



UNIVERSIDAD AUTÓNOMA DEL
ESTADO DE MORELOS

Universidad Autónoma del Estado de Morelos
Instituto de Investigación en Ciencias Básicas y Aplicadas
Centro de Investigaciones Químicas

Implementar el método de Elemento Finito a la ecuación ADR.

Aplicación al estado de Morelos.

T E S I S

QUE PARA OBTENER EL TÍTULO DE

Maestro en Ciencias

Presenta:

Mónica Andrea Navarro Ramírez

Director de Tesis:

Dr. Minhuy Hô

Co-director de Tesis:

Dr. Hugo Saldarriaga Noreña

Jurado Examinador

Presidente	Dr. Thomas Buhse
Secretario	Dr. Jorge Rivera Noriega
Vocal	Dr. Roberto Bernal Jaquez
Suplente	Dr. Mario Murillo Tovar
Suplente	Dr. Hugo Albeiro Saldarriaga Noreña

Esta tesis contó con el apoyo de la beca de Maestría otorgada por CONACYT con el número de becario **622743** (febrero 2017 - enero 2019).

RESUMEN

En las últimas décadas la contaminación del aire ha sido objeto de investigación a nivel mundial, debido a los daños que provoca, principalmente, a la humanidad y al ambiente. Es por esto que los modelos computacionales son cada vez más indispensables debido de su potencia y flexibilidad para monitorear y predecir el movimiento y transformación de la contaminación. Hasta ahora, hay pocos estudios sobre estos aspectos de contaminación en el estado [1]. Actualmente para obtener datos en Morelos se utiliza un programa llamado HYSPLIT [2] que accesa a datos de una sola estación meteorológica, la del aeropuerto, aproximadamente a 20 km del centro de la ciudad de Cuernavaca. En este proyecto, se trabaja con la ecuación de Advección-Difusión-Reacción (ADR)

$$U_t = -vU_x + \alpha U_{xx} + F$$

para modelar los fenómenos de transporte, especialmente para la transferencia de calor y energía [3].

El proyecto consistió, primero, en la derivación del algoritmo matemático del método de elemento finito (FEM) en una dimensión, empezando con la forma estacionaria, $U_t = 0$, y posteriormente la forma no estacionaria. Para cualquier problema, el uso de FEM consiste en cuatro conceptos básicos [4]: la forma fuerte, forma débil, aproximación de elemento finito y el sistema de ecuaciones lineales. Se utilizó la aproximación de Galerkin/Ritz y el método de Runge-Kutta 4 (RK4) para la resolución del sistema de ecuaciones lineales. Una vez derivadas las ecuaciones con las cuales se iba a hacer la implementación del código en el lenguaje de programación FORTRAN-90. Después, se llevó a cabo un estudio numérico de estabilidad, probando diferentes valores de velocidad, coeficiente de difusión, Δx y Δt , esto para diferentes números de nodos.

Para la parte de implementación al estado de Morelos, se construyó un perfil de viento usando datos de 27 estaciones meteorológicas del estado de los meses de octubre y noviembre de 2010. Y

se aproximaron algunas trayectorias para probar el código con valores reales.

Se tomaron unos pasos preliminares para la implementación de la ecuación en dos dimensiones. Una de las ventajas de FEM es la aplicación de una malla irregular, la cual da mayor flexibilidad y mejor adaptación a la topología. Se llevó a cabo la triangulación de la malla de tipo Delaunay, se construyó vía el algoritmo de Bowyer-Watson [5, 6] y se implementó el código.

AGRADECIMIENTOS

Quiero agradecer a mis directores de tesis, al Dr. Minhuy y al Dr. Hugo por su disponibilidad, apoyo y orientación a lo largo del proyecto,

También agradezco a los miembros del comité: Dr. Thomas Buhse, Dr. Jorge Rivera , Dr. Roberto Bernal y Dr. Mario Murillo por su apoyo, comentarios e interés mostrado por el trabajo.

Índice general

1..	<i>Introducción</i>
1.1.	Planteamiento del problema
1.2.	Hipótesis
1.3.	Objetivo
2..	<i>Fundamentación Teórica</i>
2.1.	Antecedentes
2.1.1.	Ecuaciones Diferenciales Parciales
2.1.2.	Ecuación de Advección
2.1.3.	Ecuación de Difusión
2.1.4.	Ecuación de Reacción
2.1.5.	Condiciones iniciales y de frontera
2.1.6.	Métodos Numéricos para Ecuaciones Diferenciales Parciales
2.1.7.	Método de Diferencia Finita
2.1.8.	Método de Volumen Finito
2.1.9.	Método de Elemento Finito
2.2.	Triangulación de Delaunay
2.2.1.	Algoritmo de Bowyer-Watson
3..	<i>Metodología</i>
3.1.	Método de Elemento Finito
3.2.	ADR en 1D Estacionaria
3.2.1.	Método de Residuos Ponderados
3.2.2.	Integración por partes

3.2.3.	Funciones Base	
3.2.4.	Aproximación de Galerkin/Ritz	
3.2.5.	Esquema de Cuadratura por integración numérica	
3.3.	ADR 1D No Estacionaria	
3.3.1.	Método de Residuos Ponderados	
3.3.2.	Funciones Base	
3.3.3.	Aproximación de Galerkin/Ritz	
3.3.4.	Métodos Numéricos para EDO	
3.4.	Implementación del código	
3.5.	Pruebas de estabilidad	
3.5.1.	Número de Péclet	
3.5.2.	Número de von Neumann	
3.6.	Triangulación de Delaunay	
3.6.1.	Implementación	
3.6.2.	Ordenamiento de los datos	
3.6.3.	Supertriángulo	
3.6.4.	Circuncírculos	
3.6.5.	Aristas compartidas	
3.6.6.	Añadir triángulos nuevos	
3.6.7.	Quitar supertriángulo	
3.7.	Aplicación al estado de Morelos	
3.7.1.	Perfil de viento	
4.	<i>Resultados y Discusión</i>	
4.1.	Resultados del estudio de estabilidad	
4.2.	Perfil de Viento y Trayectorias	
4.3.	Malla de Delaunay	

5.. *Conclusiones*

6.. *Apéndices*

 6.1. Código de FEM en 1D

 6.2. Código de triangulación de Delaunay

Índice de figuras

2.1.	Esquema de una varilla para la ecuación de advección.
2.2.	Esquema de una varilla para la ecuación de difusión.
2.3.	Triangulación de Delaunay.
3.1.	Cuatro funciones triangulares $\varphi_i(x)$
3.2.	Función triangular $\varphi_i(x)$ centrada en x_i
3.3.	Círculo que contiene todos los puntos.
3.4.	Supertriángulo con círculo adentro.
3.5.	Primer punto dentro del supertriángulo y los primeros 3 triángulos.
3.6.	Segundo punto dentro del supertriángulo.
3.7.	Circuncírculos de cada triángulo.
3.8.	Circuncírculos de cada triángulo.
3.9.	Se agrega el punto 3 y se calculan los circuncírculos.
3.10.	Mapa con 27 estaciones [7].
3.11.	Mapa con vectores.
3.12.	Vectores con líneas.
3.13.	Representación de la ecuación 3.31.
3.14.	Vectores para las trayectorias.
3.15.	Trayectorias.
4.1.	Gráficas de los pasos 1, 50, 100.
4.2.	Gráficas de los pasos 1, 50, 100.
4.3.	Gráficas de los pasos 1, 50, 100.
4.4.	Gráficas de los pasos 1, 50, 100.

4.5.	Gráficas de concentración inicial y pasos 25, 100 y 200.
4.6.	Gráficas de concentración inicial y pasos 25, 100 y 200.
4.7.	Gráficas de concentración inicial y pasos 10, 75 y 100.
4.8.	Gráficas de concentración inicial y pasos 10, 75 y 100.
4.9.	Gráficas de concentración inicial y pasos 25, 75 y 100.
4.10.	Gráficas de concentración inicial y pasos 25, 75 y 100.
4.11.	Gráficas de concentración inicial y pasos 25, 75 y 150.
4.12.	Gráficas de concentración inicial y pasos 25, 50 y 75.
4.13.	Gráficas de concentración inicial y pasos 25, 50 y 75.
4.14.	Gráfica de la concentración inicial y un suministro.
4.15.	Gráficas de: concentración inicial y pasos 15, 25 y 30.
4.16.	Malla de 10 puntos.
4.17.	Malla de 30 puntos.
4.18.	Malla de 150 puntos.
4.19.	Diferencias en el convex hull 10 puntos.
4.20.	Diferencias en el convex hull 30 puntos.

Índice de cuadros

- 3.1. Estaciones meteorológicas.
- 3.2. Valores de velocidad $v(km/h)$, ángulo θ y dirección de viento del 28 de octubre 2010.
- 3.3. Valores de posición x y velocidad $v(km/h)$ para la trayectoria 1.
- 3.4. Valores de posición x y velocidad $v(km/h)$ para la trayectoria 2.
- 3.5. Valores de posición x y velocidad $v(km/h)$ para la trayectoria 3.
- 4.1. Valores de coeficientes de difusión a 273 K y 0.1 MPa.

1. INTRODUCCIÓN

1.1. *Planteamiento del problema*

En las últimas décadas la contaminación del aire ha sido objeto de investigación a nivel mundial, debido a los daños que provoca a la salud humana y al ambiente. Por tal motivo, se requiere de su estudio para poder entender mejor el fenómeno y poder aproximar los impactos que ocasiona, así como las principales fuentes, y más aún, buscar posibles soluciones que ayuden a mitigar y/o controlar dichos impactos.

Para poder interpretar de manera adecuada el comportamiento de la contaminación atmosférica se ha recurrido en los últimos años al uso de herramientas computacionales que permiten integrar las diversas variables asociadas a este fenómeno entre las que destacan variables meteorológicas como velocidad y dirección de viento. Esto con el propósito de predecir las posibles fuentes de contaminantes, su transporte y posibles transformaciones químicas.

Actualmente, existe una serie de herramientas estadísticas y computacionales que permiten evaluar el comportamiento de contaminantes en las diferentes esferas ambientales; en la mayoría de los casos, el acceso es restringido o se carece del conocimiento técnico y científico para su implementación. Por tal motivo, continuando con el desarrollo del presente proyecto, se pretende desarrollar un modelo que permita predecir el comportamiento de la contaminación del aire en el estado de Morelos, tomando como punto de partida el trabajo que se logró en la tesis de licenciatura.

Este trabajo consiste en programar un modelo, el cual sea capaz, por medio del uso de métodos numéricos, de modelar los fenómenos de advección, difusión y reacción de los principales contaminantes del aire haciendo uso de la ecuación de Advección-Difusión-Reacción (ADR) (Ecn. (1.1) en 1D)

$$\frac{\partial U(x,t)}{\partial t} = -v \frac{\partial U(x,t)}{\partial x} + \alpha^2 \frac{\partial^2 U(x,t)}{\partial x^2} + F(x,t) \quad (1.1)$$

Se busca mejorar la aproximación matemática probando un método numérico diferente, como el método de elemento finito (FEM), que mejore la discretización de la ecuación de ADR. Anteriormente, se intentó con el método de diferencias finitas aplicando el método de discretización de Crank-Nicolson, para la ecuación de advección en 2D se utilizó el esquema de diferencia adelantada en tiempo, diferencia regresiva en espacio y para la ecuación de difusión el esquema de diferencia adelantada en tiempo, diferencia centrada en espacio. Una vez que el perfil haya sido determinado adecuadamente, el valor de la velocidad en cada punto de la malla se usará para resolver la ecuación.

1.2. Hipótesis

La velocidad y dirección del viento cambian con respecto a la altura, especialmente cerca del suelo [8]. El viento va a seguir la ruta de menor energía, por lo que se relaciona con las líneas de nivel. Se piensa que cuando el viento se encuentra con un obstáculo, como pueden ser montañas, edificios, entre otros, la ruta de menor energía sería rodearlos en lugar de pasar por arriba de ellos. Así mismo, el decir que el viento sigue las líneas de contorno va de la mano con la velocidad, ya que al encontrarse con esos obstáculos va implícito la disminución de ésta. Esto va a depender de la altura a la que se decida monitorear el flujo del viento. Donde no hay obstáculos, que sería a mayor altura, el flujo sería mucho más continuo, ya que la velocidad disminuye con cambios en las trayectorias. Así mismo, algo que se tiene que tomar en cuenta es la dirección de las trayectorias, es decir, si el viento va de norte a sur o viceversa. Esto debido a la topología del estado, ya que en el norte del estado es mayor la altura, que en el sur del estado. También, el relieve del estado presenta más montañas en el sur, donde colinda con el estado de Guerrero.

1.3. *Objetivo*

Mediante el uso de la ecuación de ADR, se modelarán los fenómenos de advección, difusión y reacción para contaminantes en el aire empleando el método de elemento finito mediante el uso del lenguaje de programación Fortran 90.

Para su implementación, se se recopilarán datos de velocidad y dirección de viento de las estaciones meteorológicas para crear el perfil de viento. Siguiendo los vectores del perfil de viento conseguiremos las trayectorias, con las cuales haremos pruebas cambiando los valores de α y Δt .

2. FUNDAMENTACIÓN TEÓRICA

2.1. Antecedentes

Anteriormente, el perfil de viento se hizo con datos de 4 estaciones meteorológicas, tomando el promedio de velocidad y dirección de viento de los días de muestreo en los meses de octubre y noviembre. Ahora, no se tomó el promedio, sino que se tomaron los valores de un día para todas las estaciones, que ahora se tienen 27 en todo el estado.

2.1.1. Ecuaciones Diferenciales Parciales

La ecuación ADR es una ecuación diferencial parcial que se utiliza ampliamente para representar fenómenos físicos, especialmente para el modelado de la transferencia de calor y energía, ya que toman en cuenta todos los parámetros necesarios del problema. Asimismo se ha utilizado ampliamente para describir los fenómenos de transporte. Existe una literatura abundante de su empleo en estudios de difusión de la contaminación. Para una revisión se puede consultar las referencias de Coiffier y Stensrud [3, 9]. Además, la ecuación de ADR está estrechamente relacionada con la ecuación de Navier-Stokes. El fenómeno de transporte consiste, en este caso, en la advección de un contaminante con concentración $U(x, t)$, función de espacio x y tiempo t , llevado por la velocidad v . Debido al movimiento aleatorio browniano el transporte involucra también la difusión que depende en el coeficiente de difusión α y la curvatura de la concentración. Las reacciones que ocurren en el transcurso de la trayectoria se describen por la función $F(x, t)$, puede ser que disminuyan la concentración (sumidero) o la aumenten (suministro). Puesto que la ecuación de ADR en 1D toma la forma [10, 11, 3]:

$$\frac{\partial U(x, t)}{\partial t} = -v \frac{\partial U(x, t)}{\partial x} + \alpha^2 \frac{\partial^2 U(x, t)}{\partial x^2} + F(x, t) \quad (2.1)$$

A continuación, se discute la derivación de la parte advección y difusión.

2.1.2. Ecuación de Advección

En el contexto de este trabajo, la advección denota la propagación de forma horizontal del viento. En meteorología el término de advección denota la propagación de forma horizontal del viento [10]. Cabe mencionar que en la literatura matemática la convección se define como la suma de advección y difusión.

En una dimensión, la ecuación parabólica de advección se escribe:

$$\frac{\partial U}{\partial t} = -v \frac{\partial U}{\partial x} \quad U_t = -v U_x$$

Para la derivación de esta ecuación consideramos la concentración de una materia que se expresa por $U(x, t)$ en un segmento $[x - \frac{1}{2}h, x + \frac{1}{2}h]$ $h > 0$.

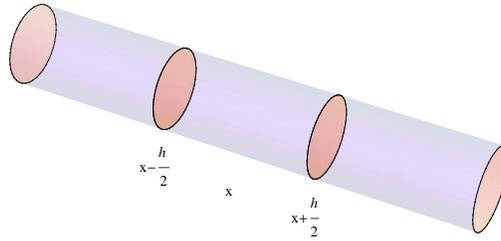


Fig. 2.1: Esquema de una varilla para la ecuación de advección.

Si el medio lleva la materia con una velocidad $v(x, t)$, entonces el flujo en la frontera $x - \frac{1}{2}h$ es:

$$v(x - \frac{1}{2}h, t)U(x - \frac{1}{2}h, t)$$

La diferencia entre las dos secciones transversales nos da el cambio de la concentración en este segmento y si dividimos por h tenemos el cambio de la concentración promedio $\bar{U}(x, t)$

$$\frac{\partial \bar{U}(x, t)}{\partial t} = \frac{1}{h} \left[v(x - \frac{1}{2}h, t)U(x - \frac{1}{2}h, t) - v(x + \frac{1}{2}h, t)U(x + \frac{1}{2}h, t) \right]$$

Cuando h tiende a 0, el lado derecho nos da la negativa de la derivada de $v(x, t)U(x, t)$ con respecto a x , obtenemos entonces, la ecuación de advección

$$U_t = -vU_x \quad (2.2)$$

considerando que v es constante con respecto a x .

2.1.3. Ecuación de Difusión

La ecuación de difusión, conocida también como la ecuación de calor, es una ecuación diferencial parcial de tipo parabólica, que describe el flujo de concentración y el proceso de difusión. El cambio de la concentración con tiempo se expresa como:

$$\frac{\partial U(x, t)}{\partial t} = \alpha^2 \frac{\partial^2 U(x, t)}{\partial x^2} \quad U_t = \alpha^2 U_{xx}$$

La derivación de ésta ecuación tiene como base el concepto de conservación de la energía [12]. Consideremos una varilla de longitud L que funciona según las siguientes hipótesis:

- La materia de la varilla es homogénea, el flujo de calor es uniforme.
- La varilla está aislada, es decir, el flujo de calor ocurre sólo en la dirección x .
- La temperatura es uniforme en la sección transversal entera.

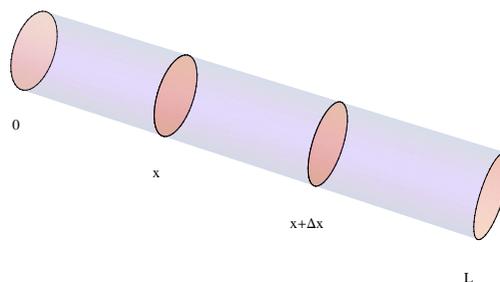


Fig. 2.2: Esquema de una varilla para la ecuación de difusión.

La conservación de energía en el segmento $[x, x + \Delta x]$ requiere que el cambio neto del calor dentro del segmento sea igual a la suma del flujo de calor neto a través de las fronteras y el cambio de calor generado dentro del segmento. El calor interno del segmento en el punto x depende en

la capacidad calorífica c , la densidad de la varilla ρ y la sección transversal de la varilla A . Así, el calor interno del segmento es la integral de x a Δx :

$$\int_x^{x+\Delta x} c\rho AU(x,t)dx$$

donde $U(x,t)$ es la temperatura en x,t y el producto $c\rho AU(x,t)$ es el calor en x . El cambio de la temperatura interna con respecto al tiempo es la derivada de la integral

$$\frac{d}{dt} \int_x^{x+\Delta x} c\rho AU(x,t)dx = c\rho A \int U_t(x,t)dx.$$

Por otro lado podemos escribir el cambio de la temperatura como la diferencia de ésta en las dos fronteras que depende en la conductividad térmica k :

$$kAU_x(x+\Delta x,t) - kAU_x(x,t) = kA \left[U_x(x+\Delta x,t) - U_x(x,t) \right]$$

y el calor generado dentro del segmento causado por una fuente externa o por una reacción interna descrita por $f(x,t)$ es

$$A \int_x^{x+\Delta x} f(x,t)dx$$

Por fin la ecuación de la conservación de energía es

$$c\rho A \int U_t(x,t)dt = kA[U(x+\Delta x,t) - U(x,t)] + A \int f(x,t)dx \quad (2.3)$$

donde la primera parte de la ecuación es el cambio total, la segunda el cambio a través de las fronteras y la última parte es el cambio causado por una fuente externa. Ahora el problema es reemplazar la última ecuación por una que no tenga integrales, así que se utiliza el teorema del valor medio, el cual dice que si $f(x)$ es una función continua en $[a,b]$, entonces existe un punto ξ tal que

$$\int_a^b f(x)dx = f(\xi)(b-a)$$

Aplicamos ese teorema a las integrales de la Ecn. (2.3), y obtenemos:

$$\int_x^{x+\Delta x} U_t(x, t) dx = U_t(\xi_1, t) \Delta x$$

$$\int_x^{x+\Delta x} f(x, t) dx = f(\xi_2, t) \Delta x$$

Se sustituyen los resultados en la ecuación de conservación de energía

$$c\rho AU_t(\xi_1, t) \Delta x = kA[U(x + \Delta x, t) - U(x, t)] + Af(\xi_2, t) \Delta x$$

Cuando Δx tiende a cero, ξ_1 y ξ_2 tienden a un valor x , así que se sustituye y se despeja para $U_t(x, t)$

$$U_t(x, t) = \frac{k}{c\rho} U_{xx}(x, t) + \frac{1}{c\rho} f(x, t), \quad (2.4)$$

la cual da la forma más común de presentar la ecuación de difusión si se sustituye el coeficiente de difusión $\alpha^2 = \frac{k}{c\rho}$ y $F(x, t) = \frac{1}{c\rho} f(x, t)$:

$$U_t = \alpha^2 U_{xx} + F(x, t) \quad (2.5)$$

Por fin podemos combinar la ecuación de advección con la ecuación de difusión para llegar a la ecuación de ADR Ecn. (2.1):

$$U_t = -vU_x + \alpha^2 U_{xx} + F(x, t) \quad (2.6)$$

2.1.4. Ecuación de Reacción

La parte de reacción son los cambios de concentración que se tiene a lo largo de la trayectoria, ya sea el aumento o la disminución de la concentración. Se puede considerar desde la perspectiva de la cinética química para una reacción en particular. Puede ser de dos tipos diferentes:

- Por agentes externos, es decir, sumistros y/o sumideros que se encuentran a lo largo de la trayectoria o cercana a ésta y que tengan un efecto.

- Por reacciones químicas y el tipo de contaminante que se este monitoreando, es decir, ya sea un contaminante primario o secundario. Según el Instituto Nacional de Ecología y Cambio Climático (INECC) [13] los contaminantes se dividen en dos grandes grupos:
 - *Contaminantes primarios*: son aquellos procedentes directamente de la fuente de emisión, por ejemplo, plomo (Pb), monóxido de carbono (CO), óxidos de azufre (SO_x), óxidos de nitrógeno (NO_x), entre otros.
 - *Contaminantes secundarios*: son aquellos que se originan por reacciones en el aire entre dos o más contaminantes primarios o con componentes naturales del aire, por ejemplo sulfatos (SO₄²⁻), nitratos (NO₃⁻), ozono (O₃), material particulado (MP), entre otros.

Ambas situaciones se describen por la función $F(x, t)$. Se usa una función exponencial para cualquiera de las dos opciones, una exponencial creciente o decreciente.

La función que se tiene actualmente en el código es

$$e^{(-(x-75)^2/100)} \left(1 + \frac{x - x_i}{x_i - x_{i-1}} \right) \qquad e^{(-(x-75)^2/100)} \left(1 - \frac{x - x_i}{x_{i+1} - x_i} \right)$$

Se necesitan dos ecuaciones porque, como se verá más adelante, la función de base que se va a usar es una función por tramos, por lo que se necesitan una función para el tramo ascendente de la función triangular y uno para el tramo descendente (Fig. 3.2).

2.1.5. Condiciones iniciales y de frontera

Para las ecuaciones diferenciales parciales se tiene que tomar en cuenta las condiciones iniciales y de frontera en un intervalo de espacio establecido, en este caso vamos a considerar el intervalo de espacio $a \leq x \leq b$ y el intervalo de tiempo $t \geq 0$. En cuanto a la modelación inicial, la fuente de contaminación se expresa como una distribución tipo Gauss, que simula una chimenea con una altura y un ancho:

$$U(i, 0) = \text{altura} \times \exp[-((x - \text{centro})/\text{ancho})^2]$$

Hay diferentes tipos de condiciones de frontera [14]

- Dirichlet (*esencial*): la solución es conocida en la frontera

$$U(a, t) = 0 \quad U(b, t) = 0$$

- Neumann (*natural*): la derivativa normal es dada a lo largo de la frontera

$$\frac{\partial U}{\partial x}(a, t) = g(t) \quad \frac{\partial U}{\partial x}(b, t) = h(t)$$

- Robin: una mezcla, donde a y b son conocidas.

$$a \frac{\partial U}{\partial x} + b U$$

Cualquiera de estas condiciones son llamadas homogéneas cuando la función específica es igual a 0. Por ejemplo, si una cuerda se mantiene fija por los dos extremos, se tendría una condición Dirichlet homogénea. Por otra parte, si un extremo está libre para moverse sin resistencia transversalmente, no habría tensión en b , por lo que $U_x = 0$. Para la condición de Robin, se podría imaginar que la cuerda se puede mover libremente pero unida a un resorte o liga que regrese la cuerda a la posición de equilibrio [15].

