly_R2*K2
rhs = sparse(fd*z)+gKy

# Impresion inicial de datos
include("muestra.jl")
muestra(x,y,z,0.0);

##########################################################
# Ciclo principal de cálculo: Inicialización
zg = zf = z; # Valores de presión de poro,
Lp = L; # Operadores de cálculo
L_ip = L_in;
gKyp = gKy;
mp = mw; # Módulo de almacenamiento
Dt = 0.01; # Resolución para guardar datos
t = 0; # Mi tiempo inicial

sumCnta = 0; # Suma contador A
matCnta = [0]; # Resultados parciales contador A

Dtmax = Dt; # Paso de tiempo maximo
isdefined(:DeltaT) && ( Dtmax = DeltaT)
c0=Dt # paso en el tiempo

Zs=[z]
Ts=[0]

##########################################################
# Ciclo principal de cálculo
for tf = Dt:Dt:Tmax
    cnta=0;
    tic()
    while t < tf
        # Calculo el paso en el tiempo
        c0 = min(tf-t,2*c0,Dtmax); # Prueba



cnt = 20; # Contador de iteraciones
        errp = 0; # Error previo, inicialización
```

```
while true
    # Actualizo mis contadores.
    cnta += 1;

    # Actualizo los operadores.
    # Calculos por region
    K1 = fK1(zg)
    mw1 = (max(0.5*(fm1(z)+fm1(zg)),1e-3)*g).*I;
    L_R1 = I_R1*(K1.*L2_R1+(Lx_R1*K1).*Lx_R1+(Ly_R1*K1).*Ly_R1);
    K2 = fK2(zg)
    mw2 = (max(0.5*(fm2(z)+fm2(zg)),1e-3)*g).*I;
    L_R2 = I_R2*(K2.*L2_R2+(Lx_R2*K2).*Lx_R2+(Ly_R2*K2).*Ly_R2);
    L_in = I_in*(K1.*Ly_R1-K2.*Ly_R2)

    L = L_R1+L_R2
    mw = I_R1*mw1+I_R2*mw2

    gKy = g*(I_R1*Ly_R1*K1+I_R2*Ly_R2*K2);
    rhs = fd*min(z+1000*c0,0)+c0*(gKy+gKyp)/2;

    # Algoritmo base
    zf = (mw+fn/2+fd+L_in/2-c0*L/2)\((-L_ip/2-fn/2+mw+c0*Lp/2)*z+rhs);

    # Reviso la aproximacion
    err = norm(zg-zf);
    if err < 2e-3 + norm(zg)*2e-3
        break;
    end

    # Si aún no converge reviso mis criterios.
    if err > errp
        cnt = cnt-10;
    else
        cnt = cnt-1;
    end

    if cnt > 0
        #println(err)
        #zg = zg+(zf-zg)*0.8;
```

```
            zg = zg+(zf−zg)∗0.7071;
            errp=err;
            continue;
        end

        println("Refinando:")
        zg = (zg+z)/2;
        c0 = c0/2;
        cnt = 20;
        errp= err
    end

    # Resultados validos
    Lp = L;
    L_ip = L_in;
    gKyp = gKy;
    z = zg = zf;
    t = t+c0;
end

#Impresion de datos
println(cnta)
sumCnta=sumCnta+cnta;
matCnta=[matCnta;sumCnta]
toc()

Zs=[Zs z]
Ts=[Ts t]
muestra(x,y,z,t);
end

println(sumCnta)
toVideo();
```

A.2.3. Impresión de datos

apendices/muestra.jl

```
using PyPlot
```

```
close()
fig = figure(figsize=(9,4))
isdefined(:video) || (video = false);

videoCnt = 0;
path = tempdir()*"/video";
if !isdir(path)
    mkdir(path)
end
#run(`rm $path/img-*.jpg`)

function muestra(x,y,z,tf)
    clf();
    title("t="*rpad(tf,5,0));
    contour(x,y,reshape(z,size(x)),10);
    colorbar();

    if video
        savefig("/tmp/video/img-"*lpad(videoCnt,5,0)*".jpg");
        global videoCnt += 1;
    end

    pause(0.01);
end

function toVideo()
    if !video
        return;
    end

    try
        println("Grabando movie.gif")
        run(`convert -delay 30 $path/img*jpg movie.gif`)
    catch
        println("No se pudo generar movie.gif")
        println("Requiere convert de imagemagick")
        return;
    end
    try
        println("Grabando movie.mpg")
```