2.1.6. Métodos Numéricos para Ecuaciones Diferenciales Parciales

En general, no existe solución analítica a la ecuación de ADR para condiciones arbitrarias de las fronteras, velocidad, etc., aunque hay varios estudios en esta dirección [16]. En la práctica, es común emplear métodos numéricos. La idea esencial de los métodos numéricos es la discretización donde se particiona la región de interés en una malla y se busca la solución de la EDP sólo en los puntos de esta malla. Matemáticamente, esto nos lleva a la aproximación de la derivada por diferentes métodos, por ejemplo el método de diferencia finita, método de elemento finito, método de volumen finito, entre otros.

Para FEM la triangulación es de primordial importancia.

2.1.7. Método de Diferencia Finita

El método de diferencia finita es el método con la aproximación más directa para discretizar las EDP. Se considera un punto en el espacio donde se toma la representación continua de la ecuación y se reemplaza con un grupo de ecuaciones discretas, llamadas ecuaciones diferenciales finitas. Este método generalmente es definido con una malla regular [17].

La meta es aproximar la solución a una ecuación diferencial, es decir, encontrar una función que satisfaga a varias derivadas en un espacio y tiempo dado. El método de diferencia finita reemplaza las derivadas en una ecuación diferencial con aproximaciones finitas diferenciales. Esto da un sistema de ecuaciones algebraicas para ser resueltas en lugar de la ecuación diferencial [18].

Sea $U(x, t)$ una función cuya n -ésima derivada parcial con respecto a x existe en un intervalo abierto que contiene los puntos x_i y h . La serie de Taylor de $U(x, t)$ alrededor el punto x_i es:

$$U(x_i + h, t) = U(x_i, t) + h \frac{\partial U(x, t)}{\partial x} + \frac{h^2}{2!} \frac{\partial^2 U(x, t)}{\partial x^2} + \dots + \frac{h^n}{n!} \frac{\partial^n U(x, t)}{\partial x^n} + R_n(x) \quad (2.7)$$

donde $R_n(x)$ es el resto de Lagrange:

$$R_n(x) = \frac{h^{n+1}}{(n+1)!} \frac{\partial^{n+1} U}{\partial x^{n+1}} \Big|_c \quad x \leq c \leq h$$

Si aproximamos $U(x_i + h, t)$ al primer orden

$$U(x_i + h, t) = U(x_i, t) + h \frac{\partial U}{\partial x} + R_1(x)$$

$$U(x_i + h, t) \approx U(x_i, t) + h \frac{\partial U}{\partial x}$$

y cuando despejamos para $\partial U / \partial x$

$$\begin{aligned}\frac{\partial U}{\partial x} = U_x &\approx \frac{U(x_i + h, t) - U(x_i, t)}{h} \\ &= \frac{U(x_i + h, t) - U(x_i, t)}{h} + O(h)\end{aligned}$$

donde $O(h)$ es el error de truncamiento cuya magnitud es del orden de h . Existen diferentes esquemas de discretización, dependiendo de la selección de los puntos usados en aproximar las derivadas.

Diferencia Adelantada: en este esquema se aproxima la derivada por la diferencia de la función en un punto y otro punto adelante en la dirección positiva de x

$$U_x \approx \frac{U(x + h, t) - U(x, t)}{h}$$

Diferencia Regresiva: el caso reverso, cuando el punto final de la diferencia es menor que el punto inicial, tenemos el esquema regresivo

$$U_x \approx \frac{U(x, t) - U(x - h, t)}{h}$$

Diferencia Centrada: como el nombre lo implica, la diferencia se realiza simétricamente sobre los dos puntos

$$U_x \approx \frac{U(x + h, t) - U(x - h, t)}{2h}$$

2.1.8. Método de Volumen Finito

El método de volumen finito es similar al método de elemento finito en cuanto a los elementos finitos de formas geométricas simples. Aparte de esto, este método es diferente, en cuanto a que los elementos se llaman celdas. Se basa en las leyes físicas de conservación, lo que entra en una celda por un lado necesita salir por la misma celda por el otro lado. Esta idea consiste en la formulación de la ecuación de flujo sobre las celdas [17].

2.1.9. Método de Elemento Finito

En matemáticas, el método de elemento finito es una técnica numérica para encontrar soluciones aproximada a problemas de valores de frontera para ecuaciones diferenciales parciales, en el cual, el dominio se divide en un número finito elementos para simplificarlo, llamados elementos finitos, los cuales constituyen la malla con formas geométricas simples [19, 17].

Una de las razones por las cuales este método es más utilizado es que es un método muy general.

2.2. Triangulación de Delaunay

La discretización del dominio de las variables es la idea fundamental de los métodos numéricos de las ecuaciones diferenciales [18], y para el método de elemento finito la triangulación es el corazón de la discretización. En este método la malla es irregular, lo cual nos da mayor flexibilidad para representar mejor el dominio y que se adapte mejor a la topología, usando el mapa de relieve con líneas de contorno para tener una nueva malla.

Según la característica de Euler-Poncaré, para v vértices y c caras de *envolvente convexa* (*convex hull*), tenemos la relación con el número de triángulos t y aristas a :

$$t = 2v - 2 - c \qquad a = 3v - 3 - c$$

Hay varias maneras en las que se pueden conectar los vértices de un conjunto de puntos para formar triángulos.

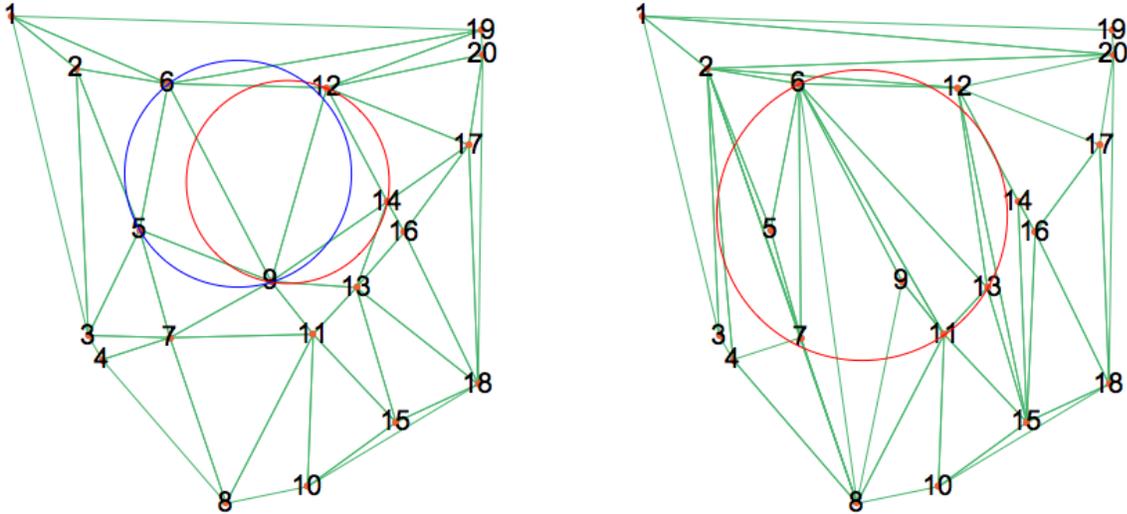


Fig. 2.3: Triangulación de Delaunay.

Entre todos los métodos de triangulación para FEM, probablemente el método más empleado es el de Delaunay, debido a la restricción que el método impone a los triángulos: sus ángulos mínimos son maximizados. Es decir la malla de Delaunay minimiza los triángulos que son muy agudos y esto ayuda llevar a las soluciones más estables, ya que un triángulo muy agudo se extiende sobre una zona extensa y probablemente sobre un gran cambio de la función y sus derivadas. Para minimizar los triángulos no deseados, los triángulos de Delaunay satisfacen la condición de que *cualquier círculo definido por uno de estos triángulos no circunscribe otros puntos*. En la Fig. 2.3 podemos ver esta condición, ya que la imagen del lado derecho no satisface la condición de triangulación de Delaunay. Hoy en día, la malla de Delaunay ha sido empleada, entre otros, en la autenticación de la huella digital, imagen mediante visión artificial, diseño ingeniería, etc.

Trabajamos implementando el algoritmo de Bowyer-Watson [20, 21], el cual consiste en encerrar los puntos con un supertriángulo e ir insertando un punto a la vez a la malla, con el cual se generarán nuevos triángulos con los demás puntos. Esto sucederá para los triángulos que circunscriben al nuevo punto, es decir, se revisará si el punto está adentro de los circuncírculos de los triángulos y si es así se quitarán las aristas de éstos triángulos para formar los nuevos triángulos correspondientes.

2.2.1. Algoritmo de Bowyer-Watson

1. Ordenar datos en orden lexicográfico.
2. Generar el supertriángulo.
3. Formar los primeros 3 triángulos con el primer punto y los vértices del supertriángulo.
4. Agregar el siguiente punto, $i = 2$.
 - a) Checar adentro de que circuncírculos (triángulos) está adentro el punto i .
 - b) Hacer una lista de todas las aristas de los triángulos en los cuales está adentro el punto i .
 - c) Eliminar aristas repetidas de la lista, ya que las restantes formarán los nuevos triángulos con el punto i .
 - d) Agregar los nuevos a la lista de triángulos.
5. Repetir el paso 4 hasta terminar los datos.

3. METODOLOGÍA

3.1. Método de Elemento Finito

Para cualquier problema, el uso de FEM consiste en cuatro conceptos básicos [4]:

- Forma fuerte
- Forma débil
- Aproximación por elemento finito
- Sistema de ecuaciones lineales

Las ecuaciones diferenciales expresan problemas físicos y comprenden una ecuación matemática a partir de reglas generales, respetando los principios constitutivos, leyes de conservación, etc. Los problemas son complementados por condiciones de frontera, las cuales proveen las propiedades de lo que sucede en las fronteras del dominio. La ecuación diferencial y las condiciones de frontera componen la *forma fuerte* del problema. Para la mayoría de los casos, no hay una solución analítica, por lo que es necesario utilizar métodos numéricos para obtener la solución aproximada. En el método de elemento finito, es necesario la transformación de la forma fuerte a la expresión equivalente llamada *forma débil*, la cual involucra ecuaciones integrales, ya que es más conveniente para la aproximación. Después, el método de elemento finito se va a aplicar a la forma débil, que permitirá transformar el problema de una forma con ecuaciones integrables a un sistema de ecuaciones lineales, en el cual las incógnitas serán constantes, comprendidas en un vector $\{\alpha\}$. Por lo que se tendrá el sistema matricial $[K]\{\alpha\} = \vec{F}$, donde $[K]$ es la matriz de carga y \vec{F} es el vector de carga.

3.2. ADR en 1D Estacionaria

La primera parte de la aproximación se hizo con la forma estacionaria de la ecuación ADR, es decir, el término de $U_t = 0$.

$$\frac{\partial U(x,t)}{\partial t} = 0 = -v \frac{\partial U(x,t)}{\partial x} + \alpha^2 \frac{\partial^2 U(x,t)}{\partial x^2} + F(x,t)$$

La forma fuerte de esta ecuación consiste en la ecuación diferencial junto con las condiciones iniciales y de frontera. La forma fuerte para la ecuación de ADR estacionaria en tiempo, para una dimensión es la siguiente:

$$v \frac{\partial U}{\partial x} - \alpha^2 \frac{\partial^2 U}{\partial x^2} = f(x) \quad 0 < x < L \quad x \in \Omega$$

3.2.1. Método de Residuos Ponderados

FEM es un método de tipo de residuos ponderados y hace uso de la forma débil de la ecuación. Si se tiene la solución exacta de la ecuación el residuo será 0 en todos los puntos del dominio. Sin embargo, como se tiene que aproximar la solución el residuo no es 0, por lo que se busca minimizarlo

$$\int_0^L w(x) R(x) dx = 0 \quad (3.1)$$

donde $R(x)$ es la función de ADR en 1D, $w(x)$ es la función de prueba y el dominio es $[0, L]$.

$$R(x) = v \frac{\partial U}{\partial x} - \alpha^2 \frac{\partial^2 U}{\partial x^2} - f(x)$$

Se sustituye la $R(x)$ en la Ecn. (3.1)

$$\int_0^L \left(v w(x) \frac{\partial U}{\partial x} - \alpha^2 \frac{\partial^2 U}{\partial x^2} w(x) - f(x) w(x) \right) dx = 0 \quad (3.2)$$

para obtener la minimización de manera promedio. Como se puede observar, en la parte de difusión el segundo elemento de la Ecn. (3.2) tiene derivadas de segundo orden, por lo que se tiene que aplicar integración por partes para disminuir o “equilibrar” la derivada con la función de prueba $w(x)$. Igualmente para que la función de prueba cumpla con las condiciones del espacio de Sobolev. Y posteriormente se verá que beneficiará a la matriz resultante, que una matriz tridiagonal y será simétrica, y de este modo simplifica el método numérico requerido.

3.2.2. Integración por partes

Se hace un recordatorio de integración por partes

$$\begin{aligned}\int_0^L (-u''v)dx &= -u'v \Big|_0^L + \int_0^L u'v'dx \\ &= \int_0^L u'v'dx\end{aligned}$$

La integración por partes de la parte de difusión

$$\int_0^L \frac{\partial^2 U}{\partial x^2} w(x) dx = \frac{\partial U}{\partial x} w(x) \Big|_0^L - \int_0^L \frac{\partial w(x)}{\partial x} \frac{\partial U}{\partial x} dx \quad (3.3)$$

Por simplicidad, se toma $B_0 = B_L = 0$, es decir $U(x, t)$ es cero en las fronteras y el primer término derecho es cero. Se sustituye la parte de difusión en la Ecn. (3.2) y se obtiene la forma débil

$$\int_0^L \left(v w(x) \frac{\partial U}{\partial x} + \alpha^2 \frac{\partial w(x)}{\partial x} \frac{\partial U}{\partial x} \right) dx = \int_0^L w(x) f(x) dx \quad (3.4)$$

Se llama débil porque en esta forma se requiere únicamente que $U(x, t) \in C_D^0\{0, L\}$, es decir, que sólo exista la primera derivada U_x en lugar de la segunda U_{xx} . Esta es una de las ventajas de la forma débil sobre la forma fuerte. Además en unos casos, las matrices resultantes son simétricas y de este modo simplifica el método numérico requerido. Se puede mostrar que el uso de esta solución es sólo para la conveniencia matemática y se puede mostrar que las dos formas, fuerte y débil, son absolutamente iguales [4]. Esta es la ecuación con la que se va a aplicar la aproximación

de Galerkin. Como se puede ver, la segunda derivada en la parte de difusión ya no está, sólo se va a trabajar con derivadas de primer grado.

3.2.3. Funciones Base

La función de prueba se debe elegir de tal manera que cumpla con la condición de frontera del problema inicial y el grado de derivada que se vaya a utilizar. Se define el *espacio de Sobolev* donde se describe la clase de función que se puede elegir

$$H_0^L = \left\{ \varphi : [0, L] \rightarrow \mathbb{R} \mid \int_0^L |\varphi|^2 dx < \infty, \int_0^L |\varphi'|^2 dx < \infty, \varphi(0) = \varphi(L) = 0 \right\}$$

El superíndice 1 de H significa que miembros de esta clase de funciones deben de tener derivadas de orden 1 y que el cuadrado de la función es integrable en el intervalo $0 < x < L$. El subíndice 0 indica que la función es igual a 0 en $x = 0$ y $x = L$. El resto indica que la función al cuadrado y su primera derivada al cuadrado, ambas, deben ser finitas [22]. La idea principal de FEM es que la función base φ_i pueda ser definida sobre las subregiones del dominio llamadas *elementos finitos*, Ω_i , y sobre cualquier subdominio. La distancia entre cada elemento se denota como h_i . Entre cada elemento, se identifican ciertos puntos, llamados *nodos* o *puntos nodales*. El conjunto de éstos elementos, elementos y nodos, componen el dominio y algunas veces se refieren a la *mallla de elementos finitos* [22]. Una vez construida la mallla, se debe construir el conjunto de funciones base, tomando en cuenta los siguientes criterios:

- Las funciones de base son generadas por funciones simples definidas elemento por elemento, sobre la mallla
- Son suficientemente suaves

Ya que tenemos la forma débil de la ecuación se aproxima $U(x, t)$ como una combinación de N elementos finitos $\varphi_i(x)$

$$U(x) = \sum_{i=1}^N c_i \varphi_i(x) \tag{3.5}$$

donde c_i son los valores en los nodos, los cuales son los que vamos a calcular, N es el número de nodos en la malla y $\varphi_i(x)$ son las funciones base que se usarán para construir la solución aproximada.

Una de las formas más empleadas de $\varphi_i(x)$ en FEM son las funciones por tramos conocidas como funciones triangulares o funciones “sombrero”

$$\varphi_i(x) = \begin{cases} 1 + \frac{x-x_i}{x_i-x_{i-1}} & x_{i-1} \leq x < x_i \\ 1 - \frac{x-x_i}{x_{i+1}-x_i} & x_i \leq x < x_{i+1} \\ 0 & x \notin \{x_{i-1}, x_{i+1}\} \end{cases} \quad (3.6)$$

La ventaja de usar los elementos finitos $\varphi_i(x)$ es directa; las funciones triangulares se traslapan solo con sí misma y con las dos funciones vecinas inmediatas (Fig. 3.1), lo que resulta en matrices tridiagonales.

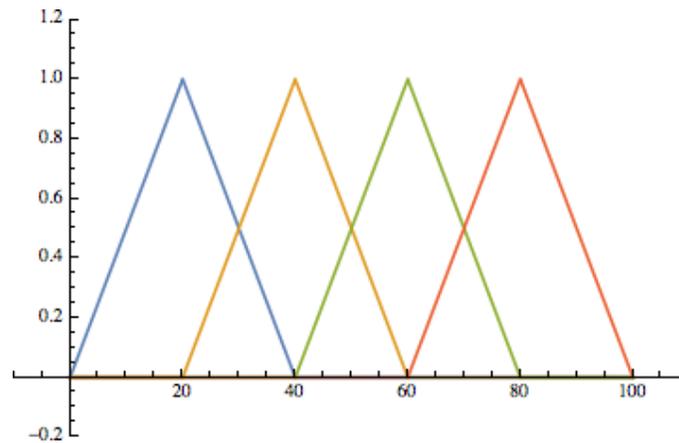


Fig. 3.1: Cuatro funciones triangulares $\varphi_i(x)$.

No se ha impuesto ninguna restricción sobre la distancia entre los nodos, h_i , se puede elegir cualquier cualquier valor para h_i para formar cualquier malla deseada. De hecho, una de las ventajas principales de FEM es la flexibilidad en adaptar la malla a la topología particular del dominio Ω . La Fig. 3.2 muestra el elemento finito centrada en x_i descrito por la función triangular $\varphi_i(x)$.

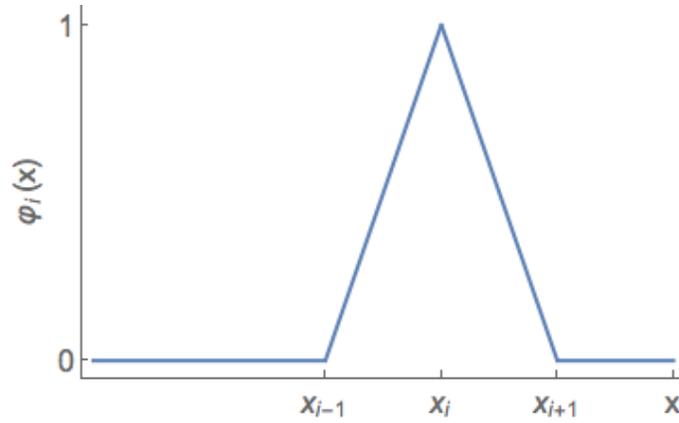


Fig. 3.2: Función triangular $\varphi_i(x)$ centrada en x_i .

La diferenciación de $\varphi_i(x)$ con respecto a x es sencillamente:

$$\frac{d\varphi_i(x)}{dx} = \varphi_i'(x) = \begin{cases} \frac{1}{x_i - x_{i-1}} & x_{i-1} \leq x < x_i \\ -\frac{1}{x_{i+1} - x_i} & x_i \leq x < x_{i+1} \\ 0 & x \notin \{x_{i-1}, x_{i+1}\} \end{cases} \quad (3.7)$$

Se sustituye la solución aproximada (Ecn. (3.5)) en la ecuación débil (Ecn. (3.4))

$$\int_0^L \left[v w(x) \left(\sum_{i=1}^N c_i \frac{\partial \varphi_i}{\partial x} \right) + \alpha^2 \frac{\partial w(x)}{\partial x} \left(\sum_{i=1}^N c_i \frac{\partial \varphi_i}{\partial x} \right) \right] dx = \int_0^L w(x) f(x) dx \quad (3.8)$$

3.2.4. Aproximación de Galerkin/Ritz

Una de las aproximaciones más usadas en FEM es la aproximación de Galerkin/Ritz, donde se usa la misma $\varphi_j(x)$ como función de prueba $w(x) = \varphi_j(x)$. La salida numérica de la solución débil involucra en la conversión de la última en la forma discreta, por lo que haciendo esta sustitución en la Ecn. (3.8) se obtiene

$$\int_0^L \left[\varphi_j(x) v \left(\sum_{i=1}^N c_i \frac{\partial \varphi_i}{\partial x} \right) + \alpha^2 \frac{\partial \varphi_j(x)}{\partial x} \left(\sum_{i=1}^N c_i \frac{\partial \varphi_i}{\partial x} \right) \right] dx = \int_0^L \varphi_j(x) f(x) dx$$

$$\sum_{j=1}^N \left[\int_0^L \left(v \varphi_j(x) \frac{\partial \varphi_i}{\partial x} + \alpha^2 \frac{\partial \varphi_j(x)}{\partial x} \frac{\partial \varphi_i}{\partial x} \right) dx \right] c_i = \int_0^L \varphi_j(x) f(x) dx \quad (3.9)$$

Donde su forma matricial es: $\mathbb{K}\alpha = \mathbf{F}$ donde α es el vector que se quiere encontrar. \mathbb{K} es la matriz de rigidez de tamaño $N \times N$

$$K_{ij} = \int_0^L \left(v \varphi_i(x) \varphi_j'(x) + \alpha^2 \varphi_i'(x) \varphi_j'(x) \right) dx \quad (3.10)$$

\mathbf{F} es el vector de carga de tamaño $N \times 1$

$$F_j = \int_0^L \varphi_j(x) f(x) dx$$

3.2.5. Esquema de Cuadratura por integración numérica

Ya que la gran mayoría de las integrales no tiene una solución analítica, es necesario utilizar los métodos numéricos, también conocido como cuadratura

$$I = \int_a^b f(x) dx \approx I_n = \sum_{i=1}^n w_i f(\xi_i) \quad (3.11)$$

donde n es el grado de la cuadratura, w_i es el peso y ξ_i son los *nodos/abcisas* para ser determinado.

En general, se consideran dos familias de integración numérica:

- Newton-Cotes (Simpson, Trapezoide, Boole ...): donde la distancia entre los nodos es igual.
- Cuadratura de Gauss: donde la distancia entre los nodos varía para adaptar mejor a la función $f(x)$.

Se aproxima $f(x)$ por el polinomio de Legendre de grado n

$$nP_n(x) = (2n - 1)xP_{n-1}(x) - (n - 1)P_{n-2}(x) \quad (3.12)$$

donde para evaluar $P_n(x)$, se necesita $P_{n-2}(x)$ y $P_{n-1}(x)$. Por ejemplo, los primeros polinomios se

calculan así:

$$\begin{aligned}
 P_0(x) &= 1 & P_1(x) &= x \\
 2P_2(x) &= (2(2) - 1)xP_{2-1}(x) - (2 - 1)P_{2-2}(x) \\
 &= 3xP_1(x) - 1P_0(x) \\
 &= 3 \cdot x \cdot x - 1 \cdot 1 = 3x^2 - 1 \\
 P_2(x) &= \frac{3x^2 - 1}{2}
 \end{aligned}$$

Como se puede ver, según la Ecn. (3.11), para evaluar la integral, se necesita los nodos y después los pesos. Aquí, los nodos son las raíces del polinomio que obtendremos a través el método de Newton-Raphson. Para este fin, se emplean las derivadas $P'_n(x)$, que se calculan por

$$P'_n(x) = \frac{n}{x^2 - 1} [xP_n(x) - P_{n-1}(x)] \quad (3.13)$$

Según el método de Newton-Raphson, la i -ésima raíz se evalúa de manera iterativa

$$\xi_i^{k+1} = \xi_i^k + \frac{P_n(\xi_i^k)}{P'_n(\xi_i^k)}$$

donde empezamos con la primera prueba

$$\xi_i^0 = \cos \left[\left(\frac{i - 1/4}{n + 1/2} \right) \pi \right]$$

para la i -ésima raíz del polinomio de grado n . Ya teniendo el valor de los nodos podemos evaluar el valor de los pesos

$$w_i = \frac{2}{(1 - \xi_i^2)[P'_n(\xi_i)]^2}$$

Para obtener la fórmula para la cuadratura tenemos que hacer un cambio de variable. Se usa el

intervalo $x \in [-1, 1]$ como estándar.

$$\begin{aligned}
 x &= \frac{b-a}{2}t + \frac{a+b}{2} \\
 \frac{dx}{dt} &= \frac{b-a}{2} \\
 dx &= \left(\frac{b-a}{2}\right)dt \\
 t = -1 \quad x &= \frac{-b+a}{2} + \frac{a+b}{2} = a \\
 t = 1 \quad x &= \frac{b+a}{2} + \frac{a+b}{2} = b \\
 t &= \frac{2x-b-a}{b-a}
 \end{aligned}$$

$$\int_a^b f(x)dx = \frac{b-a}{2} \int_{-1}^1 f\left(\frac{b-a}{2}t + \frac{a+b}{2}\right)dt$$

Esta ecuación se va a aplicar para cada valor de nodo y peso anteriormente calculados, y la aproximación de la función $f(x)$ va a ser la suma de todas esas funciones, por lo que la nueva ecuación es:

$$\int_a^b f(x)dx \approx \frac{b-a}{2} \sum_{i=1}^n w_i f\left(\frac{b-a}{2}x_i + \frac{a+b}{2}\right)dt \quad (3.14)$$

Los valores de los pesos y abcisas calculados para un polinomio de grado n son estándares, son independientes de la función, por lo que se pueden sacar de tablas de valores precalculados, aunque ya sabemos como calcularlos.

3.3. ADR 1D No Estacionaria

3.3.1. Método de Residuos Ponderados

Una vez teniendo la ecuación ADR estacionaria en una dimensión se deriva la ecuación con transformación en tiempo, por lo que se tiene que añadir el término U_t . El proceso para derivar la ecuación es igual que para la forma estacionaria. En donde se va a diferencia es en la parte de

con la aproximación de FEM. Se empieza con la forma fuerte de la ecuación

$$\frac{\partial U(x,t)}{\partial t} + v \frac{\partial U(x,t)}{\partial x} - \alpha^2 \frac{\partial^2 U(x,t)}{\partial x^2} = f(x,t) \quad (3.15)$$

con $v(x)$ siendo la velocidad, α^2 el coeficiente de difusión cuya razón da el número de Péclet. En la notación de Lagrange, la ecuación se escribe como:

$$U_t + vU_x = \alpha^2 U_{xx} + f$$

Las condiciones de frontera (BC) e iniciales (IC) del sistema son:

$$\text{BC} = \begin{cases} U(0,t) = B_0 & U(L,t) = B_L \\ \left. \frac{\partial U(x,t)}{\partial x} \right|_{x=0,L} = g(x,t) \end{cases} \quad (3.16)$$

$$\text{IC} = \quad U(x,0) = h(x,t) \quad (3.17)$$

La Ecn. (3.16) establece las condiciones de Dirichlet y de Neumann que regulan el comportamiento de la concentración de la materia $U(x,t)$ en las fronteras. La Ecn. (3.17) establece la condición inicial que puede ser una función Gaussiana u otra. Las ecuaciones anteriores constituyen la forma fuerte de la ecuación ADR 1D. La función de residuo $R(x)$ en este caso es:

$$R(x) = \frac{\partial U}{\partial t} + v \frac{\partial U}{\partial x} - \alpha^2 \frac{\partial^2 U}{\partial x^2} - f$$

Se va a multiplicar por la función de prueba $w(x)$ e integrar en $\{0, L\}$

$$\int_0^L \left[\frac{\partial U}{\partial t} w(x) + v w(x) \frac{\partial U}{\partial x} - \alpha^2 w(x) \frac{\partial^2 U}{\partial x^2} \right] dx = \int_0^L w(x) f(x) dx \quad (3.18)$$

La integración por partes de la parte de difusión es la misma que la Ecn. (3.3) Se sustituye esta expresión en la Ecn. (3.18) y se obtiene

$$\int_0^L \left[\frac{\partial U}{\partial t} w(x) + v w(x) \frac{\partial U}{\partial x} + \alpha^2 \frac{\partial w(x)}{\partial x} \frac{\partial U}{\partial x} \right] dx = \int_0^L w(x) f(x) dx \quad (3.19)$$

la cual es la forma débil.

3.3.2. Funciones Base

La aproximación de $U(x, t)$ es la misma a la Ecn. (3.5)

$$\frac{\partial U(x, t)}{\partial x} = \sum_{i=1}^N c_i(t) \varphi_i(x) \quad (3.20)$$

esta ecuación para las derivadas de U con respecto a x . Para las derivadas de U con respecto a t la función que se usará es:

$$\frac{\partial U(x, t)}{\partial t} = \sum_{i=1}^N c'_i(t) \varphi_i(x) \quad (3.21)$$

Al igual que las funciones por tramos serán las mismas que la Ecn. (3.6), funciones de triángulo. Se sustituyen las aproximaciones de $U(x, t)$, Ecn. (3.20) y Ecn. 3.21, en la forma débil (Ecn. (3.19)) para obtener

$$\int_0^L \left[\sum_{i=1}^N c'_i \varphi_i(x) w(x) + v \left(\sum_{i=1}^N c_i \frac{\partial \varphi_i(x)}{\partial x} w(x) \right) + \alpha^2 \frac{\partial w(x)}{\partial x} \left(\sum_{i=1}^N c_i \frac{\partial \varphi_i(x)}{\partial x} \right) \right] dx = \int_0^L w(x) f(x) dx \quad (3.22)$$

3.3.3. Aproximación de Galerkin/Ritz

Al igual que para la ecuación ADR 1D estacionaria se va a usar la misma función para la función de prueba $w(x) = \varphi_j(x)$. Sustituyendo esa aproximación en la Ecn. (3.22)

$$\int_0^L \sum_{i=1}^N c'_i \varphi_i(x) \varphi_j(x) dx + \int_0^L \left(v \sum_{i=1}^N c_i \frac{\partial \varphi_i(x)}{\partial x} \varphi_j(x) + \alpha^2 \frac{\partial \varphi_j(x)}{\partial x} \sum_{i=1}^N c_i \frac{\partial \varphi_i(x)}{\partial x} \right) dx = \int_0^L f(x) \varphi_j(x) dx$$

$$\sum_{i=1}^N c'_i \int_0^L \varphi_i(x) \varphi_j(x) dx + \sum_{i=1}^N \left[\int_0^L \left(v \frac{\partial \varphi_i(x)}{\partial dx} \varphi_j(x) + \alpha^2 \frac{\partial \varphi_j(x)}{\partial x} \frac{\partial \varphi_i(x)}{\partial x} \right) dx \right] c_i = \int_0^L f(x) \varphi_j(x) dx \quad (3.23)$$

Para $\varphi_j(x)$ $j = 1, \dots, N$ se construye un sistema de ecuaciones diferenciales lineales:

$$\begin{aligned} \sum_{i=1}^N c'_i \int_0^L \varphi_i \varphi_1 dx + \sum_{i=1}^N c_i \left[\int_0^L \left(v \varphi'_i \varphi_1 + \alpha^2 \varphi'_i \varphi'_1 \right) dx \right] &= \int_0^L f \varphi_1 dx \\ \sum_{i=1}^N c'_i \int_0^L \varphi_i \varphi_2 dx + \sum_{i=1}^N c_i \left[\int_0^L \left(v \varphi'_i \varphi_2 + \alpha^2 \varphi'_i \varphi'_2 \right) dx \right] &= \int_0^L f \varphi_2 dx \\ &\vdots \\ \sum_{i=1}^N c'_i \int_0^L \varphi_i \varphi_N dx + \sum_{i=1}^N c_i \left[\int_0^L \left(v \varphi'_i \varphi_N + \alpha^2 \varphi'_i \varphi'_N \right) dx \right] &= \int_0^L f \varphi_N dx \end{aligned}$$

Este sistema de ecuaciones diferenciales lineales se escribe en la forma matricial:

$$\mathbb{M} \cdot c'(t) + \mathbb{K} \cdot c(t) = \mathbf{f} \quad (3.24)$$

donde \mathbb{M} , \mathbb{K} y \mathbf{f} se nombran como matriz de masa, de rigidez y el vector de carga respectivamente.

3.3.4. Métodos Numéricos para EDO

El método popular de Runge-Kutta (RK4) es un buen candidato para resolver la Ecn. (3.24) debido a su estabilidad, combinándolo con la simplicidad de la implementación y el bajo costo computacional. Para el sistema que se tiene, desde el vector $c(t_n)$ se evalúa el vector $c(t_{n+1})$ según [23]:

$$\mathbf{c}(t_{n+1}) = \mathbf{c}(t_n) + \frac{\Delta t}{6} (\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \quad (3.25)$$

con las aproximaciones de pendiente:

$$\mathbf{k}_1 = \mathbb{M}^{-1}[-\mathbb{K} \cdot \mathbf{c}(\mathbf{t}_n) + \mathbf{f}(t_n)] \quad (3.26)$$

$$\mathbf{k}_2 = \mathbb{M}^{-1}\left[-\mathbb{K} \cdot \left(\mathbf{c}(\mathbf{t}_n) + \frac{\Delta t}{2} \mathbf{k}_1\right) + \mathbf{f}\left(t_n + \frac{\Delta t}{2}\right)\right] \quad (3.27)$$

$$\mathbf{k}_3 = \mathbb{M}^{-1}\left[-\mathbb{K} \cdot \left(\mathbf{c}(\mathbf{t}_n) + \frac{\Delta t}{2} \mathbf{k}_2\right) + \mathbf{f}\left(t_n + \frac{\Delta t}{2}\right)\right] \quad (3.28)$$

$$\mathbf{k}_4 = \mathbb{M}^{-1}\left[-\mathbb{K} \cdot \left(\mathbf{c}(\mathbf{t}_n) + \Delta t \mathbf{k}_3\right) + \mathbf{f}(t_n + \Delta t)\right] \quad (3.29)$$

3.4. Implementación del código

Una vez establecida la base matemática se implementó el código en el lenguaje Fortran 90.

El código consiste en:

- 1 módulo
- 1 programa principal
- 8 subrutinas
- 12 funciones

Los códigos se anexan en el apéndice.

3.5. Pruebas de estabilidad

Una vez hecha la implementación del código lo siguiente consistió en hacer pruebas de estabilidad para después hacer la aplicación al estado de Morelos. Se realizaron varias pruebas de estabilidad, variando el número de nodos, los valores de velocidad v , coeficiente de difusión α , Δx y Δt .

3.5.1. Número de Péclet

El número de Péclet es un número adimensional el cual mide el efecto de la advección con respecto de la difusión, es decir, la proporción de las contribuciones de la parte de advección y de la difusión [24]. Se define como

$$Pe = \frac{N_{adv}}{N_{dif}} = \frac{vL}{D} \quad (3.30)$$

donde v es la magnitud de la velocidad, L es una escala de longitud y D es el coeficiente de difusión característico. Un sistema con un número de Péclet grande tiene un efecto de difusión despreciable y su movimiento consiste principalmente por el efecto de advección. Mientras que un sistema con un número de Péclet pequeño tienen un efecto de difusión mayor, y la distribución se extiende rápidamente por el proceso de difusión [25].

3.5.2. Número de von Neumann

El análisis de estabilidad de von Neumann esta basado en el análisis de Fourier. Es la relación entre el valor de Δt y Δx con respecto al número de total de pasos.

3.6. Triangulación de Delaunay

3.6.1. Implementación

La implementación consiste en el manejo de la lista de triángulos a través de sus vértices, no de las coordenadas de cada punto. Esta lista se va actualizando para cada punto i , ya que es ahí en donde se agregan los nuevos triángulos resultantes de cada paso. Al final de cada paso, la lista debe de tener los triángulos que no se tocaron, es decir, los triángulos en los cuales el punto i no estuvo adentro del circuncírculo, y los nuevos triángulos. Los triángulos en los cuales el punto i sí estuvo adentro del circuncírculo se deben eliminar de la lista, ya que ya no existirán y serán reemplazados por los nuevos.

El código, implementado en el lenguaje Fortran 90, consiste en:

- 1 módulo
- 1 programa principal
- 8 subrutinas
- Número ilimitado de puntos (dependiendo la memoria de la computadora)

3.6.2. Ordenamiento de los datos

El hecho de que el primer paso del algoritmo sea ordenar los puntos en orden lexicográfico permitirá hacer el proceso más simple y mantener un orden con respecto a la malla, ya que esto consiste de que el punto (x_i, y_i) siempre será menor al punto (x_j, y_j) . El ordenamiento de los datos se lleva a cabo con el método burbuja.

```
1 do i = 1, puntos-1
2   do j = i+1, puntos
3     if (ordCoord(i,1) > ordCoord(j,1)) then
4       temp = ordCoord(i,1)
5       ordCoord(i,1) = ordCoord(j,1)
6       ordCoord(j,1) = temp
7       temp = ordCoord(i,2)
8       ordCoord(i,2) = ordCoord(j,2)
9       ordCoord(j,2) = temp
10    end if
11    if (ordCoord(i,1) == ordCoord(j,1)) then
12      if (ordCoord(i,2) > ordCoord(j,2)) then
13        temp = ordCoord(i,2)
14        ordCoord(i,2) = ordCoord(j,2)
15        ordCoord(j,2) = temp
16      end if
17    end if
18  end do
19 end do
```

3.6.3. Supertriángulo

El primer paso para formar el supertriángulo, es hacer un círculo que incluya todos los puntos. Para esto se calcula el centro de los puntos para después medir el radio del centro al punto más lejano en el eje y . Una vez hecho el círculo, calculamos el ápice del supertriángulo equilatero

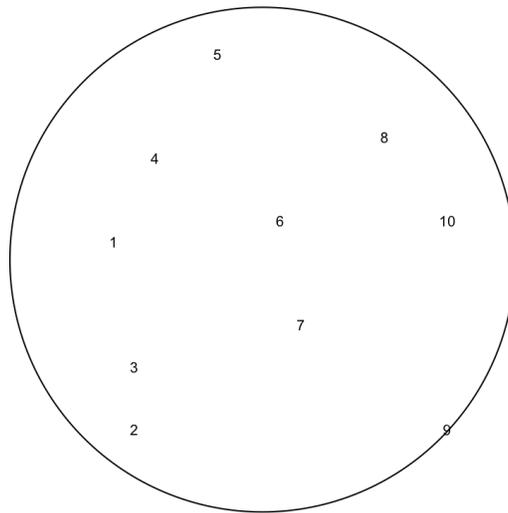


Fig. 3.3: Círculo que contiene todos los puntos.

dándole una distancia de dos veces el radio del círculo a partir del centro de éste. Los otros dos vértices se determinan por:

$$v_x = -(2radio)cos(angulo) + centro$$

$$v_y = -(2radio)sin(angulo) + centro$$

Las coordenadas de los tres vertices se añadirán al final de la lista de puntos, y se denominarán como $v_{npuntos+1}$, $v_{npuntos+2}$ y $v_{npuntos+3}$. Es decir, en la Fig. ?? hay 10 puntos, añadiendo los 3 puntos de los vértices del supertriángulo se agregarían los puntos 11, 12 y 13. Para el primer paso se va a trabajar con los vértices del supertriángulo y el primer punto para obtener los primeros triángulos. Éstos se obtienen uniendo los vértices con el puntos. Cada triángulo formado se va a agregar a una lista de triángulos, esto escribiendo los vértices de cada triángulo.

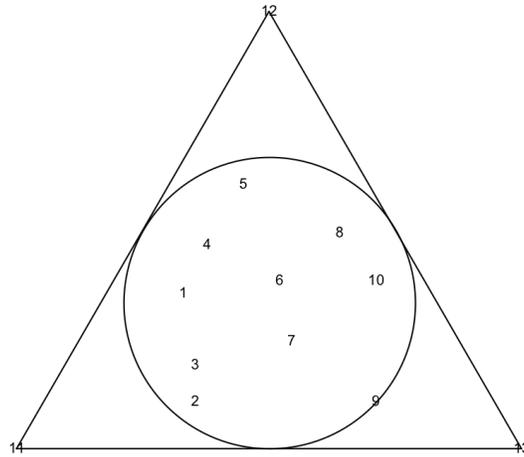


Fig. 3.4: Supertriángulo con círculo adentro.

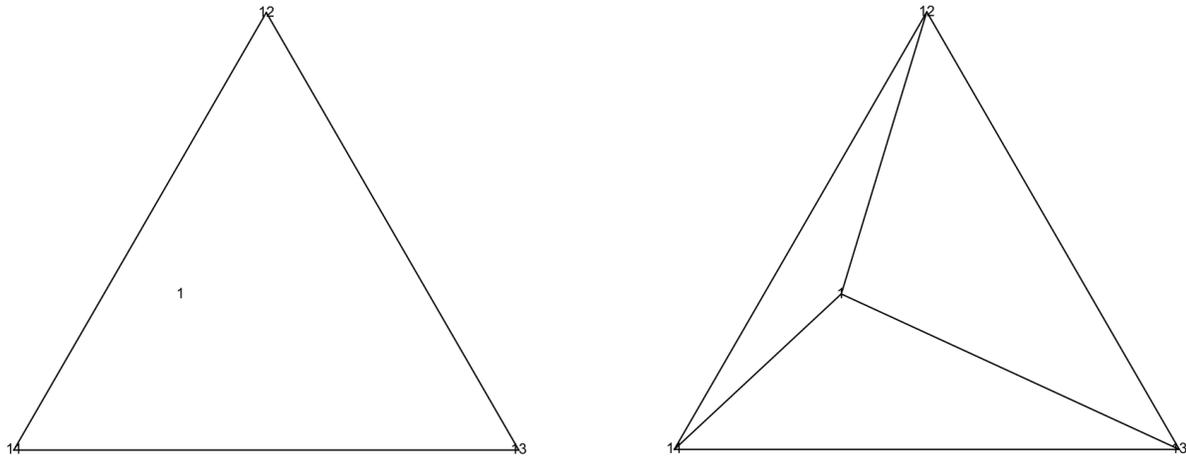


Fig. 3.5: Primer punto dentro del supertriángulo y los primeros 3 triángulos.

Lista de triángulos:

1	11	12
1	11	13
1	12	13

Después se agrega el siguiente punto.

3.6.4. Circuncírculos

Un circuncírculo es la circunferencia que debe de pasar por los tres vértices del triángulo. Lo primero que se debe de hacer es calcular el circuncírculo de cada uno de los triángulos en la malla.

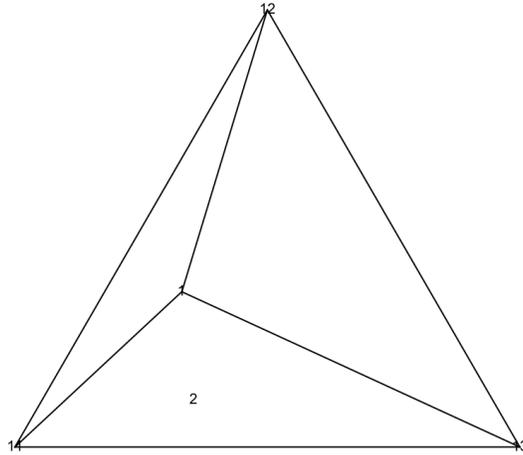


Fig. 3.6: Segundo punto dentro del supertriángulo.

Empezamos determinando el centro del circuncírculo con el siguiente algoritmo [26]

$$m = -(x_2 - x_1) / (y_2 - y_1)$$

$$mx = (x_1 + x_2) / 2$$

$$my = (y_1 + y_2) / 2$$

$$xc = (x_1 + x_3) / 2$$

$$yc = m * (xc - mx) + my$$

Después comparamos la distancia del centro del circuncírculo al punto i y el radio del circun-

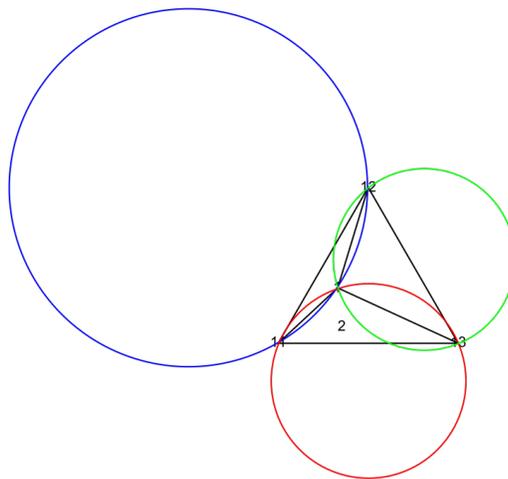


Fig. 3.7: Circuncírculos de cada triángulo.

círculo para comprobar si está adentro.

3.6.5. *Aristas compartidas*

Con la lista de triángulos se van a sacar las aristas de los triángulos en los cuales esta adentro el punto i , siempre y cuando este adentro de 2 o más triángulos ya que si sólo esta adentro de 1 será un caso diferente. Los vértices se van a acomodar de tal manera que el primer número sea el menor, es decir, si la arista es (v_1, v_2) entonces $v_1 < v_2$, ya que esto ayudará a determinar las que estén repetidas. Por ejemplo, para el triángulo (v_1, v_2, v_3) sus aristas son (v_1, v_2) , (v_1, v_3) y (v_2, v_3) . Las aristas que aparezcan más de una vez en la lista son las aristas compartidas entre triángulos vecinos, por lo tanto se eliminarán de la lista. La comparación de dos aristas se lleva a cabo por el siguiente

```
1 if (arista(i,1) == arista(j,1) .and. (arista(i,2) == arista(j,2))
```

3.6.6. *Añadir triángulos nuevos*

Las aristas restantes, las que no se repitieron, serán las que formen los nuevos triángulos con el punto i , los cuales se agregarán a la lista de triángulos que el punto i no estuvo afuera. El caso en el que el punto i solo este adentro de un triángulo no va a haber aristas repetidas, por lo que automáticamente se puede sacar la lista de aristas y hacer los nuevos triángulos (i, v_1, v_2) , (i, v_1, v_3) y (i, v_2, v_3) .

En la Fig. ?? el punto 2 únicamente esta dentro de un circuncírculo del triángulo 1 11 13. La lista de aristas será

	1	11
Lista de aristas:	1	13
	11	13

Cada una de estas aristas va a formar un triángulo nuevo uniéndolas con el punto i , en este caso el punto 2. Los nuevos triángulos se van a agregar a la lista de triángulos ya existente, solamente se

eliminará el(los) triángulo(s) en el(los) que estuvo adentro el punto i . La nueva lista de triángulos va a ser:

	1	11	12
		1	12
Lista de triángulos:	2	1	11
	2	1	13
	2	11	13

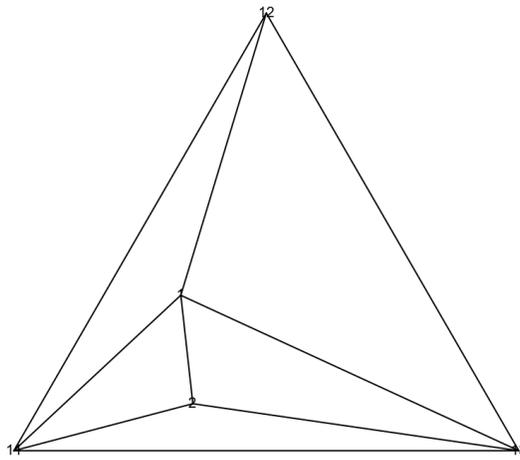


Fig. 3.8: Circuncírculos de cada triángulo.

Se agrega el siguiente punto. El punto 3 esta dentro de 2 triángulos, 2 1 11 y 2 1 13. La lista de aristas, acomodadas con el punto menor antes $(v_1, v_2) \quad v_1 < v_2$, va a tener aristas repetidas, por lo que se deben de eliminar y solamente dejar las no repetidas.

Aristas repetidas	Aristas no repetidas
1 2	2 11
2 11	1 11
1 11	2 13
1 2	1 13
2 13	
1 13	

Y al igual que con los otros puntos se agregan a la lista los nuevos triángulos.