```
        run(‘convert movie.gif movie.mpg‘)
    catch
        println(”No_se_pudo_generar_movie.mpg”)
        println(”Requiere_convert_de_imagemagick”)
        return;
    end
    try
        println(”Gravando_movie.avi”)
        run(‘ffmpeg −i movie.gif movie.avi‘)
    catch
        println(”No_se_pudo_generar_movie.avi”)
        println(”Requiere_ffmpeg_instalado”)
    end
    try
        println(”Gravando_movie.mp4”)
        run(‘ffmpeg −i movie.gif movie.mp4‘)
    catch
        println(”No_se_pudo_generar_movie.mp4”)
        println(”Requiere_ffmpeg_instalado”)
    end

    # Limpieza final
    println(”Borrando_temporales”)
    for (root, dirs, files) in walkdir(path)
        for file in files
            if ismatch(r”^img−[0−9]∗\.jpg$”, file)
                rm(root∗”/”∗file)
            end
        end
    end
    println(”Finalizado”)
end
```

A.3. Referentes al medio

A.3.1. Medio1

apendices/medio1.jl

```
function funk1( s)
    #FK Summary of this function goes here
    # Detailed explanation goes here

    #Algunas definiciones de cosntantes
    ## Regi\'on R1
    R1theta_r = 0.04;
    R1theta_s = 0.37;
    R1alfa = 0.089;
    R1n = 1.57;
    R1ks = 0.25;

    R1m = 1-1/R1n;
    R1lambda = R1m;

    ##
    K = zeros(size(s));
    u = max(-s,0);

    theta = ft1(R1alfa,u,R1n,R1m);
    K = R1ks*sqrt(theta).*(1-(1-theta.^(1/R1lambda)).^R1lambda).^2;

end

function ft1(alfa,s,n,m)
    t = 1./((alfa*s).^n+1).^m;
end

function funmw1( s)
    #FMW Summary of this function goes here
    # Detailed explanation goes here

    ## Algunas constantes
    R1theta_r = 0.04;
    R1theta_s = 0.37;
    R1alfa = 0.089;
    R1n = 1.57;
    R1m = 1-1/R1n;
```

```
    ##
    mw = zeros(size(s));
    u = max(−s,0);

    mw = (R1alfa∗(R1alfa∗u).^(R1n−1))./(((R1alfa∗u).^R1n+1).^(R1m+1));
    mw = (R1theta_s−R1theta_r)∗mw;
    mw = (R1m∗R1n)∗mw; # Parece que este termino estaba ausente

end

function funth1( s)
    #FMW Summary of this function goes here
    #Detailed explanation goes here

    ## Algunas constantes
    R1theta_r = 0.04;
    R1theta_s = 0.37;
    R1alfa = 0.089;
    R1n = 1.57;
    R1ks = 0.25;

    R1m = 1−1/R1n;
    R1lambda = R1m;

    ##
    th = zeros(size(s));
    s = max(−s,0);



    th = ft1(R1alfa,s,R1n,R1m);
end
```

A.3.2. Medio1B

apendices/medio1B.jl

```
function funk1( s)
    #FK Summary of this function goes here
    # Detailed explanation goes here
```

```
    #Algunas definiciones de cosntantes
    ## Regi\'on R1
    theta_r = 0.0001;
    theta_s = 0.4;
    n = 2.544;
    af = 5;
    nf = 2;
    mf = 1;
    ks = 0.864;

    R1m = 1-1/n;

    ##
    K = zeros(size(s));
    u = max(-s,0);

    #theta = power(log(exp(1)+power(u(s)/af,nf)),-mf);
    theta = 1./log(exp(1)+(u/af).^nf).^mf;
    K = ks*sqrt(theta).*(1-(1-theta.^(1/R1m)).^R1m).^2;
end

function ft1(alfa,s,n,m)
    t = 1./((alfa*s).^n+1).^m;
end

function funmw1( s)
    #FMW Summary of this function goes here
    # Detailed explanation goes here

    ## Algunas constantes
    theta_r = 0.0001;
    theta_s = 0.4;
    n = 2.544;
    af = 5;
    nf = 2;
    mf = 1;
    ks = 0.864;

    R1m = 1-1/n;
```

```
    R1lambda = R1m;