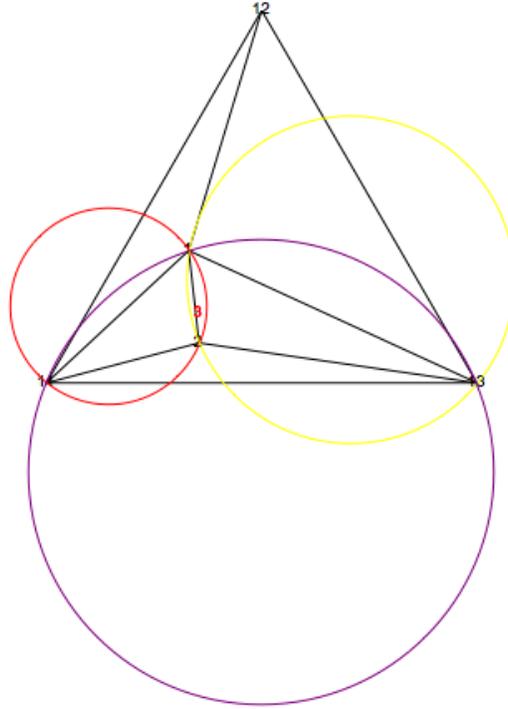


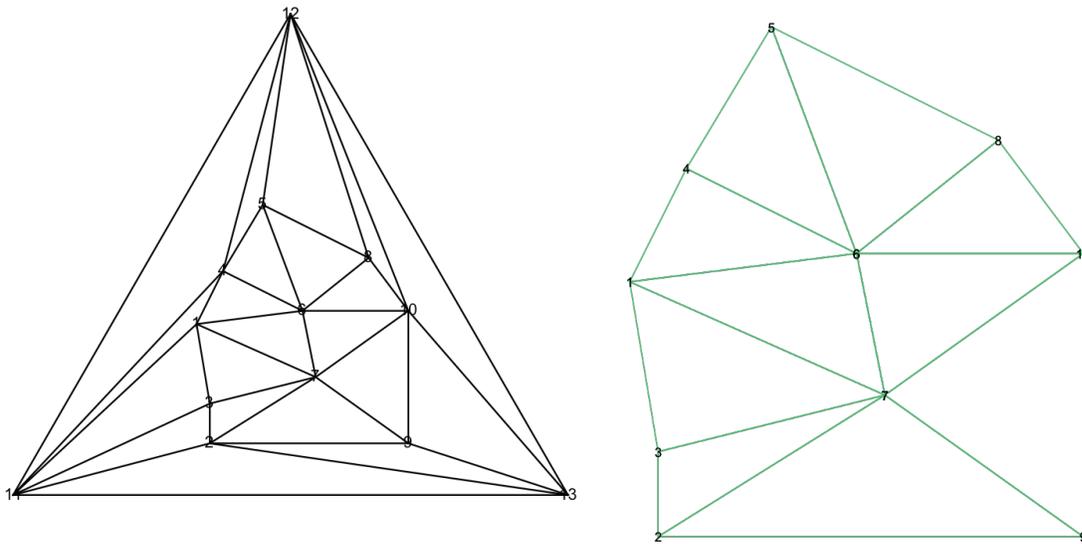
Fig. 3.9: Se agrega el punto 3 y se calculan los circuncírculos.

	1	11	12
	1	12	13
	2	11	13
Lista de triángulos:	3	2	11
	3	1	11
	3	2	13
	3	1	13

3.6.7. Quitar supertriángulo

Cuando se terminan de analizar todos los puntos i y se tienen todos los triángulos se deben de quitar los triángulos que tengan uno o más vértices del supertriángulo, $v_{\text{npuntos}+1}$, $v_{\text{npuntos}+2}$ y

$v_{\text{npuntos}+3}$.



3.7. Aplicación al estado de Morelos

Una vez hecha la implementación del código y las pruebas de estabilidad se siguió con la aplicación de Morelos. La aplicación consistió en diferentes pasos: obtener datos de las estaciones meteorológicas, hacer el perfil de viento, construir las trayectorias, para finalmente obtener el input para el código.

3.7.1. Perfil de viento

El estado de Morelos cuenta con una red de 32 estaciones meteorológicas. En el proyecto de licenciatura se obtuvieron datos de las estaciones más cercanas al punto de muestreo y para el periodo de la toma de muestras, que fueron los meses de octubre y noviembre de 2014. Sólo 3 estaciones fueron las que contaban con datos en ese periodo de tiempo. Ahora se necesitaba contar con más estaciones por lo que se decidió mantener los mismos meses y buscar el año que tuviera el mayor número de estaciones con datos. Se obtuvieron datos de velocidad y dirección de viento para 27 estaciones de los meses de octubre y noviembre de 2010 [27].

3. Metodología

Estaciones		Estaciones	
1	CEIEPO	14	INIFAP
2	Tetela del Monte	15	Puente de Ixtla
3	Tlalnepantla	16	Puente de Ixtla 2
4	Tlayacapan	17	Amacuzac
5	Tepoztlán	18	Jojutla
6	Cocoyoc	19	Tlaltizapán (El Charco)
7	Yautepec	20	Tlalquitenango
8	Emiliano Zapata	21	Ayala
9	Cuatla	22	Ocuituco
10	Mazatepec	23	Tetela del Volcán
11	Coatetelco	24	Huazulco
12	Tlaltizapán	25	Jonacatepec
13	Moyotepec	26	Axochiapan
		27	Telpancingo

Tab. 3.1: Estaciones meteorológicas.

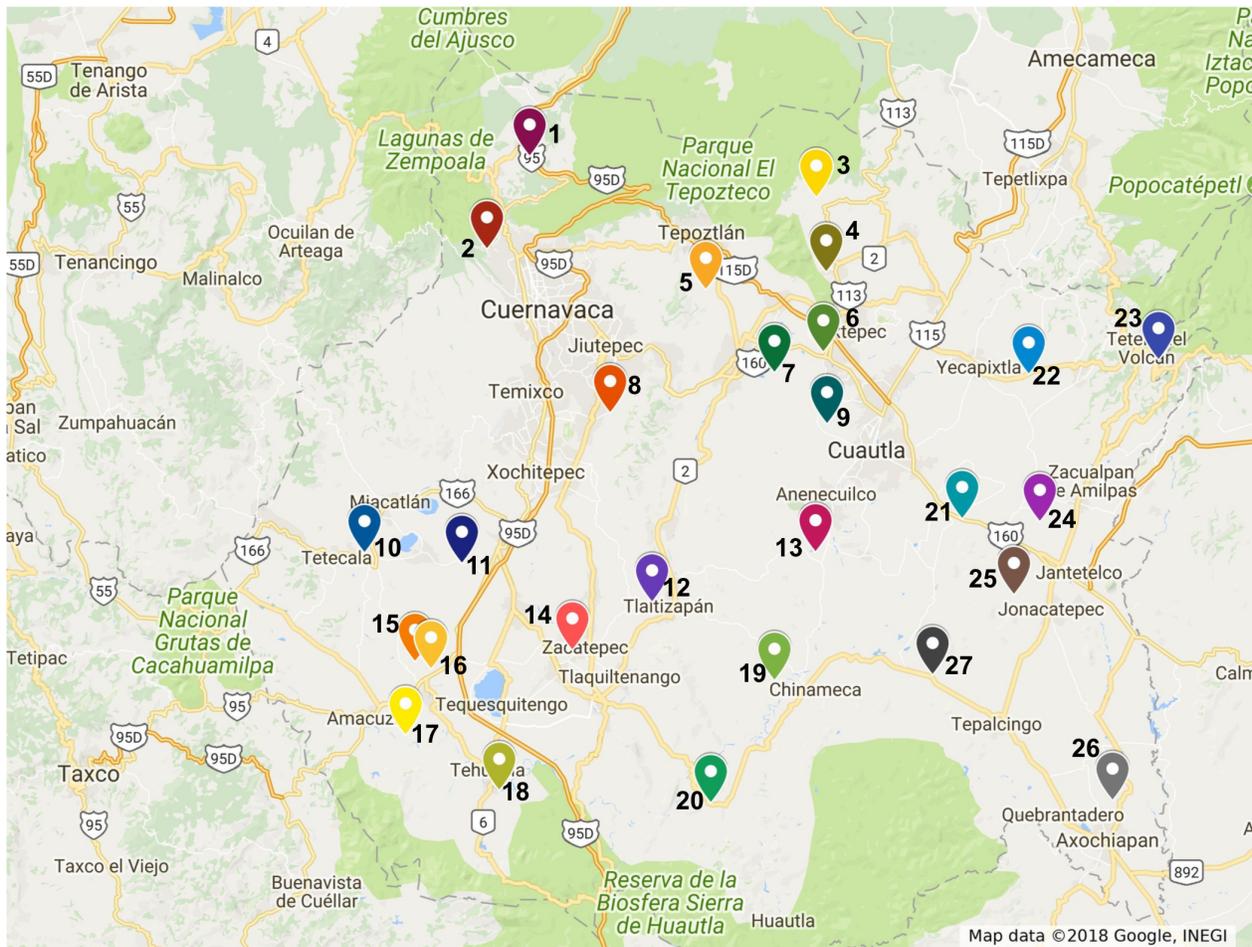


Fig. 3.10: Mapa con 27 estaciones [7].

3. Metodología

Viendo los valores se eligió un día, el 28 de octubre de 2010, para construir el perfil de viento. Lo primero fue poner los vectores en el mapa de contorno del estado [28].

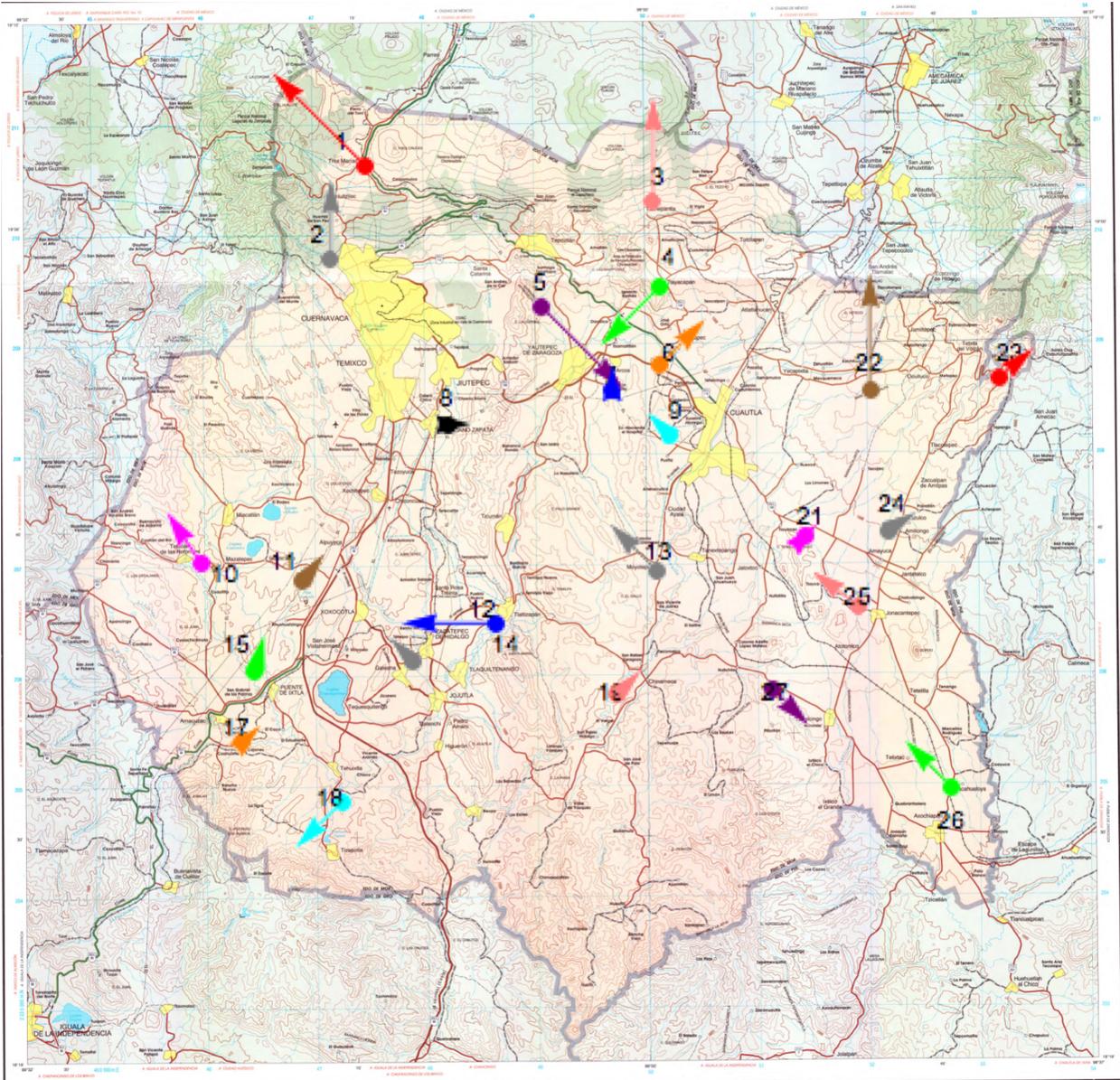


Fig. 3.11: Mapa con vectores.

3. Metodología

Estación	v	θ	dir	Estación	v	θ	dir
1 CEIEPO	8.15	336.9	NO	15 Puente de Ixtla	1.3	358.2	N
2 Tetela del Monte	1.39	357	N	16 Puente de Ixtla 2	1.13	173	S
3 Tlalnepantla	3.77	342.7	N	17 Amacuzac	0.64	62.3	NE
4 Tlayacapan	1.95	218.3	SO	18 Jojutla	0.18	226.9	SO
5 Tepoztlán	2.91	153	SE	19 Tlaltizapán (El Charco)	0.91	24.9	NE
6 Cocoyoc	1.53	50.2	NE	20 Tlalquilenango	0.	126.4	SE
7 Yautepec	0.34	345.9	N	21 Ayala	0.19	127.6	SE
8 Emiliano Zapata	0.45	109.2	E	22 Ocuituco	5.38	11.3	N
9 Cuautla	0.87	312.7	NO	23 Tetela del Volcán	1.01	27.5	NE
10 Mazatepec	0.92	359.8	N	24 Huazulco	0.45	174.2	S
11 Coatetelco	0.58	44.6	NE	25 Jonacatepec	3.6	335.7	NO
12 Tlaltizapán	6.44	250.4	O	26 Axochiapan	1.77	332.6	NO
13 Moyotepec	1.75	303.8	NO	27 Telpancingo	1.22	141.6	SE
14 INIFAP	0.72	311.8	NO				

Tab. 3.2: Valores de velocidad $v(km/h)$, ángulo θ y dirección de viento del 28 de octubre 2010.

La construcción de las trayectorias se hizo con ayuda de los vectores y del mapa de contorno, dibujando las trayectorias tomando como base las carreteras y los vectores. Lo primero que se hizo fue dibujar líneas que conectaran dos vectores cercanos (Fig. 3.12).

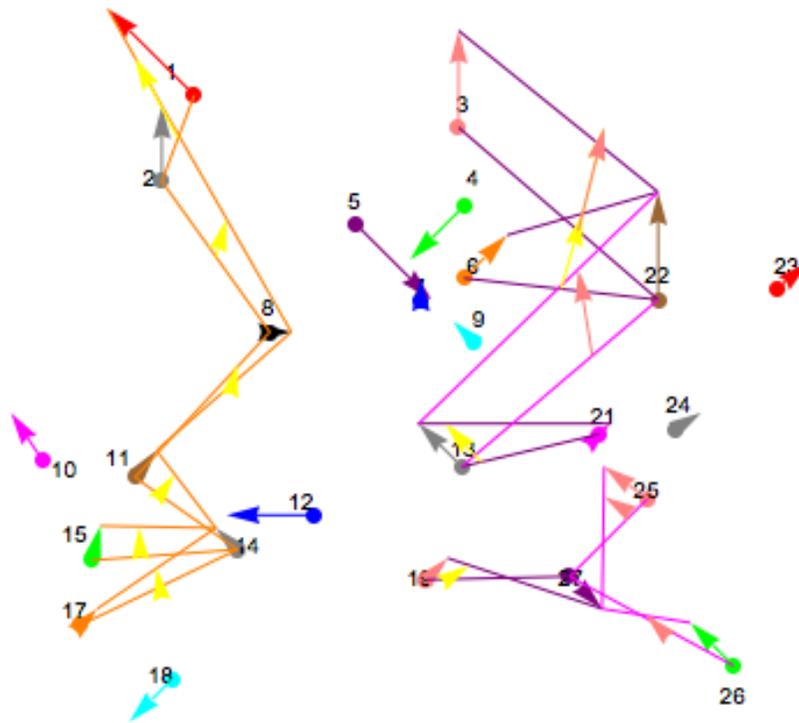


Fig. 3.12: Vectores con líneas.

Esto para poder calcular la magnitud de la velocidad y darle la dirección correcta a la trayectoria dibujada ya que para el código se necesitan los valores de velocidad y posición con respecto al eje x (ya que sólo es en una dimensión). La velocidad se interpoló y se obtuvo con una ecuación para dividir un segmento:

$$(x, y) = \left\{ \frac{mx_2 + nx_1}{m + n}, \frac{my_2 + ny_1}{m + n} \right\} \quad (3.31)$$

siendo m y n la proporción del punto P con respecto a A y B .

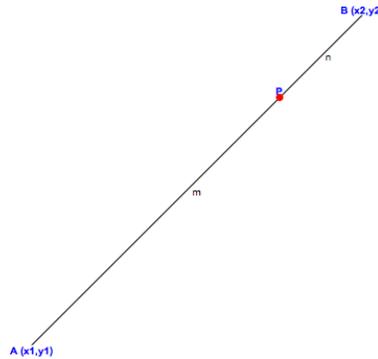


Fig. 3.13: Representación de la ecuación 3.31.

La posición en x se tomó con la escala del mapa. El eje vertical del mapa va de 0 a 40 km, y se tomó esos valores como el dominio $x \in [0, 40]$. La condición de frontera es la condición de Dirichlet, teniendo concentración 0 en las fronteras $U(0, t) = U(40, t) = 0$.

Trayectoria 1

x	v	x	v
0	0.	21	0.553
9	4.77	22	0.493
10	4.128	23	0.526
11	3.486	24	0.559
12	2.844	25	0.592
13	2.202	26	0.626
14	1.56	27	0.786
15	0.919	28	0.946
16	0.858	29	1.106
17	0.797	30	0.893
18	0.736	31	0.679
19	0.675	40	0.
20	0.614		

Tab. 3.3: Valores de posición x y velocidad $v(km/h)$ para la trayectoria 1.

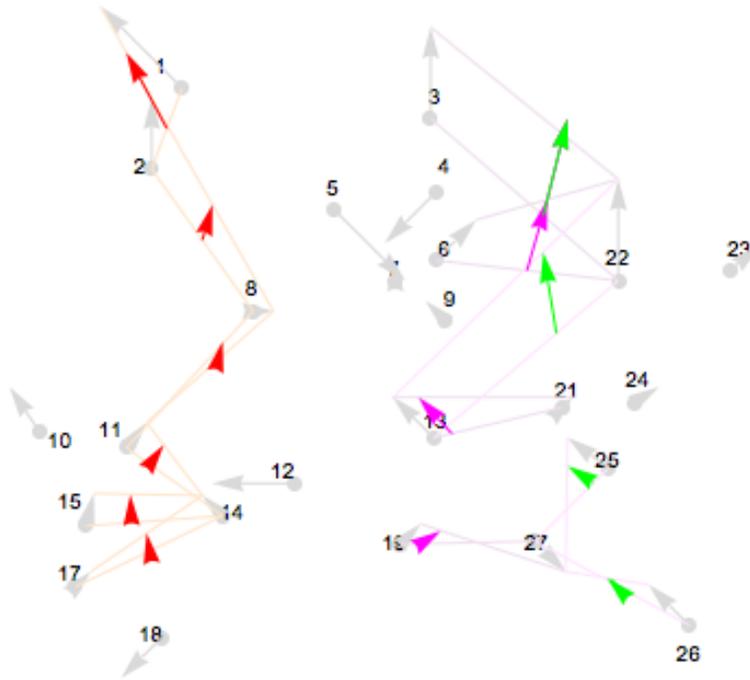


Fig. 3.14: Vectores para las trayectorias.

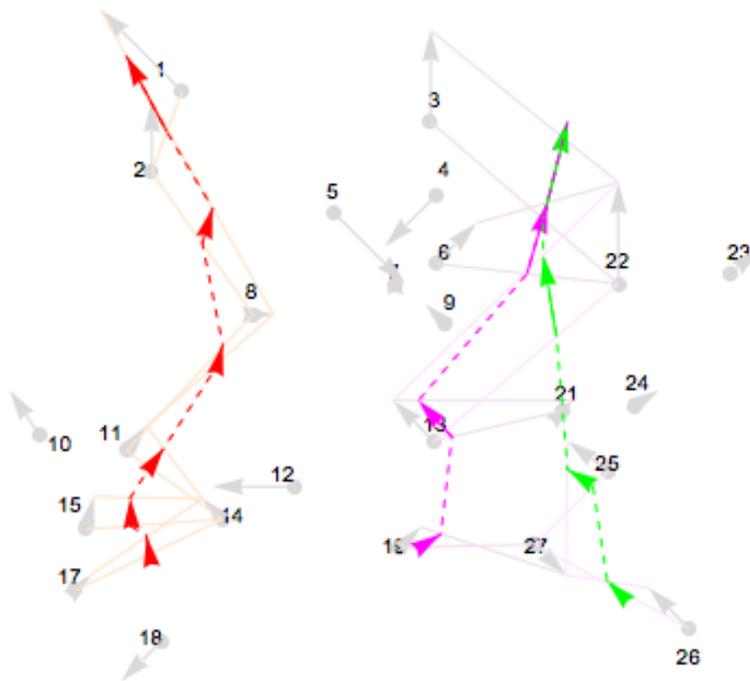


Fig. 3.15: Trayectorias.

Trayectoria 2

x	v	x	v
0	0.	22	2.075
14	4.736	23	1.799
15	4.309	24	1.527
16	3.882	25	1.412
17	3.455	26	1.297
18	3.179	27	1.182
19	2.903	28	1.067
20	2.627	29	0.954
21	2.351	40	0.

Tab. 3.4: Valores de posición x y velocidad $v(km/h)$ para la trayectoria 2.

Trayectoria 3

x	v	x	v
0	0.	25	3.384
14	4.736	26	3.253
15	4.623	27	3.124
16	4.510	28	2.891
17	4.397	29	2.658
18	4.284	30	2.425
19	4.17	31	2.192
20	4.039	32	1.959
21	3.908	33	1.726
22	3.777	34	1.495
23	3.646	40	0.
24	3.515		

Tab. 3.5: Valores de posición x y velocidad $v(km/h)$ para la trayectoria 3.

4. RESULTADOS Y DISCUSIÓN

4.1. *Resultados del estudio de estabilidad*

Los estudios que se realizaron fueron modificando los valores de nodos, velocidad v , coeficiente de difusión α , Δx y Δt . Lo que se buscaba era que los resultados numéricos, que resultaban del código, no mostraran inestabilidad numerica. Si este era el caso se modificaban los valores de las variables antes mencionadas.

El primer estudio fue el de número de Péclet. Se llevó a cabo probando con diferentes valores de velocidad (dejando la velocidad constante) y diferentes valores de α . Después esos mismos valores de α se probaron con una velocidad que va disminuyendo $v = (-0.02x + 1.1)$. Estas pruebas se realizaron para diferente números de nodos. Una de las primeras tendencias que se observaron es que conforme el número de nodos aumenta, los resultados de los estudios son mucho más representativos. Con un número pequeño de nodos no es suficiente para representar a todo el dominio. Con respecto a la velocidad, conforme menor sea habrá más tiempo para que se difunda el contaminante y viceversa, conforme mayor sea la velocidad no habrá tiempo para que se lleve a cabo la difusión, será mayor el efecto de advección que el de difusión. Por lo tanto, la tendencia del coeficiente de difusión es similar al de la velocidad. El número de nodos está relacionado con el valor de Δx . El valor de Δx corresponde a la distancia entre dos puntos (nodos) en el eje x . Entre más nodos haya, menor va a ser el valor de Δx . La Fig. 4.1 y la Fig. 4.2 es la variación de valores de Δt . La Fig. 4.2 y la Fig. 4.3 varían en el valor de α . La Fig. 4.3 y la Fig. 4.4 varían en la diferencia de velocidad, la Fig. 4.3 es con velocidad disminutiva y la Fig. 4.4 con velocidad constante.

4. Resultados y Discusión

$$\alpha = 1.5 \quad \Delta t = 0.3 \quad v = (-0.02x + 1.1)$$

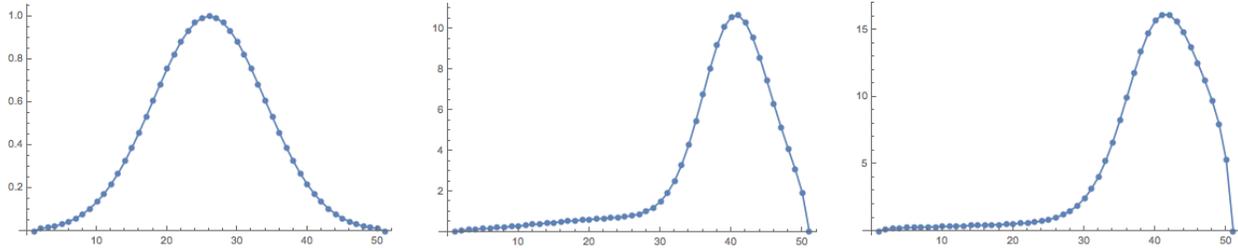


Fig. 4.1: Gráficas de los pasos 1, 50, 100.

$$\alpha = 1.5 \quad \Delta t = 0.02 \quad v = (-0.02x + 1.1)$$

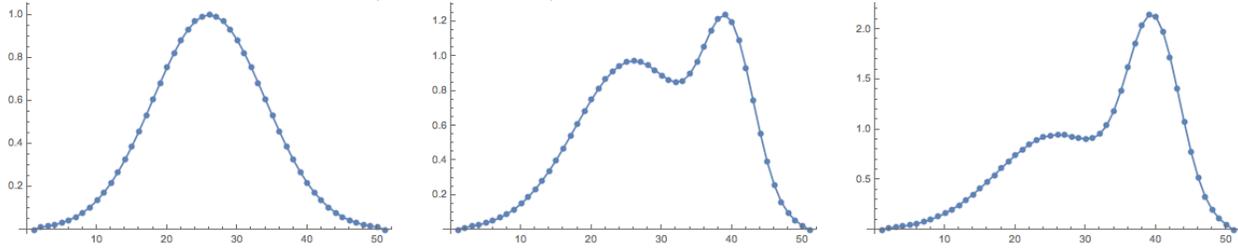


Fig. 4.2: Gráficas de los pasos 1, 50, 100.

$$\alpha = 5 \quad \Delta t = 0.02 \quad v = (-0.02x + 1.1)$$

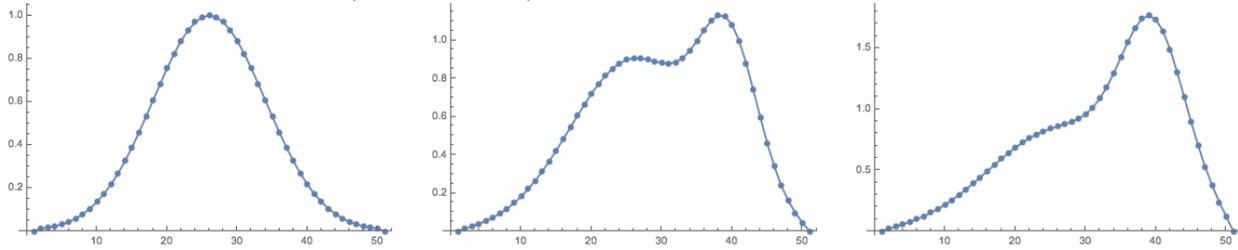


Fig. 4.3: Gráficas de los pasos 1, 50, 100.

$$\alpha = 5 \quad \Delta t = 0.02 \quad v = 5$$

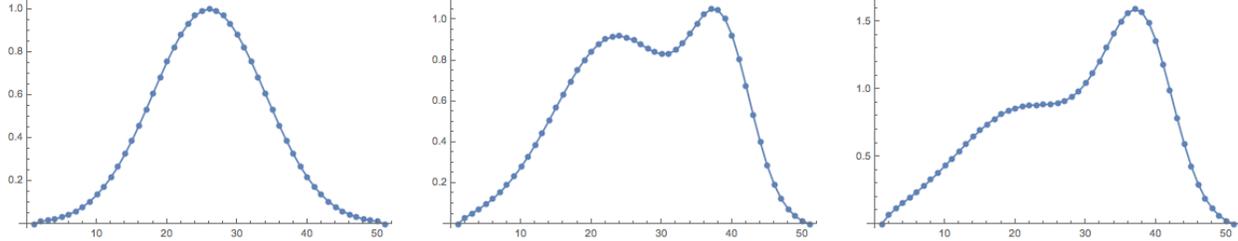


Fig. 4.4: Gráficas de los pasos 1, 50, 100.

4.2. Perfil de Viento y Trayectorias

Viendo los vectores de los datos de velocidad y dirección de viento de cada una de las estaciones en el mapa del estado, podemos observar que para ese día (28 de octubre de 2010) las trayectorias del aire van de sur a norte. La topografía del estado va incrementando la altura de sur a norte, por lo que surge la siguiente hipótesis: si las trayectorias de viento van de sur a norte y la topografía va aumentando, va a llegar un momento en el que el viento no va a pasar tomando en cuenta que llegará a toparse con una barrera, en este caso, una barrera natural de montañas. El viento podrá entrar al estado, ir de norte a sur, porque la topografía va disminuyendo, pero al contrario (de sur a norte) se necesitaría más energía para sobrepasar la barrera, es decir, las montañas. Los datos de las trayectorias se corrieron en el código, usando las posiciones en x , velocidad v y concentración inicial, una curva de Gauss. Para las tres trayectorias se usaron los valores de $\alpha = 2$ y dos valores de Δt , $\Delta t = 0.02$ y $\Delta t = 0.05$, con diferentes número de nodos; 25 nodos para la trayectoria 1, 18 nodos para la trayectoria 2 y 23 nodos para la trayectoria 3. Los resultados se muestran en forma de gráficas.

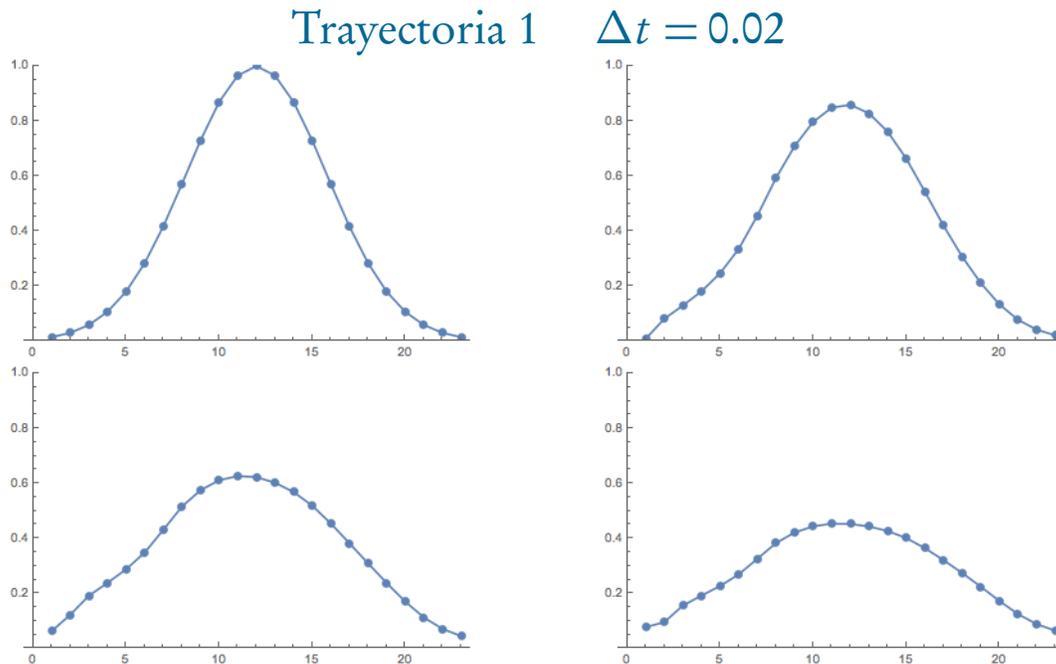


Fig. 4.5: Gráficas de concentración inicial y pasos 25, 100 y 200.

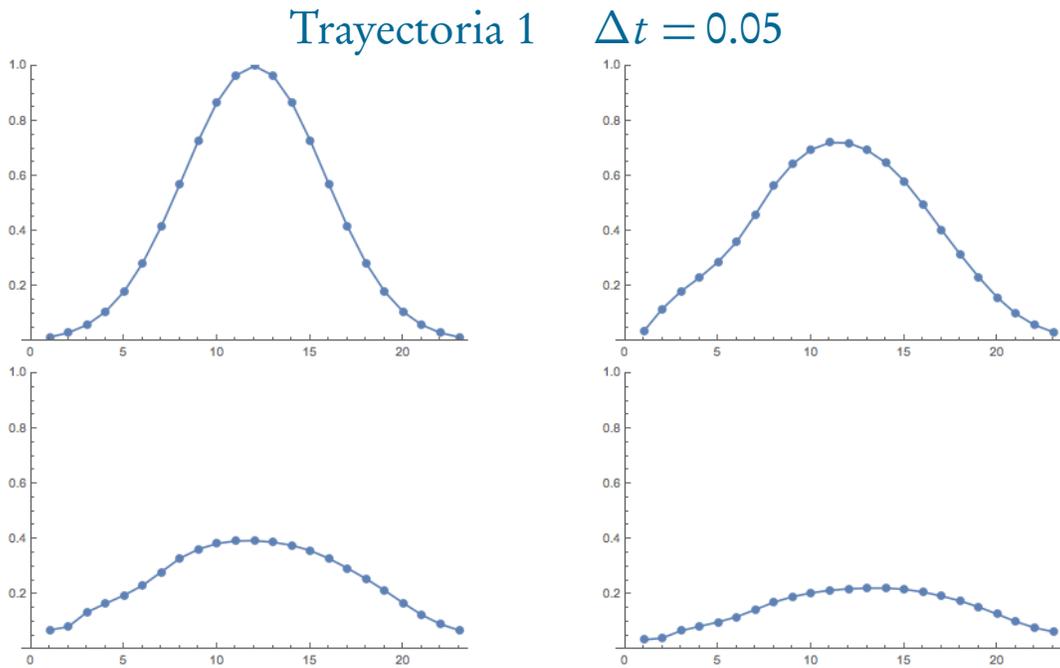


Fig. 4.6: Gráficas de concentración inicial y pasos 25, 100 y 200.

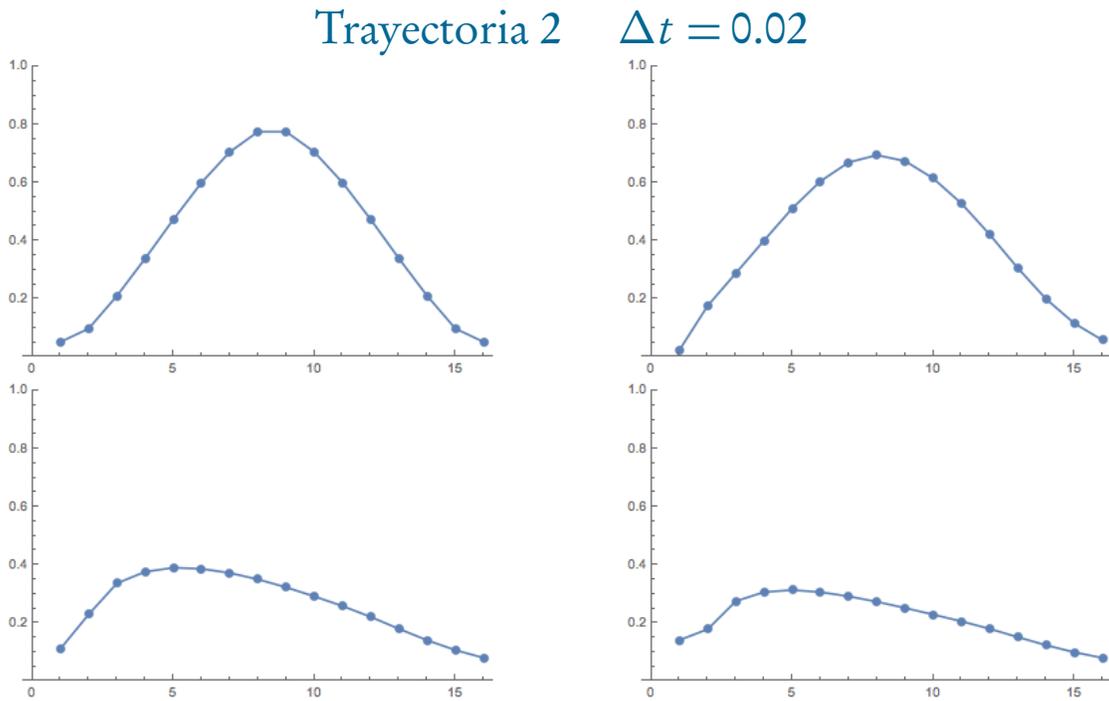


Fig. 4.7: Gráficas de concentración inicial y pasos 10, 75 y 100.

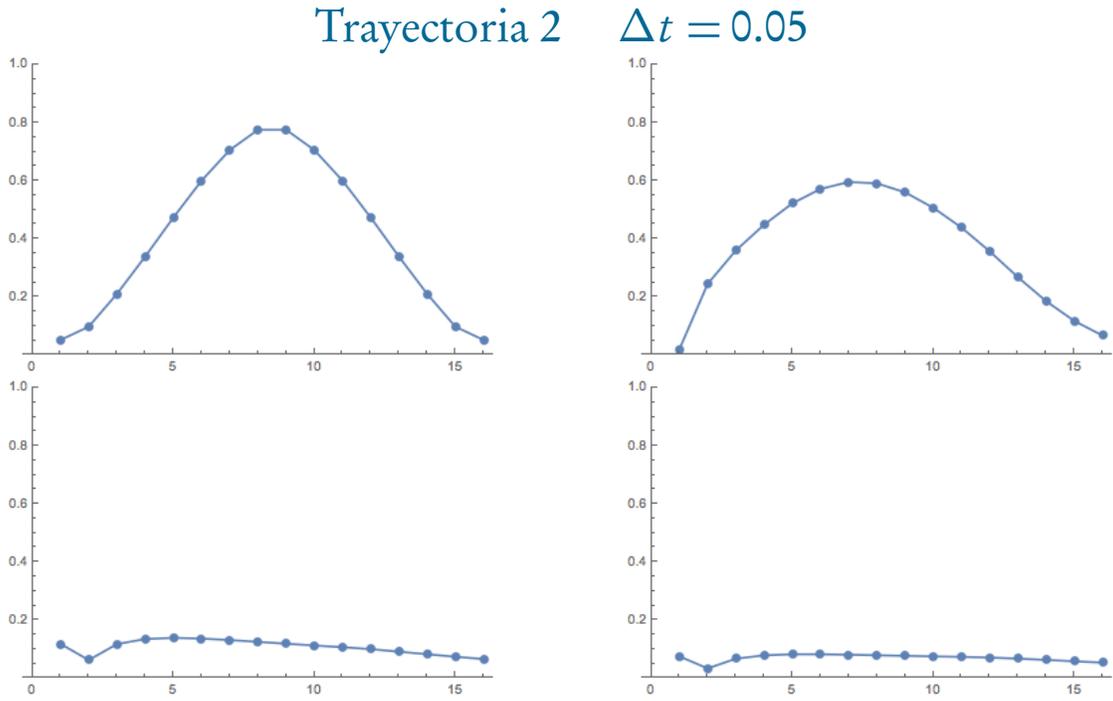


Fig. 4.8: Gráficas de concentración inicial y pasos 10, 75 y 100.

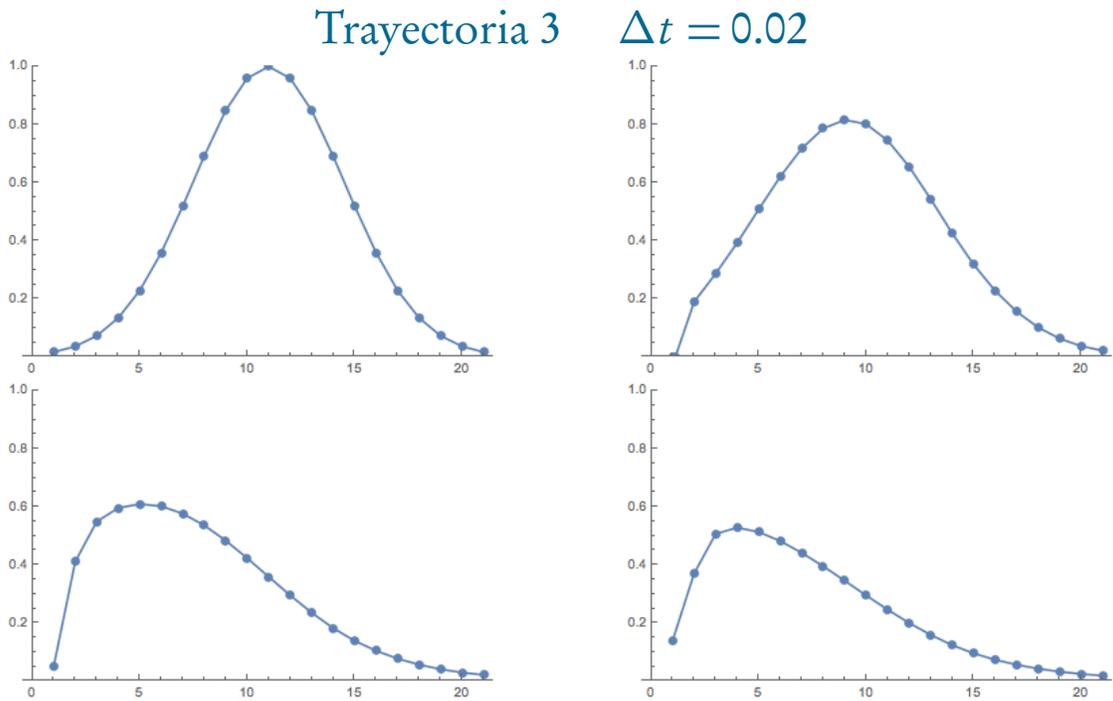


Fig. 4.9: Gráficas de concentración inicial y pasos 25, 75 y 100.

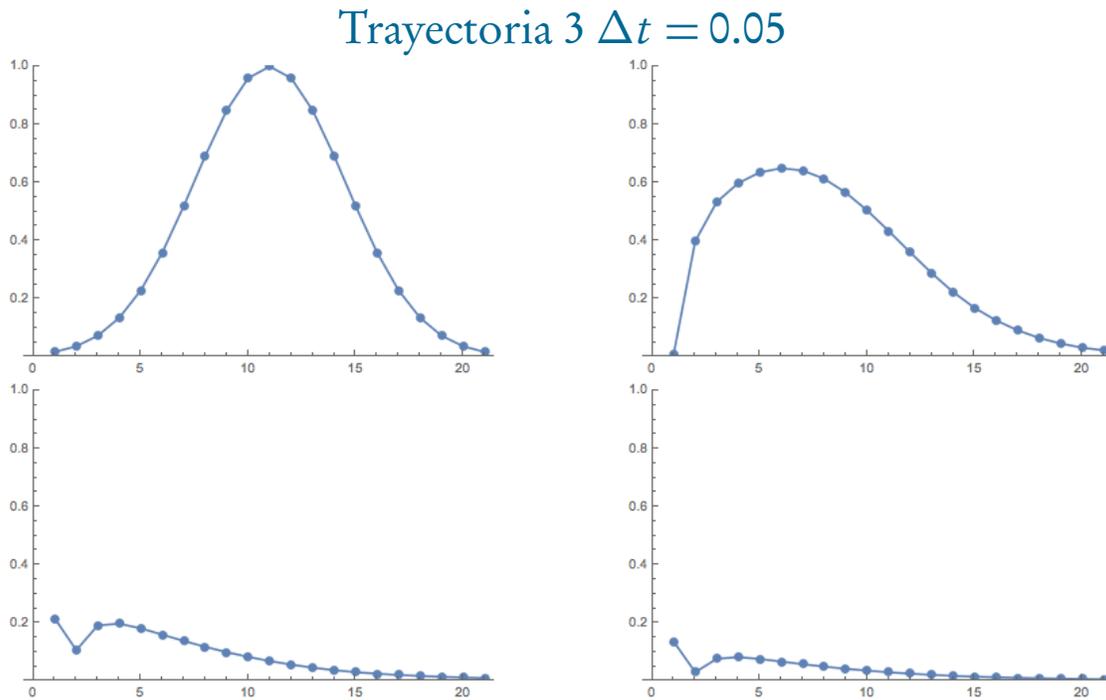


Fig. 4.10: Gráficas de concentración inicial y pasos 25, 75 y 100.

Se puede ver que la concentración va disminuyendo, esto por la parte de difusión, ya que la velocidad son valores pequeños, no pasan de 5 km/h. Esto indica que el efecto de difusión es mayor que el de advección.

Probando el código para algunos valores de coeficiente de difusión [29] que están reportados en la literatura, se observó que son valores muy pequeños, mucho más pequeños que los valores con los que se hicieron las pruebas de estabilidad. Se corrió el código para todos los gases en la tabla ???. Al ser valores muy pequeños, todos los gases, menos el H_2 , causaron inestabilidad numérica. Por lo tanto se muestran únicamente las gráficas para H_2 en las tres trayectorias, con valores de $\Delta t = 0.05$.

Coefficiente de difusión

	cm^2/s
H ₂	1.604
He	1.368
Ar	0.157
O ₂	0.192
N ₂	0.155
CO ₂	0.106
H ₂ O	0.276
CH ₄	0.188
NH ₃	0.192

Tab. 4.1: Valores de coeficientes de difusión a 273 K y 0.1 MPa.

Trayectoria 1 H₂

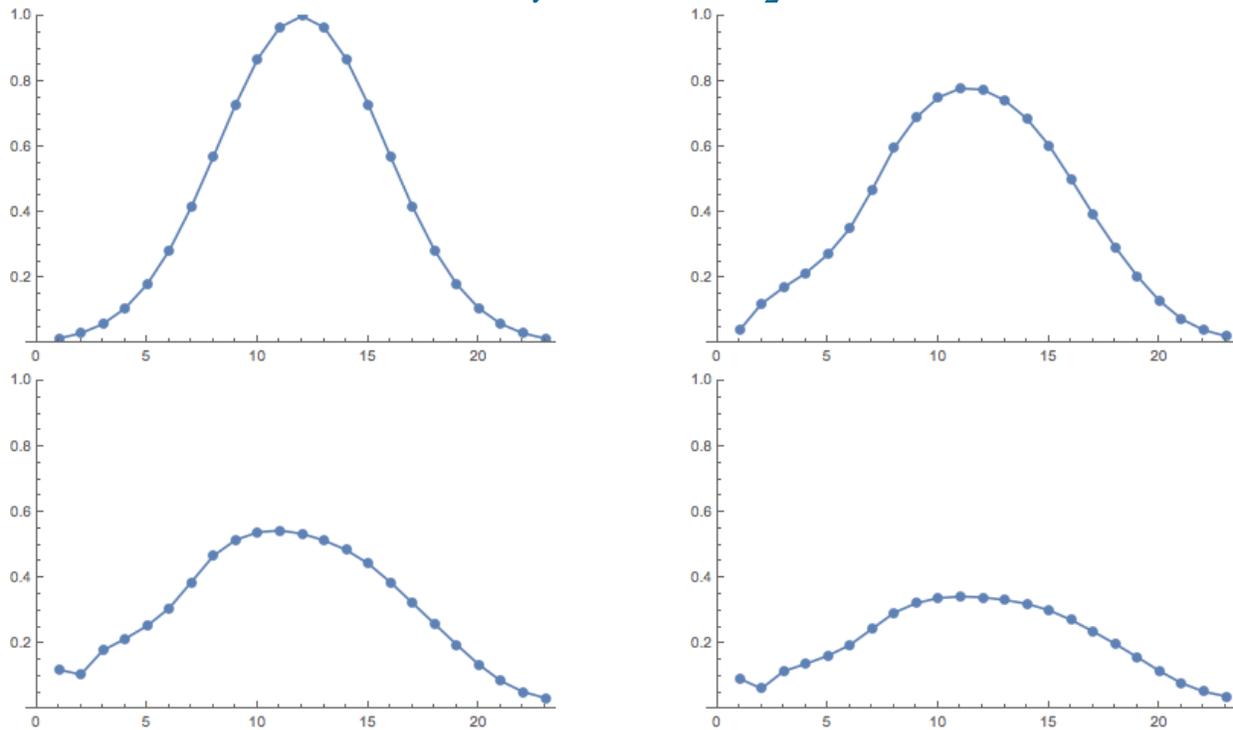


Fig. 4.11: Gráficas de concentración inicial y pasos 25, 75 y 150.

Trayectoria 2 H₂

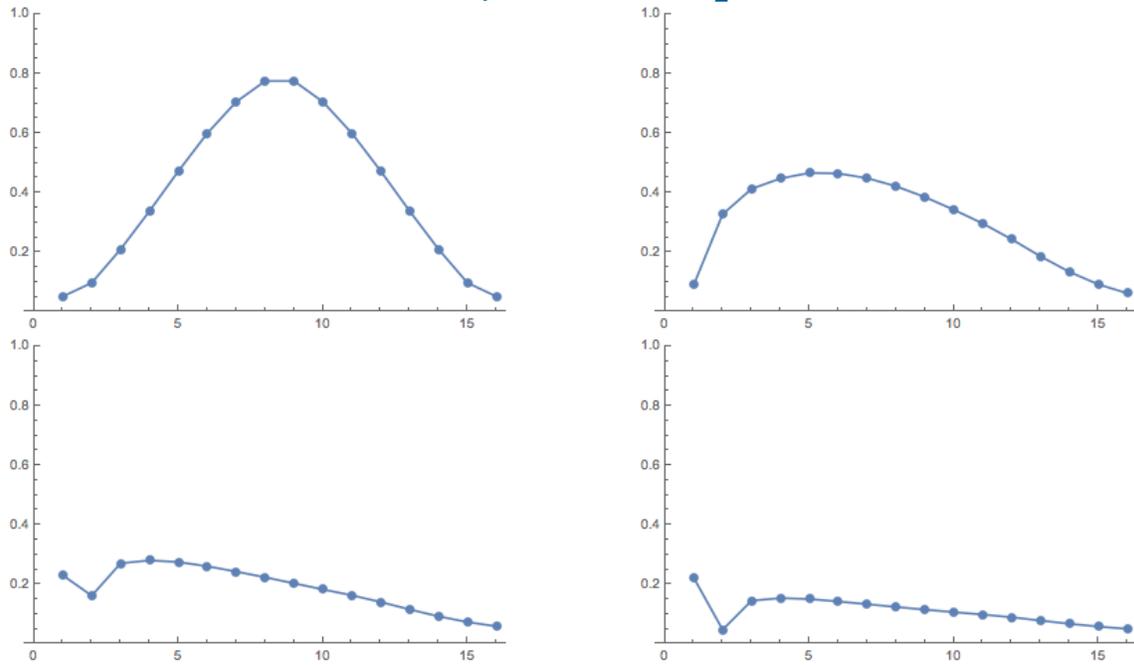


Fig. 4.12: Gráficas de concentración inicial y pasos 25, 50 y 75.