    ##
    mw = zeros(size(s));
    u = max(-s,0);

    #mw(vaux) = nf*mf*power(s/af,nf-1).*power(log(exp(1)+power(s/af,nf)),-mf-1);
    mw = nf*mf*(u/af).^(nf-1).*(log(exp(1)+(u/af).^(nf))).^(-mf-1);
    #mw(vaux) = mw(vaux)./(af*(exp(1)+power(s/af,nf)));
    mw = mw./(af*(exp(1)+(u/af).^(nf)));
    #mw(vaux) = (theta_s-theta_r)*mw(vaux);
    mw = (theta_s-theta_r)*mw;

end
```

A.3.3. Medio2

apendices/medio2.jl

```
function funk2( s)
    #FK Summary of this function goes here
    # Detailed explanation goes here


    #Algunas definiciones de cosntantes
    ## Regi\'on R1
    R1theta_r = 0.054;
    R1theta_s = 0.355;
    R1alfa = 0.00102;
    R1n = 1.43;
    R1ks = 4.32e-6;


    R1m = 1-1/R1n;
    R1lambda = R1m;


    ##
    K = zeros(size(s));
    u = max(-s,0);


    theta = ft2(R1alfa,u,R1n,R1m);
```

```
    K = R1ks*sqrt(theta).*(1-(1-theta.^(1/R1lambda)).^R1lambda).^2;

end

function ft2(alfa,s,n,m)
    t = 1./((alfa*s).^n+1).^m;
end

function funmw2( s)
    #FMW Summary of this function goes here
    # Detailed explanation goes here

    ## Algunas constantes
    R1theta_r = 0.054;
    R1theta_s = 0.355;
    R1alfa = 0.00102;
    R1n = 1.43;
    R1m = 1-1/R1n;

    ##
    mw = zeros(size(s));
    u = max(-s,0);

    mw = (R1alfa*(R1alfa*u).^(R1n-1))./(((R1alfa*u).^R1n+1).^(R1m+1));
    mw = (R1theta_s-R1theta_r)*mw;
    mw = (R1m*R1n)*mw; # Parece que este termino estaba ausente

end

function funth2( s)
    #FMW Summary of this function goes here
    #Detailed explanation goes here

    ## Algunas constantes
    R1theta_r = 0.054;
    R1theta_s = 0.355;
    R1alfa = 0.00102;
    R1n = 1.43;
    R1m = 1-1/R1n;
```

```
    R1ks = 4.32e−6;
    R1lambda = R1m;

    ##
    th = zeros(size(s));
    s = max(−s,0);



    th = ft1(R1alfa,s,R1n,R1m);
end
```

Bibliografía

AASHTO. *Mechanistic-Empirical Pavement Design Guide: A Manual of Practice* (2008)

Alonso, E.E., Gens, A., y Josa, A. A constitutive model for partially saturated soils. *Géotechnique* **40**(3):405–430 (1990)

Chávez, C. y Alonso, E.E. A constitutive model for crushed granular aggregates wich includes suction effects. *Soils and foundations* **43**(4):215–227 (2003)

Crespo, C. *Vias de comunicacion / Communication Lines (Spanish Edition)*. Editorial Limusa S.A. De C.V. (2007)

Fernández-Bonder, J. *Ecuaciones Diferenciales Parciales*. IMAS - CONICET y Departamento de Matemática, FCEyN - Univer- sidad de Buenos Aires (2014)

Fredlund, D.G., Rahardjo, H., y Fredlund, M.D. *Unsaturated Soil Mechanics in Engineering Practice*. John Wiley & Sons Inc (2012)

Genuchten, M.V. A Closed-form Equation for Predicting the Hydraulic Conductivity of Unsaturated Soils. *Soil Science Society of America journal* (44):892–898 (1980)

Gordon, W.J. y Hall, C.A. Construction of curvilinear co-ordinate systems and applications to mesh generation. *International Journal for Numerical Methods in Engineering* **7**(4):461–477 (1973)

López-Garza, G. y Martínez-Ortiz, F.H. *Ecuaciones Diferenciales Parciales*. Universidad Autónoma Metropolitana (2013)

Mualem, Y. A new model for predicting the hydraulic conductivity of unsaturated porous media. *Water Resources Research* **12**(3):513–522 (1976)

Norambuena-Contreras, J., Arbat, G., Nieto, P.G., y Castro-Fresno, D. Nonlinear numerical simulation of rainwater infiltration through road embankments by FEM. *Applied Mathematics and Computation* **219**(4):1843–1852 (2012)

Richards, L.A. Capillary conduction of liquids in porous mediums. *Journal of Applied Physics* **1**(5):318–333 (1931)

Shashkov, M. *Conservative Finite-Difference Methods on General Grids (Symbolic & Numeric Computation)*. CRC Press (1995)