Trayectoria 3 H₂

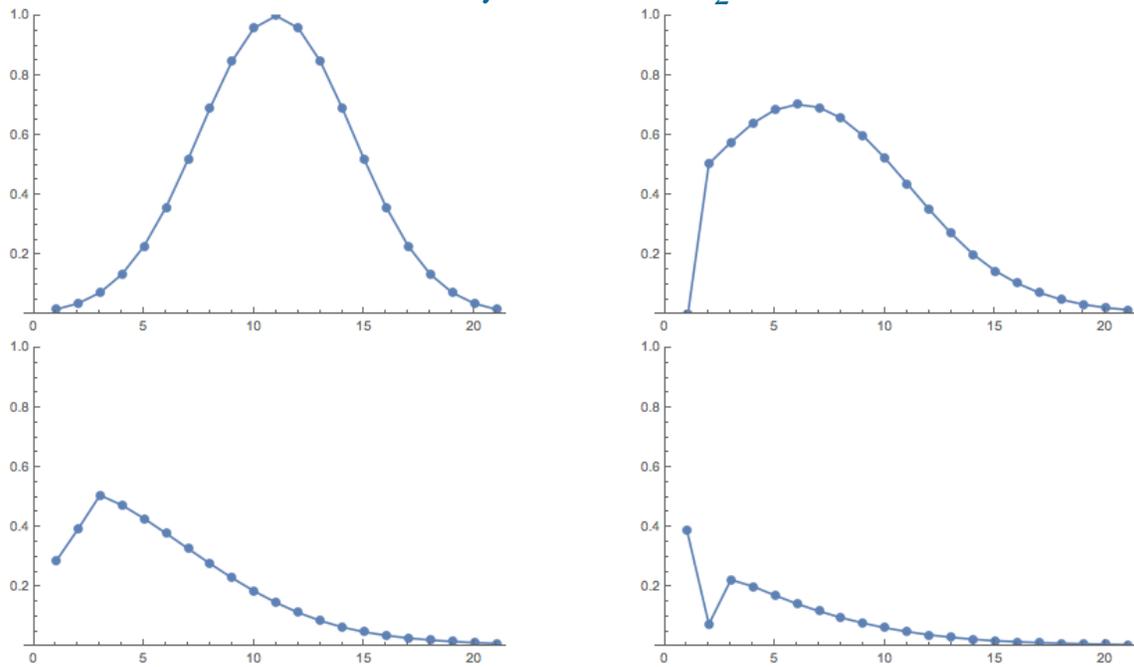


Fig. 4.13: Gráficas de concentración inicial y pasos 25, 50 y 75.

Se puede observar que son similares a las gráficas de las pruebas de trayectorias (Fig. 4.6, Fig. 4.8 y Fig. 4.10).

Por último, se hizo una prueba más. Se agregó una fuente de contaminación a lo largo de la trayectoria, por lo que se agregó una curva de Gauss simulando un suministro de contaminación. Se hizo una prueba para ver el efecto que esto tenía en la trayectoria 1 de H_2 (Fig. 4.11) con $\Delta t = 0.09$ y $peso = 0.2$ para la curva de Gauss del suministro Fig.4.15.

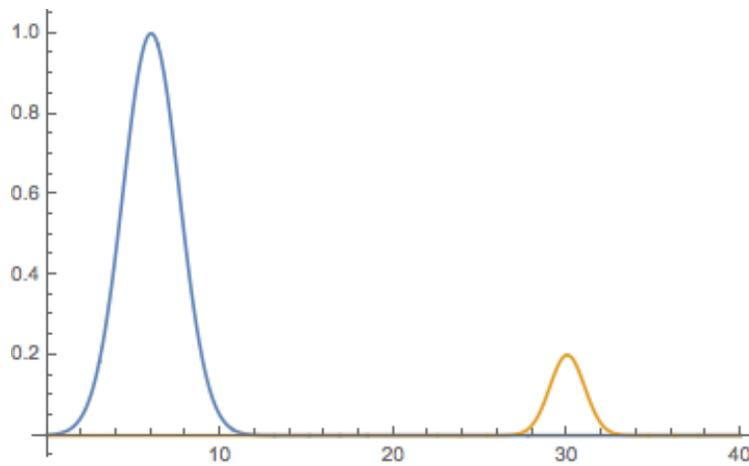


Fig. 4.14: Gráfica de la concentración inicial y un suministro.

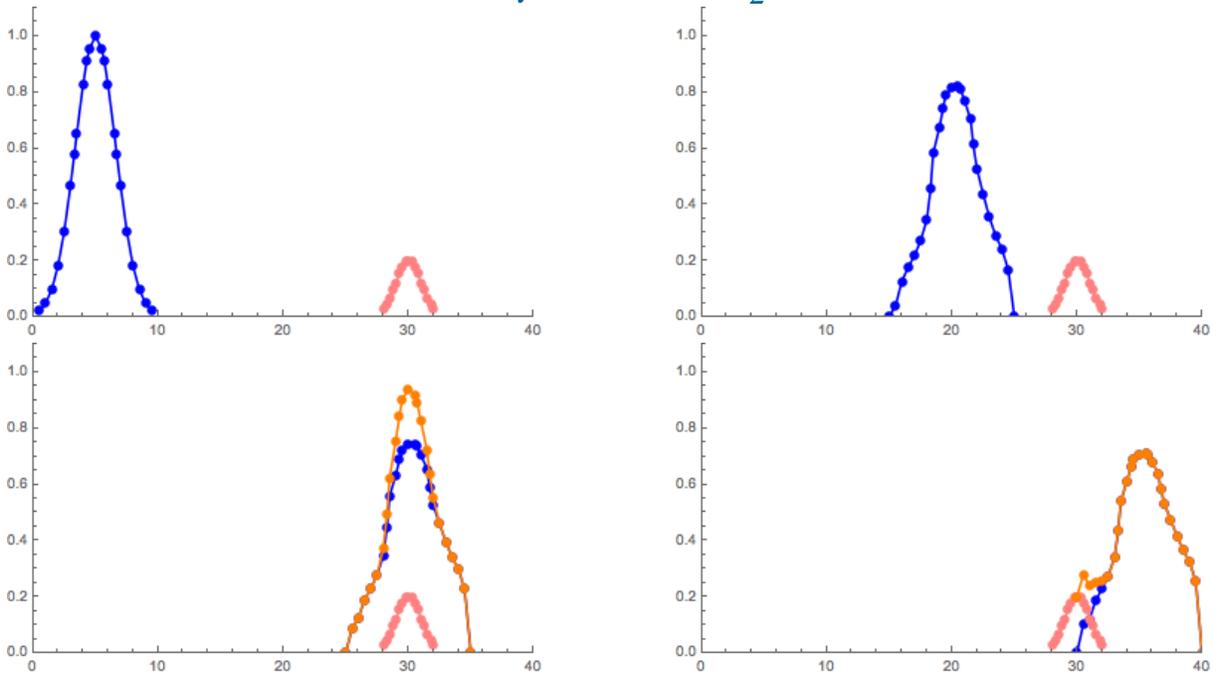
Trayectoria 1 H₂

Fig. 4.15: Gráficas de: concentración inicial y pasos 15, 25 y 30.

Esta prueba es el primer paso para que en una futuro se puede modelar los diferentes suministros y sumideros que estén a lo largo de la trayectoria. Por el momento únicamente se puso uno para ejemplificar lo que se puede hacer.

4.3. Malla de Delaunay

El código para la malla de Delaunay se probó para diferentes número de puntos y se comparó con la triangulación que viene en el programa Mathematica [30]. Las siguientes imágenes son la comparación de las mallas, las del lado izquierdo son las resultantes del código y las del lado derecho son las de Mathematica.

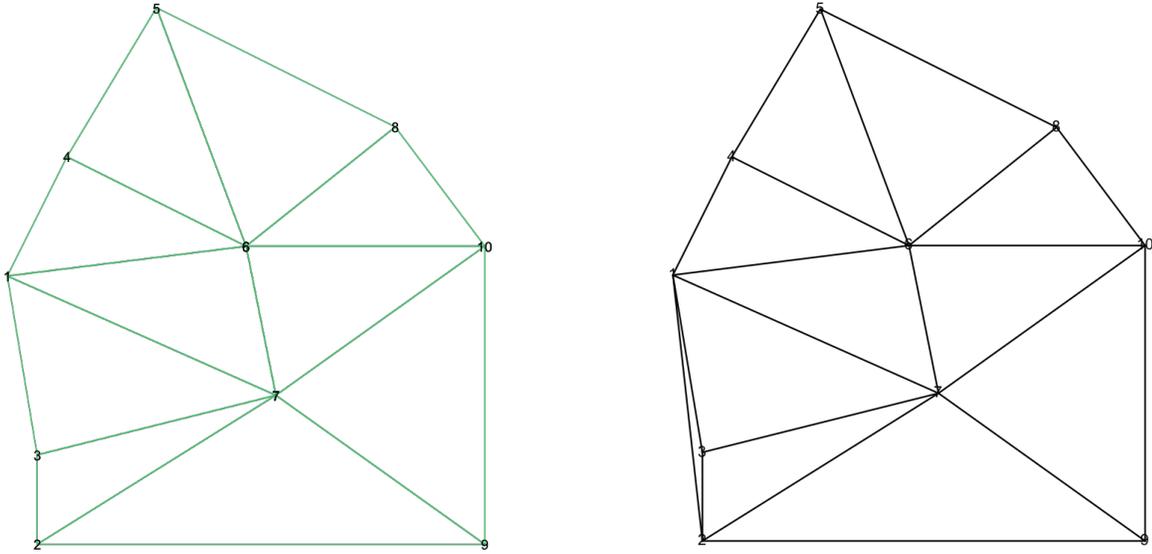


Fig. 4.16: Malla de 10 puntos.

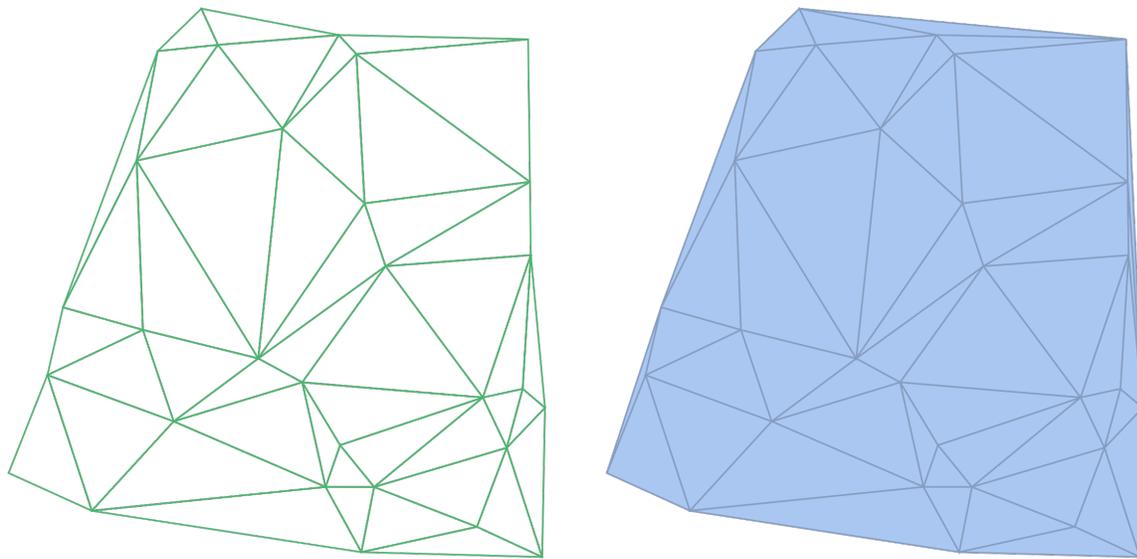


Fig. 4.17: Malla de 30 puntos.

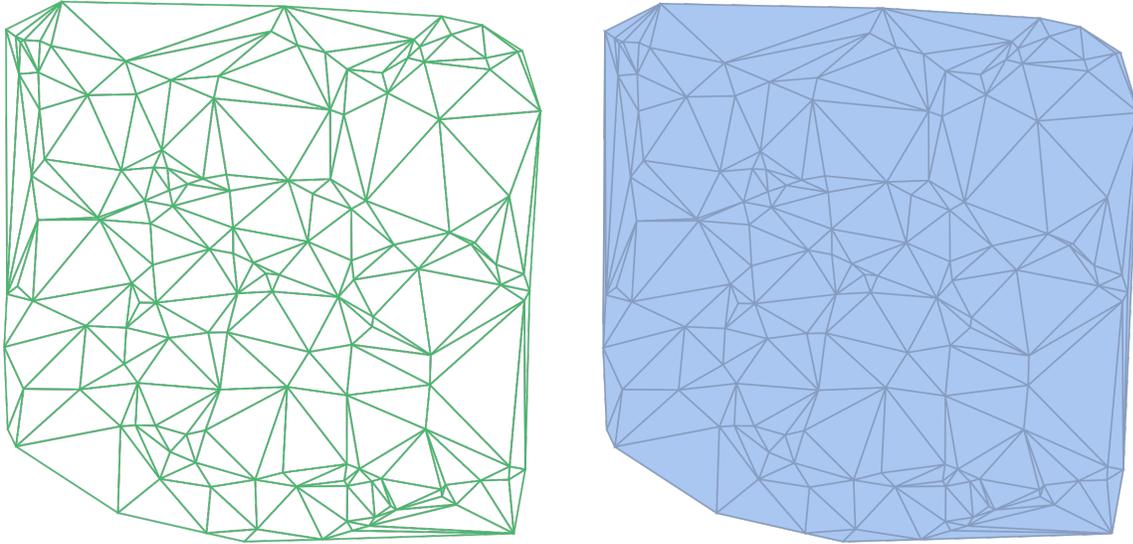


Fig. 4.18: Malla de 150 puntos.

Los resultados obtenidos con el código son prácticamente iguales comparados con los de Mathematica. Hay un detalle en el que se puede mejorar, el *convex hull*. Se puede visualizar como si los puntos fueran clavos fijados en el plano, el *convex hull* sería la región encerrada por una liga elástica alrededor de los clavos externos [5]. En la Fig. 4.16 se puede ver que en la imagen de la izquierda hace falta una arista del lado izquierdo, la que une los puntos 1 y 2 (Fig. 4.19). Al igual que en la Fig. 4.17 la imagen de la izquierda faltan dos aristas del *convex hull* (Fig. 4.20).

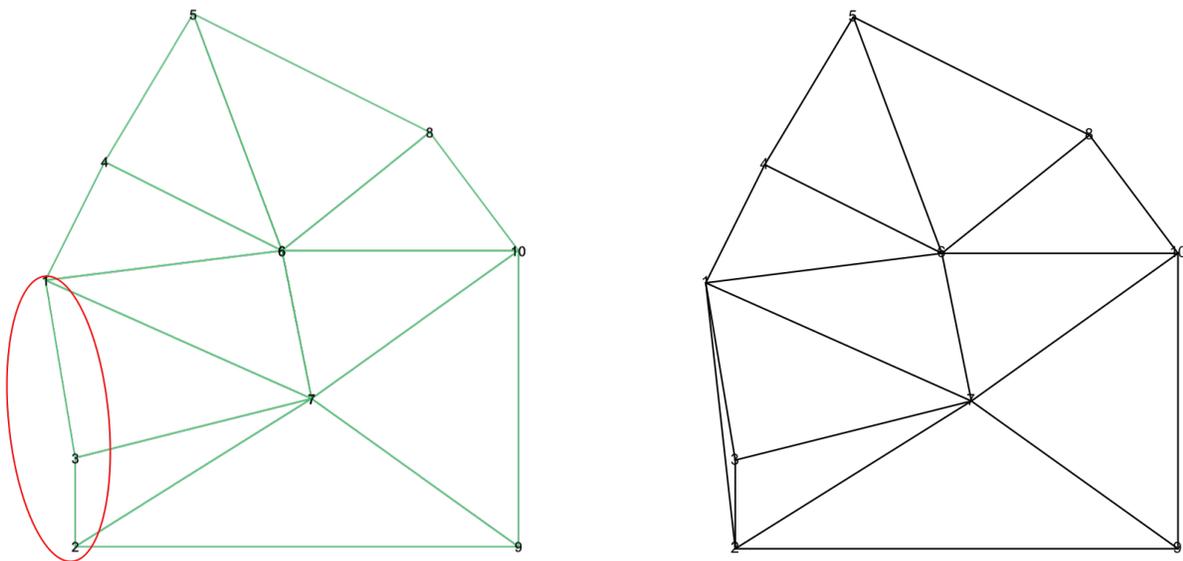


Fig. 4.19: Diferencias en el convex hull 10 puntos.

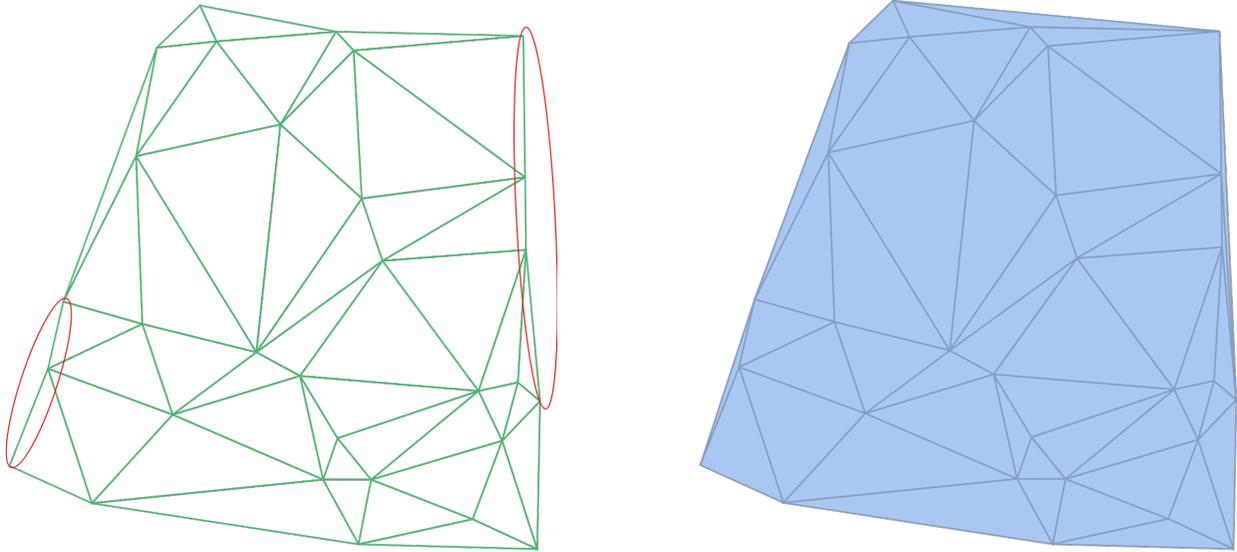


Fig. 4.20: Diferencias en el convex hull 30 puntos.

5. CONCLUSIONES

El fenómeno de transporte de contaminantes está descrito por la ecuación de advección-difusión-reacción. Las bases son los principios físicos de cada una de las partes, las cuales son independientes una de la otra.

Esta tesis se basó principalmente en tres partes fundamentales: los fenómenos físicos de la ecuación y las bases matemáticas para su resolución, la implementación del código y su aplicación al estado de Morelos.

El fenómeno de advección es la propagación horizontal del viento responsable de la transferencia de materia, el cual es dependiente de la velocidad. La difusión es la disminución, descomposición de la concentración del contaminante relacionada con el coeficiente de difusión, el cual está relacionado con densidad y temperatura. La parte de reacción es en la que más libertad se tiene. Ésta puede ser un aumento de concentración, si es una reacción de formación o si hay un suministro a lo largo de la trayectoria, o una disminución, ya sea por parte de una reacción de formación de otra especie en donde se utilice como reactivo la especie de estudio o si hay un sumidero en la trayectoria el cual absorbe concentración.

El desarrollo matemático ya es conocido, sin embargo, no se tiene en la bibliografía una derivación hecha con la cual se puede trabajar para el código. Por lo tanto, el obtener la derivación de la ecuación, tanto para la forma estacionaria como para la forma no estacionaria, fue necesario empezar desde cero.

Se logró la implementación del código para la ecuación no estacionaria en una dimensión, el cual da resultados numéricos. Se hicieron diferentes pruebas modificando los valores de número de nodos, velocidad, coeficiente de difusión y Δt . La parte de aplicación al estado de Morelos se llevó a cabo con la construcción del perfil de viento y de las trayectorias, tratando de hacer pruebas lo más apegadas a la realidad química, como aplicando valores reportados en la literatura de diferentes especies, y siempre teniendo en cuenta que son muchos los factores que influyen en

el proceso. También se pudieron hacer variaciones en la parte de reacción, agregando una función para simular un suministro de contaminantes.

El perfil de viento es la parte donde se unen muchos más factores, la mayoría fuera de nuestro dominio, ya que para hacer un perfil de viento lo suficientemente representativo del estado se necesitarían muchas estaciones de monitoreo y muchas lecturas de datos con diferentes escalas de tiempo, ya sea por hora, por día, mes, etc. Y aún teniendo esto las variables meteorológicas que se involucran son muchas; temperatura, humedad, radiación solar, estación del año, etc.

Se logró establecer la base matemática y la implementación del código para la malla de Delaunay, haciendo uso de geometría computacional. Los resultados en ésta parte son muy similares a Mathematica, un software ya establecido con el cual es fácil hacer la triangulación.

Sabemos que el resultado es un código primitivo pero cumple con las bases tanto físicas y matemáticas, el cual, si se brindara un perfil de viento completo mejoraría sus resultados.

Bibliografía

- [1] H. Saldarriga-Noreña, L. Hernández-Mena, E. Sánchez-Salinas, F. Ramos-Quintana, L. Ortiz-Hernández, R. Morales-Cueto, V. Alarcón-González, and S. Ramírez-Jiménez, “Ionic composition in aqueous extracts from PM2.5 in ambient air at the city of Cuernavaca, México,” Journal of Environmental Protection, vol. 5, pp. 1305–13 115, 2014.
- [2] A. R. Laboratory, “Hybrid single-particle lagrangian integrated trajectory (hysplit) model,” Marzo, 2015.
- [3] J. Coiffier, Fundamentals of Numerical Weather Prediction. Cambridge, 2011.
- [4] I. Koutromanos, Fundamentals of Finite Element Analysis: Linear Finite Element Analysis. Wiley, 2018.
- [5] J. O. S. L. Devadoss, Discrete and Computational Geometry. Princeton University Press, 2011.
- [6] M. v. K. M. de Berg, O. Cheong and M. Overmars, Computational Geometry Algorithms and Applications. Springer, 2008.
- [7] Google. (Agosto, 2018) Map data 2018 google, inegi. [Online]. Available: <https://www.google.com/maps/@18.7889182,-99.2179453,10z/data=!3m1!4b1!4m2!6m1!1s1ItQMBhIYDCVpYYalMGs0ZsNBLaI>
- [8] D. Vallero, Fundamentals of Air Pollution, 4th ed. Academic Press, 2008.
- [9] D. J. Stensrud, Parametrization Schemes: Key to Understanding Numerical Weather Prediction Models. Cambridge, 2011.
- [10] M. Jacobson, Fundamentals of Atmospheric Modeling. Cambridge University Press, 2005.

- [11] T. Tomkins, Numerical Weather and Climate Prediction. Cambridge, 2011.
- [12] S. Farlow, Partial Differential Equations for Scientists and Engineers. Dover Publications, 1993.
- [13] INECC. (Junio, 2013) Contaminantes primarios y secundarios. [Online]. Available: <http://www.inecc.gob.mx/calibre-informacion-basica/525-calibre-cont-primarios-secundarios>
- [14] T. T. Z. Li, Z. Qiao, Numerical Solution of Differential Equations: Introduction to Finite Difference and Finite Element Method. Cambridge University Press, 2018.
- [15] W. A. Strauss, Partial Differential Equations An Introduction. Wiley, 2008.
- [16] D. Buske, M. T. Vilhena, T. Tirabassi, and B. Bodmann, “Air pollution steady-state advection-diffusion equation: The general three-dimensional solution,” Journal of Environmental Protection, vol. 3, pp. 1124–1134, 2012.
- [17] B. Sjodin. (2018) What’s the difference between fem, fdm and fvm? [Online]. Available: <https://www.machinedesign.com/fea-and-simulation/what-s-difference-between-fem-fdm-and-fvm>
- [18] R. J. LeVeque, Finite Difference Methods for Ordinary and Partial Differential Equations. SIAM, 2007.
- [19] O. Rios-Ruiz, “Study on the one-dimensional linear advection-diffusion equation with finite differences and linear finite elements methods,” EAFIT University, Tech. Rep., 11 2015.
- [20] A. Bowyer, “Computing dirichlet tessellations,” Computer Journal, vol. 24, no. 2, pp. 162–166, 1981.
- [21] D. F. Watson, “Computing the n-dimensional delaunay tessellation with application to voronoi polytopes,” Computer Journal, vol. 24, no. 2, pp. 167–172, 1981.

- [22] J. O. E.B. Becker, G.F. Carey, Finite Elements. An introduction. Prentice-Hall, 1981.
- [23] M. S. Gockenbach, Partial Differential Equations, Analytical and Numerical Methods. SIAM, 2011.
- [24] A. Bourlioux and M. J. Gander, Modern Methods in Scientific Computing and Applications. Springer, 2011.
- [25] B. J. Kirby, Micro- and Nanoscale Fluid Mechanics. Transport in Microfluidic Devices. Cambridge, 2010.
- [26] P. Bourke. (1989) Triangulation subroutine. [Online]. Available: <https://paulbourke.net/papers/triangulate/triangulate>
- [27] INIFAP. (2012) Instituto nacional de investigaciones forestales, agrícolas y pesqueras. [Online]. Available: <http://http://clima.inifap.gob.mx/redinifap/estaciones.aspx>
- [28] INEGI. (Octubre, 2017) Mapa condensado estatal 1:175 000. [Online]. Available: <http://www.beta.inegi.org.mx/app/biblioteca/ficha.html?upc=702825687410>
- [29] I. Monstinsky. (February, 2011) Diffusion coefficient. [Online]. Available: <http://www.thermopedia.com/content/696/>
- [30] W. R. Inc., “Mathematica, Version 10,” champaign, IL, 2018.

6. APÉNDICES

6.1. Código de FEM en 1D

```
1 module ut_adr
2
3 use numerico
4
5 implicit none
6
7 real(dp), allocatable :: M(:,,:), K(:,,:), F(:), u(:), AA(:,,:), krk(:,,:), c0(:), c(:)
8 real(dp) :: peso2(2) = [1._dp, 1._dp]
9 real(dp) :: abscisa2(2) = [-1._dp/sqrt(3._dp), 1._dp/sqrt(3._dp)]
10 integer :: nodos, ncol, nodes
11 real(dp) :: peso8(8) = [0.3626837833783620_dp, &
12 0.3626837833783620_dp, &
13 0.3137066458778873_dp, 0.3137066458778873_dp, &
14 0.2223810344533745_dp, 0.2223810344533745_dp, &
15 0.1012285362903763_dp, 0.1012285362903763_dp]
16 real(dp) :: abscisa8(8) = [-0.1834346424956498_dp, &
17 0.1834346424956498_dp, &
18 -0.5255324099163290_dp, 0.5255324099163290_dp, &
19 -0.7966664774136267_dp, 0.7966664774136267_dp, &
20 -0.9602898564975363_dp, 0.9602898564975363_dp]
21 real(dp) :: a = 0._dp, b, L = 100._dp, h, dt = 0.02_dp, alfa = 5._dp
22 real(dp) :: integralmdia = 0._dp, integralsup = 0._dp
23 real(dp), allocatable :: nodelist(:), listvelo(:), puntos(:, :)
24
25 end module !ut_adr
26 !-----
27 program main
28
29 use numerico
30 use ut_adr
```

```
31
32 implicit none
33
34 integer      :: i, j, ii, jj
35 real(dp)     :: vel, pruebaM
36
37 open(10, file = '49nodos.dat')
38
39 read(10,*) nodes
40
41 nodos = nodes - 2
42 ncol = nodos + 1
43
44 allocate(M(nodos,nodos),K(nodos,nodos),F(nodos),c0(nodos),c(nodos),u(nodos), &
45 AA(nodos,ncol),krk(nodos,ncol),nodelist(nodes),listvelo(nodes),puntos(nodes,2))
46
47 read(10,*) (nodelist(i), i = 1, nodes)
48
49 read(10,*) (listvelo(i), i = 1, nodes)
50
51 read(10,*) (c0(i), i = 1, nodes)
52
53 close(10)
54
55 M = 0._dp
56 K = 0._dp
57 F = 0._dp
58 c = 0._dp
59 u = 0._dp ! Ax = u
60 AA = 0._dp
61 krk = 0._dp
62
63 write(*,*) 'Nodes'
64 do i = 1, nodes
65   write(*, '(*(f14.6))') nodelist(i)
66 end do
67
```

```

68 write(*,*) 'Lista velocidades'
69 do i = 1, nodes
70   write(*,'*(f14.6)') listvelo(i)
71 end do
72
73 do i = 1, nodes
74   puntos(i,1) = nodelist(i)
75   puntos(i,2) = listvelo(i)
76 end do
77
78 write(*,*) 'PUNTOS'
79 do i = 1, nodes
80   write(*,'*(f14.6)') (puntos(i,j), j = 1, 2)
81 end do
82
83 !----- M -----
84 !-----DIAGONAL-----
85 call pruebaMdiag(nodelist(1),nodelist(2),nodelist(3),M(1,1))
86 !-----NO DIAGONAL-----
87 call pruebaMnodia(nodelist(1),nodelist(2),nodelist(3),M(1,2))
88
89 !----- K -----
90 !-----DIAGONAL-----
91 call pruebaKdiag(puntos(1,1),puntos(1,2),puntos(2,1),puntos(2,2),puntos(3,1), &
92 puntos(3,2),K(1,1))
93 !-----NO DIAGONAL-----
94 call pruebaKsup(puntos(1,1),puntos(1,2),puntos(2,1),puntos(2,2),puntos(3,1), &
95 puntos(3,2),K(1,2))
96 !----- F -----
97 call pruebaF(nodelist(1),nodelist(2),nodelist(3),F(1))
98
99 do i = 2, nodos-1
100  call pruebaMdiag(nodelist(i),nodelist(i+1),nodelist(i+2),M(i,i))
101  call pruebaMnodia(nodelist(i),nodelist(i+1),nodelist(i+2),M(i,i-1)) !INF
102  call pruebaMnodia(nodelist(i),nodelist(i+1),nodelist(i+2),M(i,i+1)) !SUP
103  call pruebaKdiag(puntos(i,1),puntos(i,2),puntos(i+1,1),puntos(i+1,2), &
104  puntos(i+2,1),puntos(i+2,2),K(i,i))

```

```

105  call pruebaKsup(puntos(i,1),puntos(i,2),puntos(i+1,1),puntos(i+1,2), &
106  puntos(i+2,1),puntos(i+2,2),K(i,i+1))
107  call pruebaKinf(puntos(i-1,1),puntos(i-1,2),puntos(i,1),puntos(i,2), &
108  puntos(i+1,1),puntos(i+1,2),K(i,i-1))
109  call pruebaF(nodelist(i-1),nodelist(i),nodelist(i+1),F(i))
110  end do !i
111
112  !----- M -----
113  !-----DIAGONAL-----
114  call pruebaMdiag(nodelist(nodos-2),nodelist(nodos-1),nodelist(nodos), &
115  M(nodos,nodos))
116  !-----NO DIAGONAL-----
117  call pruebaMnodiag(nodelist(nodos-2),nodelist(nodos-1),nodelist(nodos), &
118  M(nodos,nodos-1))
119
120  !----- K -----
121  !-----DIAGONAL-----
122  call pruebaKdiag(puntos(nodos-2,1),puntos(nodos-2,2),puntos(nodos-1,1), &
123  puntos(nodos-1,2), puntos(nodos,1),puntos(nodos,2),K(nodos,nodos))
124  !-----NO DIAGONAL-----
125  call pruebaKinf(puntos(nodos-3,1),puntos(nodos-3,2),puntos(nodos-2,1), &
126  puntos(nodos-2,2), puntos(nodos-1,1),puntos(nodos-1,2),K(nodos,nodos-1))
127  !----- F -----
128  call pruebaF(nodelist(nodos-2),nodelist(nodos-1),nodelist(nodos),F(nodos))
129
130  write(*,*) 'M'
131  do i = 1, nodos
132    write(*,'(*(f10.4))') (M(i,j), j = 1, nodos)
133  end do !i
134
135  write(*,*) 'K'
136  do i = 1, nodos
137    write(*,'(*(f10.4))') (K(i,j), j = 1, nodos)
138  end do !i
139
140  write(*,*) 'F'
141  do i = 1, nodos

```

```
142 write(*,'*(f10.5)') F(i)
143 end do !i
144
145 write(*,*) 'c inicial'
146 do i = 1, nodos
147 write(*,'*(f14.6)') c0(i)
148 end do !i
149
150 c = c0
151
152 do jj = 1, 100
153 write(*,*) '-----PASO', jj, '-----'
154 u = 0._dp
155
156 if (jj /= 1) then !segundo ciclo
157 c0 = c
158 end if ! jj
159
160 write(*,*) 'c inicial'
161 do i = 1, nodos
162 write(*,'*(f16.6)') c(i)
163 end do !i
164
165 call rk4
166 write(*,*) '-K.c + F = u'
167 do i = 1, nodos
168 write(*,'*(f16.6)') u(i)
169 end do !i
170
171 do i = 1, nodos
172 AA(:,i) = M(:,i)
173 end do !i
174 AA(:,nodos+1) = u(:)
175
176 do ii = 1, 4
177
178 if (ii == 4) then !k4
```

```
179 write(*,*) 'AA', ii
180 do i = 1, nodos
181   write(*,'*(f16.6)') (AA(i,j), j = 1, nodos+1)
182 end do !i
183
184 call Gauss(AA,u,krk(ii,:))
185 write(*,*) 'k', ii
186 do i = 1, nodos
187   write(*,'*(f16.6)') krk(ii,i)
188 end do
189
190 write(*,*) 'F'
191 do i = 1, nodos
192   write(*,'*(f16.6)') F(i)
193 end do !i
194
195 write(*,*) 'c nuevo'
196 do i = 1, nodos
197   c(i) = c0(i) + (krk(1,i) + (2._dp*krk(2,i)) + (2._dp*krk(3,i)) + &
198     krk(4,i))*dt/6._dp
199   write(*,'*(f16.6)') c(i)
200 end do !i
201
202 AA(:,nodos+1) = u
203
204 else !k1,k2,k3
205   write(*,*) 'AA', ii
206   do i = 1, nodos
207     write(*,'*(f16.6)') (AA(i,j), j = 1, nodos+1)
208   end do !i
209
210   call Gauss(AA,u,krk(ii,:))
211   write(*,*) 'k', ii
212   do i = 1, nodos
213     write(*,'*(f16.6)') krk(ii,i)
214   end do
215
```

```
216   if (ii == 3) then
217       write(*,*) 'c nuevo'
218       do i = 1, nodos
219           c(i) = c0(i) + krk(ii,i)*dt
220           write(*,'*(f16.6)') c(i)
221       end do !i
222
223   else !k1 y k2
224
225       write(*,*) 'c nuevo'
226       do i = 1, nodos
227           c(i) = c0(i) + krk(ii,i)*(dt/2._dp)
228           write(*,'*(f16.6)') c(i)
229       end do !i
230
231   end if
232
233   write(*,*) 'F'
234   do i = 1, nodos
235       write(*,'*(f16.6)') F(i)
236   end do !i
237
238   u = 0._dp
239
240   call rk4
241   write(*,*) '-K.c + F = u'
242   do i = 1, nodos
243       write(*,'*(f16.6)') u(i)
244   end do !i
245   AA(:,nodos+1) = u
246
247   end if ! ii
248
249   end do !ii
250
251   end do !jj
252
```

```
253 end program !main
254 !-----
255 subroutine pruebaMdiag(la,cnode,lb,integral)
256
257 use numerico
258 use ut_adr
259
260 implicit none
261
262 real(dp)      :: la, lb, cnode
263 real(dp)      :: jacobco, mdiag, integral
264
265 integral = (jacobco(la,cnode)* &
266 ((peso2(1)* &
267 (mdiag(((abscisa2(1)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb))) + &
268 (peso2(2)* &
269 (mdiag(((abscisa2(2)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb)))))) + &
270 (jacobco(cnode,lb)* &
271 ((peso2(1)* &
272 (mdiag(((abscisa2(1)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb))) + &
273 (peso2(2)* &
274 (mdiag(((abscisa2(2)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb))))))
275
276 end subroutine ! pruebaMdiag
277 !-----
278 subroutine pruebaMnodiag(la,cnode,lb,integralnodiag)
279
280 use numerico
281 use ut_adr
282
283 implicit none
284
285 real(dp)      :: jacobco
286 real(dp)      :: mnodiag, integralnodiag
287
288 integralnodiag = (jacobco(la,cnode)* &
289 ((peso2(1)* &
```

```

290 (mnode((abscisa2(1)*(cnode - a) + cnode + a)/2._dp),a,cnode))) + &
291 (peso2(2)* &
292 (mnode((abscisa2(2)*(cnode - a) + cnode + a)/2._dp),a,cnode))))))
293
294 end subroutine !pruebaMnode
295 !-----
296 real(dp) function mnode(x,a,cnode,b)
297
298 use numerico
299
300 implicit none
301
302 real(dp)      :: x, a, b, cnode
303 real(dp)      :: inter_l, inter_r
304
305 inter_l = cnode - a
306 inter_r = b - cnode
307
308 if (a <= x .and. x < cnode) then !left
309   mnode = (1._dp + ((x - cnode)/inter_l)**2
310 elseif (cnode <= x .and. x < b) then !right
311   mnode = (1._dp - ((x - cnode)/inter_r)**2
312 else
313   mnode = 0._dp
314 end if
315
316 end function !mnodeLR
317 !-----
318 !NO DIAGONAL M
319 real(dp) function mnode(x,a,cnode)
320
321 use numerico
322
323 implicit none
324
325 real(dp)      :: x, a, cnode, intervalo
326

```

```

327 intervalo = cnode - a
328
329 mndiag = (1._dp + ((x - cnode)/intervalo))* &
330         (1._dp - ((x - a)/intervalo))
331
332 end function !msup
333 !-----
334 !Matriz K
335 subroutine pruebaKdiag(la,vela,cnode,velc,lb,velb,kdiag)
336
337 use numerico
338 use ut_adr
339
340 implicit none
341
342 real(dp)      :: la, lb, cnode, vela, velc, velb
343 real(dp)      :: kdiagdif, integralkdif, ikdif = 0._dp
344 real(dp)      :: kdiagadv, integralkadv = 0._dp, ikadv = 0._dp
345 real(dp)      :: kdiag, jacobobo, ec2p
346
347 !DIFUSION
348 integralkdif = (jacobobo(la,cnode)* &
349 ((peso2(1)* &
350 (kdiagdif(((abscisa2(1)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb))) + &
351 (peso2(2)* &
352 (kdiagdif(((abscisa2(2)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb)))))) + &
353 (jacobobo(cnode,lb)* &
354 ((peso2(1)* &
355 (kdiagdif(((abscisa2(1)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb))) + &
356 (peso2(2)* &
357 (kdiagdif(((abscisa2(2)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb))))))
358 ikdif = (alfa)**2*(integralkdif)
359
360 !ADVECCION
361 integralkadv = (jacobobo(la,cnode))* &
362 ((peso2(1)* &
363 (ec2p(la,vela,cnode,velc,((abscisa2(1)*(cnode - la) + cnode + la)/2._dp))))* &

```

```

364 (kdiagadv(((abscisa2(1)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb))) + &
365 (peso2(2)* &
366 (ec2p(la,vela,cnode,velc,((abscisa2(2)*(cnode - la) + cnode + la)/2._dp)))* &
367 (kdiagadv(((abscisa2(2)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb))) + &
368 (jacoboc(cnode,lb))* &
369 ((peso2(1)* &
370 (ec2p(cnode,velc,lb,velb,((abscisa2(1)*(lb - cnode) + lb + cnode)/2._dp)))* &
371 (kdiagadv(((abscisa2(1)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb))) + &
372 (peso2(2)* &
373 (ec2p(cnode,velc,lb,velb,((abscisa2(2)*(lb - cnode) + lb + cnode)/2._dp)))* &
374 (kdiagadv(((abscisa2(2)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb)))
375 ikadv = integralkadv
376
377 !DIAGONAL
378 kdiag = ikdif + ikadv
379
380 end subroutine
381 !-----
382 !K DIAGONAL DIFUSION
383 real(dp) function kdiagdif(x,a,cnode,b)
384
385 use numerico
386
387 implicit none
388
389 real(dp) :: x, a, cnode, b, intervalo_R, intervalo_L
390
391 intervalo_L = cnode - a
392 intervalo_R = b - cnode
393
394 if (a <= x .and. x < cnode) then
395   kdiagdif = (1._dp / intervalo_L)*(1._dp / intervalo_L)
396 elseif (cnode <= x .and. x < b) then
397   kdiagdif = (-1._dp / intervalo_R)*(-1._dp / intervalo_R)
398 else
399   kdiagdif = 0._dp
400 end if

```

```
401
402 end function ! kdiagdif
403 !-----
404 !K DIAGONAL ADVECCION
405 real(dp) function kdiagadv(x,a,cnode,b)
406
407 use numerico
408
409 implicit none
410
411 real(dp) :: x, a, cnode, b, intervaloL, intervaloR, vel
412
413 intervaloL = cnode - a
414 intervaloR = b - cnode
415
416 if (a <= x .and. x < cnode) then
417   kdiagadv = (1._dp / intervaloL)* &
418             (1._dp + ((x - cnode) / intervaloL))
419 elseif(cnode <= x .and. x < b) then
420   kdiagadv = (-1._dp / intervaloR)* &
421             (1._dp - ((x - cnode)/intervaloR))
422 else
423   kdiagadv = 0._dp
424 end if
425
426 end function ! kdiagadv
427 !-----
428 subroutine pruebaKsup(la,vela,cnode,velc,lb,velb,iksup)
429
430 use numerico
431 use ut_adr
432
433 implicit none
434
435 real(dp) :: la, lb, cnode, vela, velc, velb
436 real(dp) :: iknodiagDifS = 0._dp, knodiagdif, difknodiagS = 0._dp
437 real(dp) :: iknodiagAs = 0._dp, knodiagAs, iknodiagAdv = 0._dp, iknodiagSup
```

```

438 real(dp)      :: iksup, jacob, ec2p
439
440 !DIFUSION
441 difknodiagS = (jacob(la,cnode)* &
442 ((peso2(1)*(knodiagdif(((abscisa2(1)*(cnode - la) + cnode + la)/2._dp),la,cnode))) + &
443 (peso2(2)*(knodiagdif(((abscisa2(2)*(cnode - la) + cnode + la)/2._dp),la,cnode))))))
444 iknodiagDifS = (alfa)**2*(difknodiagS)
445
446 !ADVECCION
447 iknodiagAs = (jacob(cnode,lb))* &
448 ((peso2(1)* &
449 (ec2p(cnode,velc,lb,velb,((abscisa2(1)*(lb - cnode) + lb + cnode)/2._dp)))* &
450 (knodiagAs(((abscisa2(1)*(lb - cnode) + lb + cnode)/2._dp),cnode,lb))) + &
451 (peso2(2)* &
452 (ec2p(cnode,velc,lb,velb,((abscisa2(2)*(lb - cnode) + lb + cnode)/2._dp)))* &
453 (knodiagAs(((abscisa2(2)*(lb - cnode) + lb + cnode)/2._dp),cnode,lb))))
454 iknodiagSup = iknodiagAs
455
456 !SUP
457 iksup = iknodiagDifS + iknodiagSup
458
459 end subroutine !pruebaKnodiag
460 !-----
461 !NO DIAGONAL K DIFUSION
462 real(dp) function knodiagdif(x,a,cnode)
463
464 use numerico
465
466 implicit none
467
468 real(dp)  :: x, a, cnode, intervalo
469
470 intervalo = cnode - a
471
472 knodiagdif = (-1._dp / intervalo)*(1._dp / intervalo)
473
474 end function !knodiagdif

```

```

475 !-----
476 !NO DIAGONAL K ADVECCION SUPERIOR
477 real(dp) function knodiagAs(x,a,cnode)
478
479 use numerico
480
481 implicit none
482
483 real(dp)  :: x, a, cnode, intervalo, vel
484
485 intervalo = cnode - a
486
487 knodiagAs = (-1._dp / intervalo)* &
488             (1._dp + ((x - cnode)/intervalo))
489
490 end function !kadv_sup
491 !-----
492 subroutine pruebaKinf(la,vela,cnode,velc,lb,velb,ikinf)
493
494 use numerico
495 use ut_adr
496
497 implicit none
498
499 real(dp)  :: la, lb, cnode, vela, velc, velb
500 real(dp)  :: difknodiagI = 0._dp, iknodiagDifI = 0._dp, knodiagdif
501 real(dp)  :: iknodiagAi = 0._dp, iknodiagInf = 0._dp, kadvInf
502 real(dp)  :: ikinf, jacobobo, ec2p
503
504 !DIFUSION
505 difknodiagI = (jacobobo(la,cnode)* &
506              ((peso2(1)* &
507              (knodiagdif(((abscisa2(1)*(cnode - la) + cnode + la)/2._dp),la,cnode))) + &
508              (peso2(2)* &
509              (knodiagdif(((abscisa2(2)*(cnode - la) + cnode + la)/2._dp),la,cnode))))))
510 iknodiagDifI = (alfa)**2*(difknodiagI)
511

```

```

512 !ADVECCION
513 iknodiagAi = (jacobocnnode,lb))* &
514 ((peso2(1)* &
515 (ec2p(cnode,velc,lb,velb,((abscisa2(1)*(lb - cnode) + lb + cnode)/2._dp)))* &
516 (kadvInf(((abscisa2(1)*(lb - cnode) + lb + cnode)/2._dp),cnode,lb))) + &
517 (peso2(2)* &
518 (ec2p(cnode,velc,lb,velb,((abscisa2(2)*(lb - cnode) + lb + cnode)/2._dp)))* &
519 (kadvInf(((abscisa2(2)*(lb - cnode) + lb + cnode)/2._dp),cnode,lb))))
520 iknodiagInf = iknodiagAi
521
522 !INF
523 ikinf = iknodiagDifI + iknodiagInf
524
525 end subroutine !pruebaKnodiag
526 !-----
527 !NO DIAGONAL K ADVECCION INFERIOR
528 real(dp) function kadvInf(x,a,cnode)
529
530 use numerico
531
532 implicit none
533
534 real(dp) :: x, a, cnode, intervalo, vel
535
536 intervalo = cnode - a
537
538 kadvInf = (1._dp / intervalo)* &
539           (1._dp - ((x - a)/intervalo))
540
541 end function !kodiagAi
542 !-----
543 subroutine pruebaF(la,cnode,lb,integralF)
544
545 use numerico
546 use ut_adr
547
548 implicit none

```

```

549
550 real(dp)      :: la, lb, cnode
551 real(dp)      :: integralF
552 real(dp)      :: funcf, jacobob
553
554 integralF = (jacobob(la,cnode)* &
555 ((peso8(1)*funcf(((abscisa8(1)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb)) + &
556 (peso8(2)*funcf(((abscisa8(2)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb)) + &
557 (peso8(3)*funcf(((abscisa8(3)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb)) + &
558 (peso8(4)*funcf(((abscisa8(4)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb)) + &
559 (peso8(5)*funcf(((abscisa8(5)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb)) + &
560 (peso8(6)*funcf(((abscisa8(6)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb)) + &
561 (peso8(7)*funcf(((abscisa8(7)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb)) + &
562 (peso8(8)*funcf(((abscisa8(8)*(cnode - la) + cnode + la)/2._dp),la,cnode,lb)))) + &
563 (jacobob(cnode,lb)* &
564 ((peso8(1)*funcf(((abscisa8(1)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb)) + &
565 (peso8(2)*funcf(((abscisa8(2)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb)) + &
566 (peso8(3)*funcf(((abscisa8(3)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb)) + &
567 (peso8(4)*funcf(((abscisa8(4)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb)) + &
568 (peso8(5)*funcf(((abscisa8(5)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb)) + &
569 (peso8(6)*funcf(((abscisa8(6)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb)) + &
570 (peso8(7)*funcf(((abscisa8(7)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb)) + &
571 (peso8(8)*funcf(((abscisa8(8)*(lb - cnode) + lb + cnode)/2._dp),la,cnode,lb))))
572
573 end subroutine !pruebaF
574 !-----
575 !FUNCION F
576 real(dp) function funcf(x,a,cnode,b)
577
578 use numerico
579
580 implicit none
581
582 real(dp)  :: x, a, b, cnode, inter_l, inter_r
583 real(dp)  :: gaussextra
584
585 inter_l = cnode - a

```

```
586 inter_r = b - cnode
587
588 if (a <= x .and. x < cnode) then !left
589   funcf = (exp(-(x - 75._dp)**2/100._dp))* &
590           (1._dp + ((x - cnode)/inter_l)) + &
591           gaussextra(x)
592 elseif (cnode <= x .and. x < b) then !right
593   funcf = (exp(-(x - 75._dp)**2/100._dp))* &
594           (1._dp - ((x - cnode)/inter_r)) + &
595           gaussextra(x)
596 else
597   funcf = 0._dp
598 end if
599
600 end function !knodiagAi
601 !-----
602 real(dp) function gaussextra(x)
603
604 use numerico
605 use ut_adr
606
607 implicit none
608
609 real(dp)      :: x
610
611 gaussextra = pesoConc*exp(-0.5_dp*(x-posicionCivac)**2)
612
613 end function !gaussextra
614 !-----
615 real(dp) function vel(exq)
616
617 use numerico
618
619 implicit none
620
621 real(dp)      :: exq
622
```

```
623 vel = (-0.02_dp*exq + 1.1_dp)
624
625 end function !vel
626 !-----
627 real(dp) function ec2p(x1,y1,x2,y2,exq)
628
629 use numerico
630
631 implicit none
632
633 real(dp)      :: exq, x1, y1, x2, y2
634
635 ec2p = (y2 - y1)*((exq - x1)/(x2 - x1)) + y1
636
637 end function !vel
638
639 !-----
640 real(dp) function jacobco(x,y)
641
642 use numerico
643
644 implicit none
645
646 real(dp)      :: x, y
647
648 jacobco = (y - x)/2._dp
649
650 end function !jacobco
651 !-----
652 subroutine rk4
653
654 use numerico
655 use ut_adr
656
657 implicit none
658
659 integer      :: i, j
```

```
660
661 do i = 1, nodos
662   do j = 1, nodos
663     u(i) = u(i) + (-K(i,j)*c(j))
664   end do !j
665   u(i) = u(i) + F(i)
666 end do !i
667
668 end subroutine !rk4
669 !-----
670 subroutine Gauss(Amat,bu,x)
671
672 use numerico
673 use ut_adr
674
675 implicit none
676
677 integer      :: i, j, z, zz
678 integer      :: columna, renglon, colmax
679 real(dp)     :: pivoto = 0._dp, suma = 0._dp
680 real(dp)     :: Amat(nodos,ncol), Amat0(nodos,ncol)
681 real(dp)     :: x(nodos), bu(nodos), tempcol(ncol)
682
683 Amat0 = Amat
684
685 do j = 1, nodos-1
686   do i = j+1, nodos
687     pivoto = Amat(i,j)/Amat(j,j)
688     Amat(i,:) = Amat(i,:) - pivoto*(Amat(j,:))
689   end do ! i = j+1
690 end do ! j = 1
691
692 x(nodos) = Amat(nodos,ncol)/Amat(nodos,ncol-1)
693
694 do i = nodos-1, 1, -1
695   suma = 0
696   do j = i+1, nodos
```

```
697 suma = suma + Amat(i,j)*x(j)
698 end do !j
699 x(i) = (Amat(i,ncol) - suma)/Amat(i,i)
700 end do ! i
701
702 Amat = Amat0
703
704 return
705
706 end subroutine !Gauss
```

6.2. Código de triangulación de Delaunay

```
1 program femMorelos
2
3 use numerico
4 use malla
5
6 implicit none
7
8 call leerMalla
9 call burbujaSort
10 call centroCirculo
11 call superTriangulo
12 call pruebaPaso1
13
14 end program femMorelos
```

```
1 module malla
2
3 use numerico
4
5 implicit none
6 save
7 integer :: npuntos, nvert, ntriang, ciclo, contador = 0
```

```

8  real(dp), allocatable      :: coord(:,:), ordCoord(:,:), vertices(:,:)
9  real(dp)                  :: centroCirculoInterno(2), A(2),B(2),C(2)
10 real(dp)                  :: radioCirculoInterno, distRadio, distDist
11 real(dp)                  :: AB, AC, BC, centro(2)
12 real(dp)                  :: circunCentro(2), radio, distr
13 integer, allocatable      :: listaTriang(:,:), arista(:,:), &
14                             adentroCirculo(:), nuevosTriang(:,:)
15 integer, allocatable      :: compartidas(:,:), quitarST(:,:), &
16                             finales(:,:), aristaSR(:,:)
17
18 end module malla
19
20 !-----
21 !Subrutina para leer numero de total de lineas
22 !y coordenadas xy de cada punto en el archivo de datos
23
24 subroutine leerMalla
25
26 use numerico
27 use malla
28
29 implicit none
30
31 integer          :: i, ii, ierror
32 integer, parameter :: MAXLINE = 10000
33 real(dp)        :: maxValor
34
35 open(unit = 10, file = 'random_puntos6.dat', IOSTAT = ierror)
36
37 do i = 1, MAXLINE
38     read(10,*,IOSTAT=ierror)
39     if(ierror /= 0) then
40         exit
41     else
42         npuntos = npuntos + 1
43     end if ! (ierror /= 0)
44 end do ! i = 1, MAXLINE

```

```
45
46 !write(*,*) "Total de puntos en input: ", npuntos
47
48 rewind(10)
49
50 allocate(coord(npuntos,2))      !Matriz original
51 allocate(ordCoord(npuntos,2))  !Matriz que se va a ordenar
52
53 ! leer Malla
54 do ii = 1, npuntos
55     read(10, *) coord(ii,:)
56 end do ! ii = 1, npuntos
57
58 close(10)
59
60 ordCoord(:, :) = coord(:, :)
61
62 write(*,*) ' '
63 do ii = 1, npuntos
64     write(*, '(6(f15.4))') coord(ii,:)
65 end do ! ii = 1, npuntos
66
67 deallocate(coord)              !deallocate matriz original
68
69 end subroutine leerMalla
70
71 !-----
72 !Metodo burbuja para ordenar valores de xy
73 !de manera lexicografica (menor a mayor)
74
75 subroutine burbujaSort
76
77 use numerico
78 use malla
79
80 implicit none
81
```

```
82 real(dp)      :: temp
83 integer       :: ii, j
84
85 do ii = 1, npuntos-1
86   do j = ii+1, npuntos
87     if (ordCoord(ii,1) > ordCoord(j,1)) then
88       temp = ordCoord(ii,1)
89       ordCoord(ii,1) = ordCoord(j,1)
90       ordCoord(j,1) = temp
91       temp = ordCoord(ii,2)
92       ordCoord(ii,2) = ordCoord(j,2)
93       ordCoord(j,2) = temp
94     end if
95
96     if (ordCoord(ii,1) == ordCoord(j,1)) then
97       if (ordCoord(ii,2) > ordCoord(j,2)) then
98         temp = ordCoord(ii,2)
99         ordCoord(ii,2) = ordCoord(j,2)
100        ordCoord(j,2) = temp
101      end if
102    end if
103  end do
104 end do
105
106 write(*,*) 'Puntos acomodados'
107 write(*,*) ' '
108 do ii = 1, npuntos
109   write(*,'(6(f15.4))') ordCoord(ii,:)
110 end do ! ii = 1, npuntos
111
112 end subroutine burbujaSort
113 !-----
114 !Subrutina para sacar el centro de los puntos,
115 !calcular las distancias para cada punto centro-punto
116 !y calcular el punto A del super triangulo
117
118 subroutine centroCirculo
```

```
119
120 use malla
121 use numerico
122
123 implicit none
124
125 real(dp)      :: sumax = 0._dp, sumay = 0._dp, dist
126 integer      :: ii
127
128 do ii = 1, npuntos
129     sumax = sumax + ordCoord(ii,1)
130     sumay = sumay + ordCoord(ii,2)
131 end do !ii
132
133 sumax = sumax/npuntos
134 sumay = sumay/npuntos
135
136 centroCirculoInterno(1) = sumax
137 centroCirculoInterno(2) = sumay
138
139 write(*,*) 'centro circulo interno', centroCirculoInterno
140
141 do ii = 1, npuntos
142     call distanciaPuntos(centroCirculoInterno,ordCoord(ii,:),dist)
143     if (dist > radioCirculoInterno) radioCirculoInterno = dist
144 end do
145
146 !write(*,*) 'radio circulo interno', radioCirculoInterno
147 !write(*,*) ' '
148
149 A(1) = centroCirculoInterno(1)
150 A(2) = centroCirculoInterno(2)+(2*radioCirculoInterno)
151
152 end subroutine centroCirculo
153 !-----
154 !Subrutina para sacar las coordenadas del
155 !super triangulo afuera del circulo
```

```
156
157 subroutine superTriangulo
158
159 use numerico
160 use malla
161
162 implicit none
163
164 real(dp)      :: Bx, By, Cx, Cy
165 real(dp)      :: angulo, area, radioCircumcenter
166
167 angulo = (30._dp*pi)/180._dp
168
169 Bx = -(2*radioCirculoInterno)*cos(angulo) + centroCirculoInterno(1)
170 By = -(2*radioCirculoInterno)*sin(angulo) + centroCirculoInterno(2)
171 B(1) = Bx
172 B(2) = By
173
174 Cx = (2*radioCirculoInterno)*cos(angulo) + centroCirculoInterno(1)
175 Cy = -(2*radioCirculoInterno)*sin(angulo) + centroCirculoInterno(2)
176 C(1) = Cx
177 C(2) = Cy
178
179 !write(*,*) 'A', A
180 !write(*,*) 'C', C
181 !write(*,*) 'B', B
182 !write(*,*) ' '
183
184 call distanciaPuntos(A,B,AB)
185 call distanciaPuntos(A,C,AC)
186 call distanciaPuntos(B,C,BC)
187
188 !write(*,*) 'SUPER TRIANGULO'
189 !write(*,*) 'Distancia AB-AC-BC', AB
190
191 end subroutine superTriangulo
192 !-----
```

```
193 subroutine pruebaPaso1
194
195 use malla
196 use numerico
197
198 implicit none
199 integer          :: ii, j, t, m, m1, h, n, z1
200 integer          :: j3, l, l2, l5, l6, q, r, s
201 integer          :: vueltas = 0
202 integer          :: ntriangMax, contadorAdentro, contadorAfuera
203 integer          :: ipunto(npuntos)
204 integer          :: naristas, contadorExterno = 0
205 integer          :: contadorRepetidas, contadorAppend
206 integer          :: contadorST = 0, contadorFinal = 0
207
208 nvert = npuntos + 3
209 ntriangMax = 2*nvert - 2
210
211 !write(*,*) ' '
212 !write(*,*) 'puntos:', npuntos
213 !write(*,*) 'triangulos:', ntriangMax
214 !write(*,*) ' '
215
216 allocate (vertices(nvert,2), listaTriang(ntriangMax,3), &
217           adentroCirculo(ntriangMax), nuevosTriang(ntriangMax,3), &
218           arista(ntriangMax,2), compartidas(ntriangMax,2), &
219           quitarST(ntriangMax,3), finales(ntriangMax,3), &
220           aristaSR(ntriangMax,2))
221
222 !A es el apice del super triangulo equilatero
223 !C es el que esta a la derecha de A (conforme a las manecillas del reloj)
224 !B es el que esta a la izquierda de A
225 vertices(npuntos+1,:) = B(:)
226 vertices(npuntos+2,:) = A(:)
227 vertices(npuntos+3,:) = C(:)
228
229 do ii = 1, npuntos
```

```
230     vertices(ii,:) = ordCoord(ii,:)
231 end do
232
233 deallocate(ordCoord)      !deallocate matriz que se va a ordenar
234
235 !write(*,*) 'Vertices'
236 do ii = 1, npuntos
237     ipunto(ii) = ii
238     ! write(*,*) ipunto(ii), vertices(ii,:)
239 end do
240
241 !PRIMEROS 3 TRIANGULOS DE P1 CON SUPERTRIANGULO
242 do ii = npuntos+1, npuntos+2
243     n = 1
244     do j = npuntos+2, npuntos+3
245         if (ii == j) cycle
246             vueltas = vueltas + 1
247             listaTriang(vueltas,n) = n
248             listaTriang(vueltas,n+1) = ii
249             listaTriang(vueltas,n+2) = j
250             ntriang = ntriang + 1
251         end do ! j
252     end do !ii
253
254     !write(*,*) ' '
255     !write(*,*) 'Triangulos iniciales'
256     do ii = 1, ntriang
257         ! write(*,*) (listaTriang(ii,j), j = 1, 3)
258     end do !ii
259
260
261 do h = 2, npuntos
262     arista=0
263     contador=0
264     contadorAdentro = 0
265     contadorAfuera = 0
266     compartidas = 0
```

```

267 contadorAppend = 0
268 naristas = ntriang
269
270 write(*,*) '-----PUNTO-----', h write(*,*) ' '
271 do ii = 1, naristas !ntriang
272   j3 = 1
273   !CALCULAR SI EL PUNTO h ESTA ADENTRO O AFUERA DEL CENTRO
274   call puntoAdentroPaso1(vertices(listaTriang(ii,j3),:), &
275                          vertices(listaTriang(ii,j3+1),:), &
276                          vertices(listaTriang(ii,j3+2),:), centro(:))
277   call distanciaPuntos(vertices(listaTriang(ii,j3+1),:),centro(:),distRadio)
278   call distanciaPuntos(vertices(ipunto(h),:),centro(:),distDist)
279 ! write(*,*) centro(:), distRadio
280   if (distDist .le. distRadio) then
281     contadorAdentro = contadorAdentro + 1
282     adentroCirculo(contadorAdentro) = ii
283     naristas = contadorAdentro*3
284     write(*,*) 'PUNTO ADENTRO: ', listaTriang(ii,:)
285     ntriang = ntriang - 1
286
287     !OBTENER ARISTAS
288     !Aristas de punto i con puntos del triangulo dentro del circulo
289     call obtenerAristas(ntriangMax,listaTriang,contadorAdentro,&
290                       adentroCirculo,naristas,contador)
291     write(*,*) 'CONTADOR', contador
292   else !distDis
293     contadorAfuera = contadorAfuera + 1
294     nuevosTriang(contadorAfuera,:) = listaTriang(ii,:)
295   end if !distDist
296 end do !ii
297
298 write(*,*) 'CONTADOR ADENTRO', contadorAdentro
299 write(*,*) 'CONTADOR AFUERA', contadorAfuera
300 write(*,*) 'NARISTAS', naristas
301 write(*,*) 'NTRIANG', ntriang
302
303 write(*,*) 'Triangulos que NO se tocan (afuera)', contadorAfuera

```

```
304 do ii = 1, contadorAfuera
305   write(*,*) nuevosTriang(ii,:)
306 end do !ii
307
308   !CHECAR ARISTAS REPETIDAS
309 if (contadorAdentro /= 1) then
310   contadorRepetidas = 0
311   do l = 1, naristas-1
312     do m = l+1, naristas
313       if (arista(l,1) == arista(m,1) .and. arista(l,2) == arista(m,2)) then
314         if (arista(l,1) == 0 .and. arista(m,1) == 0 &
315           .and. arista(l,2) == 0 .and. arista(m,2) == 0) then
316           exit
317         else
318           write(*,*) 'renglones repetidos', l, m
319           write(*,*) 'REPETIDAS', arista(l,:), arista(m,:)
320           contadorRepetidas = contadorRepetidas + 1
321           compartidas(contadorRepetidas,:) = arista(l,:)
322         end if ! ==0
323       end if ! lados
324     end do ! m
325   end do ! l
326
327   write(*,*) 'ARISTAS'
328   do z1 = 1, naristas
329     write(*,*) arista(z1,:)
330   end do !z1
331
332   write(*,*) 'ARISTAS REPETIDAS'
333   do z1 = 1, contadorRepetidas
334     write(*,*) compartidas(z1,:)
335   end do !z1
336
337   write(*,*) 'Triangulos que NO se tocan (afuera)'
338   do ii = 1, contadorAfuera
339     write(*,*) nuevosTriang(ii,:)
340   end do !ii
```

```
341
342 aristaSR = 0
343 q = 0
344 t = 0
345
346 !DESCARTAR ARISTAS REPETIDAS PARA HACER TRIANGULOS NUEVOS
347 do s = 1, contadorRepetidas
348   do r = 1, naristas
349     if (arista(r,1) == compartidas(s,1) .and. arista(r,2) == &
350         compartidas(s,2)) then
351       t = t + 1
352     else
353       q = q + 1
354       aristaSR(q,:) = arista(r,:)
355     end if !
356   end do !r
357   arista = aristaSR
358   q = 0
359 end do !s
360
361 naristas = naristas - t
362 arista = aristaSR
363
364 write(*,*) 'ARISTAS LIMPIAS'
365 do ii = 1, naristas
366   write(*,*) aristaSR(ii,:)
367 end do !ii
368
369 write(*,*) 'TRIANGULOS NUEVOS'
370 l2 = 0
371 do l = ntriang+1, ntriang+naristas
372   l2 = l2 + 1
373   nuevosTriang(l,1) = ipunto(h)
374   nuevosTriang(l,2) = arista(l2,1)
375   nuevosTriang(l,3) = arista(l2,2)
376   write(*,*) nuevosTriang(l,:)
377 end do !l
```

```
378
379 ntriang = ntriang + naristas
380 write(*,*) 'NUMERO DE TRIANGULOS', ntriang
381 write(*,*) 'TRIANGULOS FINALES'
382 do ii = 1, ntriang
383     listaTriang(ii,:) = nuevosTriang(ii,:)
384     write(*,*) listaTriang(ii,:)
385 end do ! ii
386
387 else !-----punto adentro de 1 solo triángulo-----
388
389 write(*,*) 'ARISTAS DE PUNTO DENTRO DE 1 SOLO TRIANGULO'
390 do z1 = 1, naristas
391     write(*,*) (arista(z1,ii), ii = 1,2)
392 end do !z1
393
394 write(*,*) 'Triangulos que NO SE TOCAN (afuera)', contadorAfuera
395 do ii = 1, contadorAfuera
396     write(*,*) nuevosTriang(ii,:)
397 end do !ii
398
399 !AGREGAR NUEVOS TRIANGULOS PUNTO ADENTRO DE 1 TRIANGULO
400 l6 = 0
401 do l5 = ntriang+1, ntriang+naristas
402     l6 = l6 + 1
403     nuevosTriang(l5,1) = ipunto(h)
404     nuevosTriang(l5,2) = arista(l6,1)
405     nuevosTriang(l5,3) = arista(l6,2)
406 end do !l5
407
408 write(*,*) 'TRIANGULOS NUEVOS'
409 do ii = contadorAfuera+1, contadorAfuera+3
410     write(*,*) nuevosTriang(ii,:)
411 end do !ii
412
413 ntriang = ntriang + naristas
414
```

```
415 do m1 = 1, ntriang
416     listaTriang(m1,:) = nuevosTriang(m1,:)
417 end do ! m1
418
419 write(*,*) 'TRIANGULOS FINALES'
420 do ii = 1, ntriang
421     write(*,*) listaTriang(ii,:)
422 end do !ii
423
424 end if ! contadorAdentro /= 0
425
426 end do !h
427
428 contadorAppend = 0
429
430 !QUITAR TRIANGULOS QUE ESTEN UNIDOS AL SUPERTRIANGULO
431
432 do j = 1, ntriang
433     if(listaTriang(j,2) == npuntos+1 .or. listaTriang(j,2) == npuntos+2 &
434         .or. listaTriang(j,2) == npuntos+3) then
435         cycle
436     else
437         contadorST = contadorST + 1
438         quitarST(contadorST,:) = listaTriang(j,:)
439     end if
440 end do !j
441
442 do j = 1, contadorST
443     if(quitarST(j,3) == npuntos+1 .or. quitarST(j,3) == npuntos+2 &
444         .or. quitarST(j,3) == npuntos+3) then
445         cycle
446     else
447         contadorFinal = contadorFinal + 1
448         finales(contadorFinal,:) = quitarST(j,:)
449     end if
450 end do !j
451
```

```
452 write(*,*) 'TRIANGULOS FINALES', contadorFinal
453 do q = 1, contadorFinal
454   write(*,*) finales(q,:)
455 end do !q
456
457 deallocate (vertices, listaTriang, adentroCirculo, nuevosTriang, arista, &
458            compartidas, quitarST, finales, aristaSR)
459
460 end subroutine pruebaPaso1
461 !-----
462 subroutine puntoAdentroPaso1(P1,P2,P3,puntoCentro)
463
464 use malla
465 use numerico
466
467 implicit none
468
469 real(dp)      :: P1(2), P2(2), P3(2), puntoCentro(2)
470 real(dp)      :: eps = 1.0e-6
471 real(dp)      :: m2, mx2, my2, xc, yc, m1, mx1, my1
472 real(dp)      :: dx, dy, rsqr, drsqr, r, dr
473 integer       :: PAdentro
474
475 if (abs(P2(2) - P1(2)).lt.eps) then
476   m2 = -(P3(1) - P2(1))/(P3(2) - P2(2))
477   mx2 = (P2(1) + P3(1))/2.0_dp
478   my2 = (P2(2) + P3(2))/2.0_dp
479   xc = (P2(1) + P1(1))/2.0_dp
480   yc = m2 *(xc - mx2) + my2
481 else if (abs(P3(2) - P2(2)).lt.eps) then
482   m1 = -(P2(1) - P1(1))/(P2(2) - P1(2))
483   mx1 = (P2(1) + P1(1))/2.0_dp
484   my1 = (P2(2) + P1(2))/2.0_dp
485   xc = (P2(1) + P3(1))/2.0_dp
486   yc = m1 *(xc - mx1) + my1
487 else
488   m1 = -(P2(1) - P1(1))/(P2(2) - P1(2))
```

```

489  m2 = -(P3(1) - P2(1))/(P3(2) - P2(2))
490  mx1 = (P2(1) + P1(1))/2.0_dp
491  mx2 = (P2(1) + P3(1))/2.0_dp
492  my1 = (P2(2) + P1(2))/2.0_dp
493  my2 = (P2(2) + P3(2))/2.0_dp
494  xc = (m1*mx1 - m2*mx2 + my2 - my1)/(m1 - m2)
495  yc = m1 *(xc - mx1) + my1
496  end if
497
498  puntoCentro(1) = xc
499  puntoCentro(2) = yc
500
501  end subroutine puntoAdentroPaso1
502  !-----
503  subroutine distanciaPuntos(P,Q,R)
504
505  use malla
506  use numerico
507
508  implicit none
509
510  real(dp)      :: R, P(2), Q(2)
511
512  R = sqrt((P(1) - Q(1))**2 + (P(2) - Q(2))**2)
513
514  end subroutine distanciaPuntos
515  !-----
516  subroutine obtenerAristas(totalTriangulos,triangulos,contAdentroSub, &
517                          circuloAdentro,numLados,contadorAr)
518
519  use numerico
520  use malla
521
522  implicit none
523
524  integer      :: totalTriangulos, triangulos(totalTriangulos,3)
525  integer      :: i, y, z, circuloAdentro(totalTriangulos), contAdentroSub

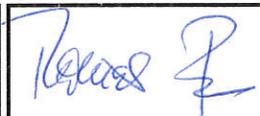
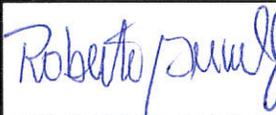
```

```
526 integer      :: numLados !naristas
527 integer      :: contadorAr !contador
528
529 do z = 1, 3
530   if (z == 1) then
531     if (triangulos(circuloAdentro(contAdentroSub),1) .lt. &
532         triangulos(circuloAdentro(contAdentroSub),2)) then
533       arista(z+contadorAr,1) = triangulos(circuloAdentro(contAdentroSub),1)
534       arista(z+contadorAr,2) = triangulos(circuloAdentro(contAdentroSub),2)
535     else
536       arista(z+contadorAr,1) = triangulos(circuloAdentro(contAdentroSub),2)
537       arista(z+contadorAr,2) = triangulos(circuloAdentro(contAdentroSub),1)
538     end if
539   elseif (z == 2) then
540     if (triangulos(circuloAdentro(contAdentroSub),1) .lt. &
541         triangulos(circuloAdentro(contAdentroSub),3)) then
542       arista(z+contadorAr,1) = triangulos(circuloAdentro(contAdentroSub),1)
543       arista(z+contadorAr,2) = triangulos(circuloAdentro(contAdentroSub),3)
544     else
545       arista(z+contadorAr,1) = triangulos(circuloAdentro(contAdentroSub),3)
546       arista(z+contadorAr,2) = triangulos(circuloAdentro(contAdentroSub),1)
547     end if
548   else
549     if (triangulos(circuloAdentro(contAdentroSub),2) .lt. &
550         triangulos(circuloAdentro(contAdentroSub),3)) then
551       arista(z+contadorAr,1) = triangulos(circuloAdentro(contAdentroSub),2)
552       arista(z+contadorAr,2) = triangulos(circuloAdentro(contAdentroSub),3)
553     else
554       arista(z+contadorAr,1) = triangulos(circuloAdentro(contAdentroSub),3)
555       arista(z+contadorAr,2) = triangulos(circuloAdentro(contAdentroSub),2)
556     end if
557   end if !z
558 end do !z
559 contadorAr = contadorAr + 3
560
561 end subroutine obtenerAristas
```

DR. VICTOR BARBA LÓPEZ
COORDINADOR DEL POSGRADO EN CIENCIAS
PRESENTE

Atendiendo a la solicitud para emitir DICTAMEN sobre la revisión de la TESIS titulada ***“Implementar el método de Elemento Finito a la ecuación de ADR. Aplicación al estado de Morelos”*** que presenta la alumna **Mónica Andrea Navarro Ramírez (5620170601)** para obtener el título de **Maestro en Ciencias**.

Nos permitimos informarle que nuestro voto es:

NOMBRE	DICTAMEN	FIRMA
Dr. Thomas Buhse CIQ-UAEM	Aprobado	
Dr. Jorge Rivera Noriega CINC-UAEM	Aprobado	
Dr. Roberto Bernal Jaquez UAM-Cuajimalpa	Aprobado	
Dr. Mario Murillo Tovar CIQ-UAEM	Aprobado	
Dr. Hugo Albeiro Saldarriaga Noreña CIQ-UAEM	APROBADO	